

**Kitlist**  
0.6.6

Generated by Doxygen 1.5.6

Mon Nov 16 18:30:30 2009



# Contents

<b>1</b>	<b>Kitlist Documentation</b>	<b>1</b>
1.1	Introduction . . . . .	1
1.2	Additional Documentation . . . . .	1
<b>2</b>	<b>Namespace Index</b>	<b>3</b>
2.1	Namespace List . . . . .	3
<b>3</b>	<b>Class Index</b>	<b>5</b>
3.1	Class Hierarchy . . . . .	5
<b>4</b>	<b>Class Index</b>	<b>7</b>
4.1	Class List . . . . .	7
<b>5</b>	<b>File Index</b>	<b>9</b>
5.1	File List . . . . .	9
<b>6</b>	<b>Namespace Documentation</b>	<b>11</b>
6.1	anonymous_namespace{kitlistgui.cpp} Namespace Reference . . . . .	11
6.1.1	Variable Documentation . . . . .	12
6.1.1.1	DEFAULT_FILENAME . . . . .	12
6.1.1.2	DEFAULT_FILENAME_EXTENSION . . . . .	12
6.1.1.3	DEFAULT_MAX_RECENT_FILES . . . . .	12
6.1.1.4	GCONF_KEY . . . . .	12
6.1.1.5	GCONF_KEY_CURRENT_FILENAME . . . . .	13
6.1.1.6	GCONF_KEY_MAX_RECENT_FILES . . . . .	13
6.1.1.7	GCONF_KEY_RECENT_FILES . . . . .	13
6.1.1.8	GLADE_APP_FILE . . . . .	13
6.1.1.9	item_target_custom . . . . .	13
6.1.1.10	item_target_text . . . . .	13
6.1.1.11	SB_ITEM_COUNT . . . . .	14

6.1.1.12	SB_MSG	14
6.1.1.13	SB_SAVE	14
6.1.1.14	XML_ELEMENT_ID	14
<b>7</b>	<b>Class Documentation</b>	<b>15</b>
7.1	Category Class Reference	15
7.1.1	Detailed Description	16
7.1.2	Constructor & Destructor Documentation	16
7.1.2.1	~Category	16
7.1.3	Member Function Documentation	16
7.1.3.1	set_id	16
7.1.3.2	get_id	17
7.1.3.3	set_name	17
7.1.3.4	get_name	17
7.1.3.5	add_item	17
7.1.3.6	remove_item	17
7.1.3.7	item_count	18
7.1.3.8	has_items	18
7.1.3.9	foreach_item	18
7.1.3.10	execute	18
7.1.4	Friends And Related Function Documentation	18
7.1.4.1	CategoryCompareName	18
7.1.4.2	CategoryCompareId	18
7.1.4.3	KitModel	19
7.1.5	Member Data Documentation	19
7.1.5.1	m_id	19
7.1.5.2	m_name	19
7.1.5.3	m_items	19
7.2	CategoryCompareId Class Reference	20
7.2.1	Detailed Description	20
7.2.2	Constructor & Destructor Documentation	20
7.2.2.1	CategoryCompareId	20
7.2.3	Member Function Documentation	20
7.2.3.1	operator()	20
7.3	CategoryCompareName Class Reference	21
7.3.1	Detailed Description	21
7.3.2	Constructor & Destructor Documentation	21

---

7.3.2.1	CategoryCompareName	21
7.3.3	Member Function Documentation	21
7.3.3.1	operator()	21
7.4	GuiState Class Reference	22
7.4.1	Detailed Description	22
7.4.2	Constructor & Destructor Documentation	22
7.4.2.1	GuiState	22
7.4.3	Member Function Documentation	23
7.4.3.1	is_dirty	23
7.4.3.2	set_dirty	23
7.4.3.3	is_deleted	23
7.4.3.4	set_deleted	23
7.4.3.5	set_new_flag	23
7.4.3.6	is_new	23
7.4.3.7	reset	23
7.4.4	Member Data Documentation	24
7.4.4.1	m_dirty	24
7.4.4.2	m_deleted	24
7.4.4.3	m_new	24
7.5	Item Class Reference	25
7.5.1	Detailed Description	25
7.5.2	Constructor & Destructor Documentation	26
7.5.2.1	Item	26
7.5.2.2	Item	26
7.5.3	Member Function Documentation	26
7.5.3.1	set_id	26
7.5.3.2	get_id	26
7.5.3.3	set_description	26
7.5.3.4	get_description	26
7.5.3.5	set_checked	26
7.5.3.6	get_checked	27
7.5.4	Friends And Related Function Documentation	27
7.5.4.1	ItemCompareName	27
7.5.4.2	ItemCompareId	27
7.5.4.3	operator<<	27
7.5.5	Member Data Documentation	27

7.5.5.1	<code>m_id</code>	27
7.5.5.2	<code>m_desc</code>	27
7.5.5.3	<code>m_checked</code>	27
7.6	ItemCompareId Class Reference	29
7.6.1	Detailed Description	29
7.6.2	Constructor & Destructor Documentation	29
7.6.2.1	ItemCompareId	29
7.6.3	Member Function Documentation	29
7.6.3.1	<code>operator()</code>	29
7.7	ItemCompareName Class Reference	30
7.7.1	Detailed Description	30
7.7.2	Constructor & Destructor Documentation	30
7.7.2.1	ItemCompareName	30
7.7.3	Member Function Documentation	30
7.7.3.1	<code>operator()</code>	30
7.8	ItemFunctor Class Reference	31
7.8.1	Detailed Description	31
7.8.2	Member Function Documentation	31
7.8.2.1	<code>operator()</code>	31
7.9	KitList Class Reference	32
7.9.1	Detailed Description	33
7.9.2	Constructor & Destructor Documentation	33
7.9.2.1	KitList	33
7.9.2.2	<code>~KitList</code>	33
7.9.3	Member Function Documentation	33
7.9.3.1	<code>get_dao</code>	33
7.9.3.2	<code>add_item</code>	34
7.9.3.3	<code>add_item</code>	34
7.9.3.4	<code>append_items_to_category</code>	34
7.9.3.5	<code>associate_item_with_category</code>	34
7.9.3.6	<code>list_items_start</code>	35
7.9.3.7	<code>list_item</code>	35
7.9.3.8	<code>list_items_end</code>	35
7.9.3.9	<code>list_items</code>	35
7.9.3.10	<code>on_list_item</code>	36
7.9.3.11	<code>list_items</code>	36

---

7.9.3.12	<a href="#">list_items</a>	36
7.9.3.13	<a href="#">execute</a>	36
7.9.3.14	<a href="#">tick_items</a>	36
7.9.3.15	<a href="#">tick_items</a>	36
7.9.3.16	<a href="#">tick_items</a>	37
7.9.3.17	<a href="#">list_categories</a>	37
7.9.3.18	<a href="#">new_category</a>	37
7.9.3.19	<a href="#">delete_category</a>	37
7.9.3.20	<a href="#">delete_item</a>	38
7.9.3.21	<a href="#">remove_item_from_category</a>	38
7.9.3.22	<a href="#">set_item_flag</a>	38
7.9.3.23	<a href="#">unset_item_flag</a>	38
7.9.3.24	<a href="#">set_category_flag</a>	38
7.9.3.25	<a href="#">unset_category_flag</a>	38
7.9.3.26	<a href="#">set_all_flags</a>	39
7.9.3.27	<a href="#">unset_all_flags</a>	39
7.9.4	<a href="#">Member Data Documentation</a>	39
7.9.4.1	<a href="#">m_dao</a>	39
7.10	<a href="#">KitListDao Class Reference</a>	40
7.10.1	<a href="#">Detailed Description</a>	41
7.10.2	<a href="#">Constructor &amp; Destructor Documentation</a>	41
7.10.2.1	<a href="#">KitListDao</a>	41
7.10.2.2	<a href="#">~KitListDao</a>	41
7.10.3	<a href="#">Member Function Documentation</a>	41
7.10.3.1	<a href="#">get_model</a>	41
7.10.3.2	<a href="#">save_model</a>	42
7.10.3.3	<a href="#">set_verbose</a>	42
7.10.3.4	<a href="#">is_verbose</a>	42
7.10.3.5	<a href="#">get_category</a>	42
7.10.3.6	<a href="#">get_all_items</a>	42
7.10.3.7	<a href="#">add_item</a>	43
7.10.3.8	<a href="#">add_item</a>	43
7.10.3.9	<a href="#">append_items_to_category</a>	43
7.10.3.10	<a href="#">associate_item_with_category</a>	43
7.10.3.11	<a href="#">get_categories</a>	44
7.10.3.12	<a href="#">new_category</a>	44

7.10.3.13	<code>delete_item</code>	44
7.10.3.14	<code>update_item_checked_state</code>	44
7.10.3.15	<code>remove_item_from_category</code>	44
7.10.3.16	<code>get_next_item_id</code>	45
7.10.3.17	<code>get_next_category_id</code>	45
7.10.3.18	<code>delete_category</code>	45
7.10.3.19	<code>set_item_flag</code>	45
7.10.3.20	<code>unset_item_flag</code>	45
7.10.3.21	<code>set_category_flag</code>	45
7.10.3.22	<code>unset_category_flag</code>	46
7.10.3.23	<code>set_all_flags</code>	46
7.10.3.24	<code>unset_all_flags</code>	46
7.10.3.25	<code>require_filename</code>	46
7.10.4	Member Data Documentation	46
7.10.4.1	<code>m_verbose_flag</code>	46
7.11	KitListGui Class Reference	47
7.11.1	Detailed Description	51
7.11.2	Constructor & Destructor Documentation	51
7.11.2.1	<code>KitListGui</code>	51
7.11.2.2	<code>~KitListGui</code>	51
7.11.3	Member Function Documentation	52
7.11.3.1	<code>init</code>	52
7.11.3.2	<code>get_max_recent_files</code>	52
7.11.3.3	<code>get_selected_items</code>	52
7.11.3.4	<code>add_items</code>	52
7.11.3.5	<code>close_add_item_window</code>	53
7.11.3.6	<code>cancel_add_item_window</code>	53
7.11.3.7	<code>close_add_category_window</code>	53
7.11.3.8	<code>cancel_add_category_window</code>	53
7.11.3.9	<code>get_selected_category</code>	53
7.11.3.10	<code>init_add_item_window</code>	54
7.11.3.11	<code>delete_selected_items</code>	54
7.11.3.12	<code>copy_selected_items_to_clipboard</code>	54
7.11.3.13	<code>confirm_lose_changes</code>	54
7.11.3.14	<code>update_recent_files_menu</code>	55
7.11.3.15	<code>update_recent_files</code>	55



---

7.11.3.16	<a href="#">on_delete_event</a>	55
7.11.3.17	<a href="#">on_menu_quit</a>	55
7.11.3.18	<a href="#">on_menu_file_new</a>	55
7.11.3.19	<a href="#">on_menu_file_open</a>	56
7.11.3.20	<a href="#">open_file</a>	56
7.11.3.21	<a href="#">on_menu_save</a>	56
7.11.3.22	<a href="#">on_menu_save_as</a>	56
7.11.3.23	<a href="#">on_menu_recent_file</a>	57
7.11.3.24	<a href="#">on_menu_add</a>	57
7.11.3.25	<a href="#">on_menu_delete</a>	57
7.11.3.26	<a href="#">on_menu_cut</a>	57
7.11.3.27	<a href="#">on_menu_copy</a>	57
7.11.3.28	<a href="#">on_menu_paste</a>	58
7.11.3.29	<a href="#">on_menu_show_all</a>	58
7.11.3.30	<a href="#">on_menu_show_checked</a>	58
7.11.3.31	<a href="#">on_menu_show_unchecked</a>	58
7.11.3.32	<a href="#">on_menu_select_all</a>	58
7.11.3.33	<a href="#">on_menu_check_selected</a>	58
7.11.3.34	<a href="#">on_menu_uncheck_selected</a>	59
7.11.3.35	<a href="#">on_menu_create_category</a>	59
7.11.3.36	<a href="#">on_menu_delete_category</a>	59
7.11.3.37	<a href="#">on_menu_rename_category</a>	59
7.11.3.38	<a href="#">on_menu_help_about</a>	59
7.11.3.39	<a href="#">on_menu_help_contents</a>	60
7.11.3.40	<a href="#">on_clipboard_get</a>	60
7.11.3.41	<a href="#">on_clipboard_clear</a>	60
7.11.3.42	<a href="#">on_clipboard_received</a>	60
7.11.3.43	<a href="#">on_category_change</a>	60
7.11.3.44	<a href="#">on_cell_edit</a>	61
7.11.3.45	<a href="#">choose_filename</a>	61
7.11.3.46	<a href="#">update_paste_status</a>	61
7.11.3.47	<a href="#">paste_status_received</a>	61
7.11.3.48	<a href="#">paste_from_xml</a>	62
7.11.3.49	<a href="#">refresh_item_list</a>	62
7.11.3.50	<a href="#">refresh_category_list</a>	62
7.11.3.51	<a href="#">selected_row_callback</a>	63

7.11.3.52	set_selected	63
7.11.3.53	toggle_selected	63
7.11.3.54	on_row_changed	63
7.11.3.55	update_item_count	63
7.11.3.56	raise	64
7.11.3.57	safe_open_file	64
7.11.3.58	run	64
7.11.4	Member Data Documentation	64
7.11.4.1	m_filename	64
7.11.4.2	m_clipboard_items	64
7.11.4.3	m_ignore_list_events	64
7.11.4.4	m_kit	65
7.11.4.5	m_window	65
7.11.4.6	m_window_add_item	65
7.11.4.7	m_window_add_category	65
7.11.4.8	m_entry_add_item	65
7.11.4.9	m_entry_add_category	65
7.11.4.10	m_file_save_menu_item	66
7.11.4.11	m_file_save_tool_button	66
7.11.4.12	m_recent_files_menu_item	66
7.11.4.13	m_paste_menu_item	66
7.11.4.14	m_checkbutton_add_item	66
7.11.4.15	m_category_combo	66
7.11.4.16	m_ref_category_list_store	66
7.11.4.17	m_category_cols	67
7.11.4.18	m_item_tree_view	67
7.11.4.19	m_item_cols	67
7.11.4.20	m_service	67
7.11.4.21	m_ref_item_tree_model	67
7.11.4.22	m_status_bar	67
7.11.4.23	m_state	68
7.11.4.24	m_current_cat_id	68
7.12	KitModel Class Reference	69
7.12.1	Detailed Description	70
7.12.2	Constructor & Destructor Documentation	70
7.12.2.1	KitModel	70

---

7.12.2.2	~KitModel	71
7.12.3	Member Function Documentation	71
7.12.3.1	foreach_item_iter	71
7.12.3.2	foreach_item	71
7.12.3.3	foreach_category_iter	71
7.12.3.4	foreach_category	71
7.12.3.5	find_category	71
7.12.3.6	find_item	72
7.12.3.7	get_categories	72
7.12.3.8	get_all_items	72
7.12.3.9	add_category	72
7.12.3.10	add_item	72
7.12.3.11	add_item	72
7.12.3.12	copy_items	73
7.12.3.13	filter	73
7.12.3.14	set_dirty	73
7.12.3.15	is_dirty	73
7.12.3.16	show_all	73
7.12.3.17	show_checked_only	74
7.12.3.18	show_unchecked_only	74
7.12.3.19	reset	74
7.12.3.20	purge	74
7.12.4	Member Data Documentation	74
7.12.4.1	m_dirty	74
7.12.4.2	m_category_map	75
7.12.4.3	m_item_map	75
7.12.4.4	m_item_filter	75
7.13	KitParser Class Reference	76
7.13.1	Detailed Description	77
7.13.2	Constructor & Destructor Documentation	77
7.13.2.1	KitParser	77
7.13.2.2	~KitParser	77
7.13.3	Member Function Documentation	77
7.13.3.1	process_item	77
7.13.3.2	process_category	78
7.13.3.3	process_category_item	78

---

7.13.3.4	<a href="#">on_start_document</a>	78
7.13.3.5	<a href="#">on_end_document</a>	78
7.13.3.6	<a href="#">on_start_element</a>	78
7.13.3.7	<a href="#">on_end_element</a>	79
7.13.3.8	<a href="#">on_characters</a>	79
7.13.3.9	<a href="#">on_comment</a>	79
7.13.3.10	<a href="#">on_warning</a>	79
7.13.3.11	<a href="#">on_error</a>	79
7.13.3.12	<a href="#">on_fatal_error</a>	79
7.13.4	<a href="#">Member Data Documentation</a>	80
7.13.4.1	<a href="#">m_category</a>	80
7.13.4.2	<a href="#">m_item</a>	80
7.13.4.3	<a href="#">m_cdata</a>	80
7.13.4.4	<a href="#">m_model</a>	80
7.14	<a href="#">ModelCategory Class Reference</a>	81
7.14.1	<a href="#">Detailed Description</a>	82
7.14.2	<a href="#">Constructor &amp; Destructor Documentation</a>	82
7.14.2.1	<a href="#">ModelCategory</a>	82
7.14.2.2	<a href="#">~ModelCategory</a>	82
7.14.3	<a href="#">Member Function Documentation</a>	82
7.14.3.1	<a href="#">get_model_items</a>	82
7.14.3.2	<a href="#">get_items</a>	82
7.14.3.3	<a href="#">get_removed_children</a>	82
7.14.3.4	<a href="#">get_added_children</a>	83
7.14.3.5	<a href="#">add_item</a>	83
7.14.3.6	<a href="#">remove_item</a>	83
7.14.3.7	<a href="#">remove_items</a>	83
7.14.3.8	<a href="#">reset</a>	83
7.14.3.9	<a href="#">purge</a>	84
7.14.4	<a href="#">Friends And Related Function Documentation</a>	84
7.14.4.1	<a href="#">KitModel</a>	84
7.14.5	<a href="#">Member Data Documentation</a>	84
7.14.5.1	<a href="#">m_removed_children</a>	84
7.14.5.2	<a href="#">m_added_children</a>	84
7.15	<a href="#">ModelCategoryColumns Class Reference</a>	85
7.15.1	<a href="#">Detailed Description</a>	85

---

7.15.2	Constructor & Destructor Documentation	85
7.15.2.1	ModelCategoryColumns	85
7.15.3	Member Data Documentation	85
7.15.3.1	m_col_text	85
7.15.3.2	m_col_num	85
7.16	ModelItem Class Reference	86
7.16.1	Detailed Description	86
7.16.2	Constructor & Destructor Documentation	86
7.16.2.1	ModelItem	86
7.16.3	Member Function Documentation	86
7.16.3.1	set_checked	86
7.16.4	Friends And Related Function Documentation	87
7.16.4.1	ModelItemCompareId	87
7.17	ModelItemColumns Class Reference	88
7.17.1	Detailed Description	88
7.17.2	Constructor & Destructor Documentation	88
7.17.2.1	ModelItemColumns	88
7.17.3	Member Data Documentation	88
7.17.3.1	m_col_text	88
7.17.3.2	m_col_checked	88
7.17.3.3	m_col_num	88
7.18	ModelItemCompareId Class Reference	90
7.18.1	Detailed Description	90
7.18.2	Constructor & Destructor Documentation	90
7.18.2.1	ModelItemCompareId	90
7.18.3	Member Function Documentation	90
7.18.3.1	operator()	90
7.19	Service Class Reference	91
7.19.1	Detailed Description	92
7.19.2	Constructor & Destructor Documentation	93
7.19.2.1	Service	93
7.19.2.2	~Service	93
7.19.3	Member Function Documentation	93
7.19.3.1	load_model	93
7.19.3.2	get_next_item_id	93
7.19.3.3	get_next_category_id	93

7.19.3.4	find_item	93
7.19.3.5	find_category	94
7.19.3.6	copy_items	94
7.19.3.7	create_item	94
7.19.3.8	delete_item	94
7.19.3.9	delete_category	94
7.19.3.10	create_category	95
7.19.3.11	is_model_dirty	95
7.19.3.12	set_model_dirty	95
7.19.3.13	filter	95
7.19.3.14	create_default_model	96
7.19.3.15	open_as_xml	96
7.19.3.16	save	96
7.19.3.17	save_as_xml	96
7.19.3.18	update_item	97
7.19.3.19	get_items	97
7.19.3.20	get_categories	97
7.19.3.21	show_all	97
7.19.3.22	show_checked_only	98
7.19.3.23	show_unchecked_only	98
7.19.3.24	select_items	98
7.19.3.25	toggle_selected_items	98
7.19.3.26	require_filename	98
7.19.4	Member Data Documentation	99
7.19.4.1	m_dao	99
7.19.4.2	m_model	99
7.20	TickItem Class Reference	100
7.20.1	Detailed Description	100
7.20.2	Constructor & Destructor Documentation	100
7.20.2.1	TickItem	100
7.20.3	Member Function Documentation	100
7.20.3.1	operator()	100
7.20.4	Member Data Documentation	100
7.20.4.1	m_changed	100
7.21	XmlDao Class Reference	102
7.21.1	Detailed Description	104

---

7.21.2	Constructor & Destructor Documentation	104
7.21.2.1	XmlDao	104
7.21.3	Member Function Documentation	104
7.21.3.1	add_item_to_dom	104
7.21.3.2	add_category_item_to_dom	104
7.21.3.3	add_category_to_dom	104
7.21.3.4	get_model	105
7.21.3.5	get_model	105
7.21.3.6	save_model	105
7.21.3.7	save_model	105
7.21.3.8	get_category	106
7.21.3.9	get_all_items	106
7.21.3.10	add_item	106
7.21.3.11	add_item	106
7.21.3.12	append_items_to_category	107
7.21.3.13	associate_item_with_category	107
7.21.3.14	get_categories	107
7.21.3.15	new_category	107
7.21.3.16	delete_item	108
7.21.3.17	update_item_checked_state	108
7.21.3.18	remove_item_from_category	108
7.21.3.19	get_next_item_id	108
7.21.3.20	get_next_category_id	109
7.21.3.21	delete_category	109
7.21.3.22	set_item_flag	109
7.21.3.23	unset_item_flag	109
7.21.3.24	set_category_flag	109
7.21.3.25	unset_category_flag	109
7.21.3.26	set_all_flags	110
7.21.3.27	unset_all_flags	110
7.21.3.28	set_filename	110
7.21.3.29	require_filename	110
7.21.4	Member Data Documentation	110
7.21.4.1	m_filename	110
7.21.4.2	m_items_node	111
7.21.4.3	m_categories_node	111

7.21.4.4	<code>m_cat_items_node</code>	111
7.21.4.5	<code>m_max_item_id</code>	111
7.21.4.6	<code>m_max_category_id</code>	111
<b>8</b>	<b>File Documentation</b>	<b>113</b>
8.1	<code>category.cpp</code> File Reference	113
8.2	<code>category.hpp</code> File Reference	114
8.2.1	Define Documentation	114
8.2.1.1	<code>CATEGORY_H</code>	114
8.2.2	Typedef Documentation	114
8.2.2.1	<code>CategoryContainer</code>	114
8.2.2.2	<code>CategoryIter</code>	114
8.3	<code>item.hpp</code> File Reference	115
8.3.1	Define Documentation	115
8.3.1.1	<code>ITEM_H</code>	115
8.3.2	Typedef Documentation	115
8.3.2.1	<code>ItemContainer</code>	115
8.3.2.2	<code>ItemIter</code>	116
8.3.2.3	<code>ItemList</code>	116
8.3.2.4	<code>ItemListIter</code>	116
8.3.2.5	<code>SlotForeachItem</code>	116
8.4	<code>kitlistdao.hpp</code> File Reference	117
8.4.1	Define Documentation	117
8.4.1.1	<code>KIT_LIST_DAO_H</code>	117
8.4.2	Enumeration Type Documentation	117
8.4.2.1	<code>item_choice</code>	117
8.5	<code>kitlistgui.cpp</code> File Reference	118
8.5.1	Define Documentation	120
8.5.1.1	<code>KITLIST_SERVICE_IFACE</code>	120
8.5.1.2	<code>KITLIST_SERVICE_NAME</code>	120
8.5.1.3	<code>KITLIST_SERVICE_OBJECT</code>	120
8.5.2	Typedef Documentation	120
8.5.2.1	<code>type_children</code>	120
8.5.3	Function Documentation	120
8.5.3.1	<code>file_exists</code>	120
8.5.3.2	<code>get_glade_ref_ptr</code>	121
8.5.3.3	<code>load_resource_glade_file</code>	121



---

8.6	kitlistgui.hpp File Reference	122
8.6.1	Define Documentation	123
8.6.1.1	KIT_LIST_GUI_H	123
8.6.2	Enumeration Type Documentation	123
8.6.2.1	gui_state	123
8.6.3	Variable Documentation	123
8.6.3.1	CHECKED_COL_POSITION	123
8.7	kitlistpgsqldao.cpp File Reference	124
8.8	kitlistpgsqldao.hpp File Reference	125
8.8.1	Define Documentation	125
8.8.1.1	KIT_LIST_PGSQL_DAO_H	125
8.9	kitmodel.cpp File Reference	126
8.10	kitmodel.hpp File Reference	127
8.10.1	Define Documentation	128
8.10.1.1	KIT_MODEL_H	128
8.10.2	Typedef Documentation	128
8.10.2.1	CategoryMap	128
8.10.2.2	CategoryMapIter	128
8.10.2.3	ItemMap	128
8.10.2.4	ItemMapIter	128
8.10.2.5	ModelCategoryContainer	128
8.10.2.6	ModelCategoryIter	128
8.10.2.7	ModelItemContainer	128
8.10.2.8	ModelItemIter	128
8.10.2.9	SlotForeachCategory	128
8.10.2.10	SlotForeachCategoryIter	128
8.10.2.11	SlotForeachModelItem	129
8.10.2.12	SlotForeachModelItemIter	129
8.10.3	Enumeration Type Documentation	129
8.10.3.1	item_filter_types	129
8.11	kitparser.cpp File Reference	130
8.12	kitparser.hpp File Reference	131
8.12.1	Define Documentation	131
8.12.1.1	KIT_PARSER_H	131
8.13	main.cpp File Reference	132
8.13.1	Function Documentation	132

---

8.13.1.1	main	132
8.13.2	Variable Documentation	133
8.13.2.1	html_flag	133
8.13.2.2	verbose_flag	133
8.14	service.cpp File Reference	134
8.15	service.hpp File Reference	135
8.15.1	Define Documentation	135
8.15.1.1	SERVICE_H	135
8.16	xmldao.cpp File Reference	136
8.17	xmldao.hpp File Reference	137
8.17.1	Define Documentation	137
8.17.1.1	NYI	137
8.17.1.2	XMLDAO_H	137

# 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:

<a href="#">anonymous_namespace{kitlistgui.cpp}</a> . . . . .	11
---	----



# Chapter 3

## Class Index

### 3.1 Class Hierarchy

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

Category . . . . .	15
ModelCategory . . . . .	81
CategoryCompareId . . . . .	20
CategoryCompareName . . . . .	21
GuiState . . . . .	22
ModelCategory . . . . .	81
ModelItem . . . . .	86
Item . . . . .	25
ModelItem . . . . .	86
ItemCompareId . . . . .	29
ItemCompareName . . . . .	30
ItemFunctor . . . . .	31
TickItem . . . . .	100
KitList . . . . .	32
KitListDao . . . . .	40
XmlDao . . . . .	102
KitListGui . . . . .	47
KitModel . . . . .	69
KitParser . . . . .	76
ModelCategoryColumns . . . . .	85
ModelItemColumns . . . . .	88
ModelItemCompareId . . . . .	90
Service . . . . .	91





# Chapter 4

## Class Index

### 4.1 Class List

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

<a href="#">Category</a> (Represents a <a href="#">Category</a> ) . . . . .	15
<a href="#">CategoryCompareId</a> (Comparator used for comparing <a href="#">Categories</a> by id) . . . . .	20
<a href="#">CategoryCompareName</a> (Comparator used for sorting <a href="#">Categories</a> by name) . . . . .	21
<a href="#">GuiState</a> (Class encapsulating state of an object in the data model) . . . . .	22
<a href="#">Item</a> (Represents an <a href="#">Item</a> ) . . . . .	25
<a href="#">ItemCompareId</a> (Comparator used for comparing <a href="#">Items</a> by id) . . . . .	29
<a href="#">ItemCompareName</a> (Comparator used for sorting <a href="#">Items</a> by name) . . . . .	30
<a href="#">ItemFunctor</a> (Functor for processing items) . . . . .	31
<a href="#">KitList</a> (Main application class) . . . . .	32
<a href="#">KitListDao</a> (Defines the methods that an implementation of this class must implement) . . . . .	40
<a href="#">KitListGui</a> (Encapsulates the methods for the application's GUI front end) . . . . .	47
<a href="#">KitModel</a> (Holds a rich graph of objects representing the application's data model) . . . . .	69
<a href="#">KitParser</a> (SaxParser implementation for reading the <a href="#">KitModel</a> from an XML document) . . . . .	76
<a href="#">ModelCategory</a> (Represents a <a href="#">Category</a> combined with <a href="#">GuiState</a> attributes) . . . . .	81
<a href="#">ModelCategoryColumns</a> (A definition for displaying a <a href="#">ModelCategory</a> in a combo box) . . . . .	85
<a href="#">ModelItem</a> (Represents an <a href="#">Item</a> combined with <a href="#">GuiState</a> attributes) . . . . .	86
<a href="#">ModelItemColumns</a> (A definition for displaying an item in a multi-column list) . . . . .	88
<a href="#">ModelItemCompareId</a> (Comparator for comparing items by their unique ID) . . . . .	90
<a href="#">Service</a> (Business/service layer implementation) . . . . .	91
<a href="#">TickItem</a> . . . . .	100
<a href="#">XmlDao</a> (Implementation of a <a href="#">KitListDao</a> using XML as the persistence store) . . . . .	102



# Chapter 5

## File Index

### 5.1 File List

Here is a list of all files with brief descriptions:

<a href="#">category.cpp</a>	113
<a href="#">category.hpp</a>	114
<a href="#">item.hpp</a>	115
<a href="#">kitlistdao.hpp</a>	117
<a href="#">kitlistgui.cpp</a>	118
<a href="#">kitlistgui.hpp</a>	122
<a href="#">kitlistpgsqldao.cpp</a>	124
<a href="#">kitlistpgsqldao.hpp</a>	125
<a href="#">kitmodel.cpp</a>	126
<a href="#">kitmodel.hpp</a>	127
<a href="#">kitparser.cpp</a>	130
<a href="#">kitparser.hpp</a>	131
<a href="#">main.cpp</a>	132
<a href="#">service.cpp</a>	134
<a href="#">service.hpp</a>	135
<a href="#">xmldao.cpp</a>	136
<a href="#">xmldao.hpp</a>	137



# 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 bar 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 clipboard 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 quint 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 quint anonymous\_namespace{kitlistgui.cpp}::SB\_MSG = SB\_SAVE + 1**

Status bar 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 quint 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 clipboard 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

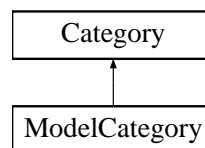
## Class Documentation

### 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:

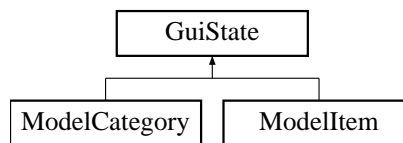
- [category.hpp](#)

## 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 identify 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:

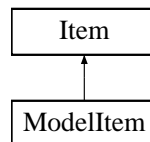
- [kitmodel.hpp](#)
- [kitmodel.cpp](#)

## 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](#)

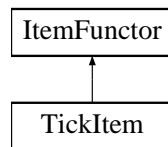


## 7.8 ItemFuncor Class Reference

Funcor for processing items.

```
#include <item.hpp>
```

Inheritance diagram for ItemFuncor::



### Public Member Functions

- virtual bool [operator\(\)](#) ([Item](#) &item)=0

#### 7.8.1 Detailed Description

Funcor for processing items.

Intended to have the [operator\(\)](#) overridden by an actual implementation.

Definition at line 85 of file [item.hpp](#).

#### 7.8.2 Member Function Documentation

##### 7.8.2.1 virtual bool [ItemFuncor::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, [ItemFuncor](#) &funcor)
- 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:

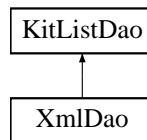
- [main.cpp](#)

## 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 if 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, and 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 its 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** `ModellItem * 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 (ModellItem * 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 (ModellItem * 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:

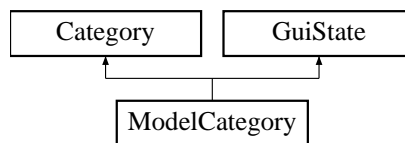
- [kitparser.hpp](#)
- [kitparser.cpp](#)

## 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:

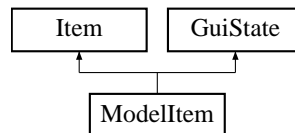
- [kitlistgui.hpp](#)

## 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](#) ()
- [ModellItem](#) \* [find\\_item](#) (long id)
- [ModelCategory](#) \* [find\\_category](#) (long cat\_id)
- void [copy\\_items](#) (const [ModellItemContainer](#) &items, long cat\_id)  
*Copies items to the specified category.*
- [Item](#) \* [create\\_item](#) (long cat\_id)  
*Creates a new [ModellItem](#) 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 persistence 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](#) implementation.

##### 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 persistence 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`.

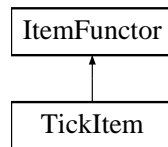
Referenced by `copy_items()`, `create_category()`, `create_default_model()`, `create_item()`, `delete_category()`, `delete_item()`, `filter()`, `find_category()`, `find_item()`, `get_categories()`, `get_items()`, `is_model_dirty()`, `open_as_xml()`, `save()`, `save_as_xml()`, `select_items()`, `Service()`, `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:

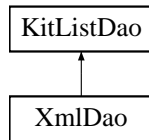
- [main.cpp](#)

## 7.21 XmlDao Class Reference

Implementation of a [KitListDao](#) using XML as the persistence store.

```
#include <xml dao.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/uststring.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 bar 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 clipboard 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

- ~Category
  - Category, 16
- ~KitList
  - KitList, 33
- ~KitListDao
  - KitListDao, 41
- ~KitListGui
  - KitListGui, 51
- ~KitModel
  - KitModel, 70
- ~KitParser
  - KitParser, 77
- ~ModelCategory
  - ModelCategory, 82
- ~Service
  - Service, 93
- ADD\_CATEGORY
  - kitlistgui.hpp, 123
- add\_category
  - KitModel, 72
- add\_category\_item\_to\_dom
  - XmlDao, 104
- add\_category\_to\_dom
  - XmlDao, 104
- add\_item
  - Category, 17
  - KitList, 33, 34
  - KitListDao, 43
  - KitModel, 72
  - ModelCategory, 83
  - XmlDao, 106
- add\_item\_to\_dom
  - XmlDao, 104
- add\_items
  - KitListGui, 52
- ALL
  - kitmodel.hpp, 129
- ALL\_ITEMS
  - kitlistdao.hpp, 117
- anonymous\_namespace{kitlistgui.cpp}, 11
  - DEFAULT\_FILENAME, 12
  - DEFAULT\_FILENAME\_EXTENSION, 12
  - DEFAULT\_MAX\_RECENT\_FILES, 12
  - GCONF\_KEY, 12
  - GCONF\_KEY\_CURRENT\_FILENAME, 12
  - GCONF\_KEY\_MAX\_RECENT\_FILES, 13
  - GCONF\_KEY\_RECENT\_FILES, 13
  - GLADE\_APP\_FILE, 13
  - item\_target\_custom, 13
  - item\_target\_text, 13
  - SB\_ITEM\_COUNT, 13
  - SB\_MSG, 14
  - SB\_SAVE, 14
  - XML\_ELEMENT\_ID, 14
- append\_items\_to\_category
  - KitList, 34
  - KitListDao, 43
  - XmlDao, 107
- associate\_item\_with\_category
  - KitList, 34
  - KitListDao, 43
  - XmlDao, 107
- cancel\_add\_category\_window
  - KitListGui, 53
- cancel\_add\_item\_window
  - KitListGui, 53
- Category, 15
  - ~Category, 16
  - add\_item, 17
  - CategoryCompareId, 18
  - CategoryCompareName, 18
  - execute, 18
  - foreach\_item, 18
  - get\_id, 16
  - get\_name, 17
  - has\_items, 18
  - item\_count, 17
  - KitModel, 18
  - m\_id, 19
  - m\_items, 19
  - m\_name, 19
  - remove\_item, 17
  - set\_id, 16
  - set\_name, 17
- category.cpp, 113
- category.hpp, 114
  - CATEGORY\_H, 114
  - CategoryContainer, 114

- CategoryIter, 114
- CATEGORY\_H
  - category.hpp, 114
- CategoryCompareId, 20
  - Category, 18
  - CategoryCompareId, 20
  - operator(), 20
- CategoryCompareName, 21
  - Category, 18
  - CategoryCompareName, 21
  - operator(), 21
- CategoryContainer
  - category.hpp, 114
- CategoryIter
  - category.hpp, 114
- CategoryMap
  - kitmodel.hpp, 128
- CategoryMapIter
  - kitmodel.hpp, 128
- CHECKED
  - kitmodel.hpp, 129
- CHECKED\_ITEMS
  - kitlistdao.hpp, 117
- CHECKED\_COL\_POSITION
  - kitlistgui.hpp, 123
- choose\_filename
  - KitListGui, 61
- close\_add\_category\_window
  - KitListGui, 53
- close\_add\_item\_window
  - KitListGui, 53
- confirm\_lose\_changes
  - KitListGui, 54
- copy\_items
  - KitModel, 73
  - Service, 94
- copy\_selected\_items\_to\_clipboard
  - KitListGui, 54
- create\_category
  - Service, 95
- create\_default\_model
  - Service, 96
- create\_item
  - Service, 94
- DEFAULT\_FILENAME
  - anonymous\_namespace{kitlistgui.cpp}, 12
- DEFAULT\_FILENAME\_EXTENSION
  - anonymous\_namespace{kitlistgui.cpp}, 12
- DEFAULT\_MAX\_RECENT\_FILES
  - anonymous\_namespace{kitlistgui.cpp}, 12
- delete\_category
  - KitList, 37
  - KitListDao, 45
- Service, 94
- XmlDao, 109
- delete\_item
  - KitList, 37
  - KitListDao, 44
  - Service, 94
  - XmlDao, 108
- delete\_selected\_items
  - KitListGui, 54
- execute
  - Category, 18
  - KitList, 36
- file\_exists
  - kitlistgui.cpp, 120
- filter
  - KitModel, 73
  - Service, 95
- find\_category
  - KitModel, 71
  - Service, 93
- find\_item
  - KitModel, 71
  - Service, 93
- foreach\_category
  - KitModel, 71
- foreach\_category\_iter
  - KitModel, 71
- foreach\_item
  - Category, 18
  - KitModel, 71
- foreach\_item\_iter
  - KitModel, 71
- GCONF\_KEY
  - anonymous\_namespace{kitlistgui.cpp}, 12
- GCONF\_KEY\_CURRENT\_FILENAME
  - anonymous\_namespace{kitlistgui.cpp}, 12
- GCONF\_KEY\_MAX\_RECENT\_FILES
  - anonymous\_namespace{kitlistgui.cpp}, 13
- GCONF\_KEY\_RECENT\_FILES
  - anonymous\_namespace{kitlistgui.cpp}, 13
- get\_added\_children
  - ModelCategory, 82
- get\_all\_items
  - KitListDao, 42
  - KitModel, 72
  - XmlDao, 106
- get\_categories
  - KitListDao, 44
  - KitModel, 72
  - Service, 97
  - XmlDao, 107

- get\_category
  - KitListDao, 42
  - XmlDao, 106
- get\_checked
  - Item, 27
- get\_dao
  - KitList, 33
- get\_description
  - Item, 26
- get\_glade\_ref\_ptr
  - kitlistgui.cpp, 121
- get\_id
  - Category, 16
  - Item, 26
- get\_items
  - ModelCategory, 82
  - Service, 97
- get\_max\_recent\_files
  - KitListGui, 52
- get\_model
  - KitListDao, 41
  - XmlDao, 104, 105
- get\_model\_items
  - ModelCategory, 82
- get\_name
  - Category, 17
- get\_next\_category\_id
  - KitListDao, 45
  - Service, 93
  - XmlDao, 108
- get\_next\_item\_id
  - KitListDao, 45
  - Service, 93
  - XmlDao, 108
- get\_removed\_children
  - ModelCategory, 82
- get\_selected\_category
  - KitListGui, 53
- get\_selected\_items
  - KitListGui, 52
- GLADE\_APP\_FILE
  - anonymous\_namespace{kitlistgui.cpp}, 13
- gui\_state
  - kitlistgui.hpp, 123
- GuiState, 22
  - GuiState, 22
  - is\_deleted, 23
  - is\_dirty, 23
  - is\_new, 23
  - m\_deleted, 24
  - m\_dirty, 24
  - m\_new, 24
  - reset, 23
  - set\_deleted, 23
  - set\_dirty, 23
  - set\_new\_flag, 23
- has\_items
  - Category, 18
- html\_flag
  - main.cpp, 133
- init
  - KitListGui, 52
- init\_add\_item\_window
  - KitListGui, 54
- is\_deleted
  - GuiState, 23
- is\_dirty
  - GuiState, 23
  - KitModel, 73
- is\_model\_dirty
  - Service, 95
- is\_new
  - GuiState, 23
- is\_verbose
  - KitListDao, 42
- Item, 25
  - get\_checked, 27
  - get\_description, 26
  - get\_id, 26
  - Item, 26
  - ItemCompareId, 27
  - ItemCompareName, 27
  - m\_checked, 27
  - m\_desc, 27
  - m\_id, 27
  - operator<<, 27
  - set\_checked, 26
  - set\_description, 26
  - set\_id, 26
- item.hpp, 115
  - ITEM\_H, 115
  - ItemContainer, 115
  - ItemIter, 115
  - ItemList, 116
  - ItemListIter, 116
  - SlotForeachItem, 116
- item\_choice
  - kitlistdao.hpp, 117
- item\_count
  - Category, 17
- item\_filter\_types
  - kitmodel.hpp, 129
- ITEM\_H
  - item.hpp, 115
- item\_target\_custom
  - anonymous\_namespace{kitlistgui.cpp}, 13

- item\_target\_text
  - anonymous\_namespace{kitlistgui.cpp}, 13
- ItemCompareId, 29
  - Item, 27
  - ItemCompareId, 29
  - operator(), 29
- ItemCompareName, 30
  - Item, 27
  - ItemCompareName, 30
  - operator(), 30
- ItemContainer
  - item.hpp, 115
- ItemFunctor, 31
  - operator(), 31
- ItemIter
  - item.hpp, 115
- ItemList
  - item.hpp, 116
- ItemListIter
  - item.hpp, 116
- ItemMap
  - kitmodel.hpp, 128
- ItemMapIter
  - kitmodel.hpp, 128
- KIT\_LIST\_DAO\_H
  - kitlistdao.hpp, 117
- KIT\_LIST\_GUI\_H
  - kitlistgui.hpp, 123
- KIT\_LIST\_PGSQL\_DAO\_H
  - kitlistpgsqldao.hpp, 125
- KIT\_MODEL\_H
  - kitmodel.hpp, 128
- KIT\_PARSER\_H
  - kitparser.hpp, 131
- KitList, 32
  - ~KitList, 33
  - add\_item, 33, 34
  - append\_items\_to\_category, 34
  - associate\_item\_with\_category, 34
  - delete\_category, 37
  - delete\_item, 37
  - execute, 36
  - get\_dao, 33
  - KitList, 33
  - list\_categories, 37
  - list\_item, 35
  - list\_items, 35, 36
  - list\_items\_end, 35
  - list\_items\_start, 34
  - m\_dao, 39
  - new\_category, 37
  - on\_list\_item, 35
  - remove\_item\_from\_category, 38
  - set\_all\_flags, 39
  - set\_category\_flag, 38
  - set\_item\_flag, 38
  - tick\_items, 36, 37
  - unset\_all\_flags, 39
  - unset\_category\_flag, 38
  - unset\_item\_flag, 38
- KITLIST\_SERVICE\_IFACE
  - kitlistgui.cpp, 120
- KITLIST\_SERVICE\_NAME
  - kitlistgui.cpp, 120
- KITLIST\_SERVICE\_OBJECT
  - kitlistgui.cpp, 120
- KitListDao, 40
  - ~KitListDao, 41
  - add\_item, 43
  - append\_items\_to\_category, 43
  - associate\_item\_with\_category, 43
  - delete\_category, 45
  - delete\_item, 44
  - get\_all\_items, 42
  - get\_categories, 44
  - get\_category, 42
  - get\_model, 41
  - get\_next\_category\_id, 45
  - get\_next\_item\_id, 45
  - is\_verbose, 42
  - KitListDao, 41
  - m\_verbose\_flag, 46
  - new\_category, 44
  - remove\_item\_from\_category, 44
  - require\_filename, 46
  - save\_model, 42
  - set\_all\_flags, 46
  - set\_category\_flag, 45
  - set\_item\_flag, 45
  - set\_verbose, 42
  - unset\_all\_flags, 46
  - unset\_category\_flag, 45
  - unset\_item\_flag, 45
  - update\_item\_checked\_state, 44
- kitlistdao.hpp, 117
  - ALL\_ITEMS, 117
  - CHECKED\_ITEMS, 117
  - item\_choice, 117
  - KIT\_LIST\_DAO\_H, 117
  - UNCHECKED\_ITEMS, 117
- KitListGui, 47
  - ~KitListGui, 51
  - add\_items, 52
  - cancel\_add\_category\_window, 53
  - cancel\_add\_item\_window, 53
  - choose\_filename, 61
  - close\_add\_category\_window, 53

- close\_add\_item\_window, 53
- confirm\_lose\_changes, 54
- copy\_selected\_items\_to\_clipboard, 54
- delete\_selected\_items, 54
- get\_max\_recent\_files, 52
- get\_selected\_category, 53
- get\_selected\_items, 52
- init, 52
- init\_add\_item\_window, 54
- KitListGui, 51
- m\_category\_cols, 66
- m\_category\_combo, 66
- m\_checkbutton\_add\_item, 66
- m\_clipboard\_items, 64
- m\_current\_cat\_id, 68
- m\_entry\_add\_category, 65
- m\_entry\_add\_item, 65
- m\_file\_save\_menu\_item, 65
- m\_file\_save\_tool\_button, 66
- m\_filename, 64
- m\_ignore\_list\_events, 64
- m\_item\_cols, 67
- m\_item\_tree\_view, 67
- m\_kit, 65
- m\_paste\_menu\_item, 66
- m\_recent\_files\_menu\_item, 66
- m\_ref\_category\_list\_store, 66
- m\_ref\_item\_tree\_model, 67
- m\_service, 67
- m\_state, 67
- m\_status\_bar, 67
- m\_window, 65
- m\_window\_add\_category, 65
- m\_window\_add\_item, 65
- on\_category\_change, 60
- on\_cell\_edit, 61
- on\_clipboard\_clear, 60
- on\_clipboard\_get, 60
- on\_clipboard\_received, 60
- on\_delete\_event, 55
- on\_menu\_add, 57
- on\_menu\_check\_selected, 58
- on\_menu\_copy, 57
- on\_menu\_create\_category, 59
- on\_menu\_cut, 57
- on\_menu\_delete, 57
- on\_menu\_delete\_category, 59
- on\_menu\_file\_new, 55
- on\_menu\_file\_open, 56
- on\_menu\_help\_about, 59
- on\_menu\_help\_contents, 60
- on\_menu\_paste, 57
- on\_menu\_quit, 55
- on\_menu\_recent\_file, 56
- on\_menu\_rename\_category, 59
- on\_menu\_save, 56
- on\_menu\_save\_as, 56
- on\_menu\_select\_all, 58
- on\_menu\_show\_all, 58
- on\_menu\_show\_checked, 58
- on\_menu\_show\_unchecked, 58
- on\_menu\_uncheck\_selected, 59
- on\_row\_changed, 63
- open\_file, 56
- paste\_from\_xml, 62
- paste\_status\_received, 61
- raise, 63
- refresh\_category\_list, 62
- refresh\_item\_list, 62
- run, 64
- safe\_open\_file, 64
- selected\_row\_callback, 62
- set\_selected, 63
- toggle\_selected, 63
- update\_item\_count, 63
- update\_paste\_status, 61
- update\_recent\_files, 55
- update\_recent\_files\_menu, 54
- kitlistgui.cpp, 118
  - file\_exists, 120
  - get\_glade\_ref\_ptr, 121
  - KITLIST\_SERVICE\_IFACE, 120
  - KITLIST\_SERVICE\_NAME, 120
  - KITLIST\_SERVICE\_OBJECT, 120
  - load\_resource\_glade\_file, 121
  - type\_children, 120
- kitlistgui.hpp, 122
  - ADD\_CATEGORY, 123
  - CHECKED\_COL\_POSITION, 123
  - gui\_state, 123
  - KIT\_LIST\_GUI\_H, 123
  - RENAME\_CATEGORY, 123
- kitlistpgsdao.cpp, 124
- kitlistpgsdao.hpp, 125
  - KIT\_LIST\_PGSQL\_DAO\_H, 125
- KitModel, 69
  - ~KitModel, 70
  - add\_category, 72
  - add\_item, 72
  - Category, 18
  - copy\_items, 73
  - filter, 73
  - find\_category, 71
  - find\_item, 71
  - foreach\_category, 71
  - foreach\_category\_iter, 71
  - foreach\_item, 71
  - foreach\_item\_iter, 71



- get\_all\_items, 72
- get\_categories, 72
- is\_dirty, 73
- KitModel, 70
- m\_category\_map, 75
- m\_dirty, 74
- m\_item\_filter, 75
- m\_item\_map, 75
- ModelCategory, 84
- purge, 74
- reset, 74
- set\_dirty, 73
- show\_all, 73
- show\_checked\_only, 74
- show\_unchecked\_only, 74
- kitmodel.cpp, 126
- kitmodel.hpp, 127
  - ALL, 129
  - CategoryMap, 128
  - CategoryMapIter, 128
  - CHECKED, 129
  - item\_filter\_types, 129
  - ItemMap, 128
  - ItemMapIter, 128
  - KIT\_MODEL\_H, 128
  - ModelCategoryContainer, 128
  - ModelCategoryIter, 128
  - ModelItemContainer, 128
  - ModelItemIter, 128
  - SlotForEachCategory, 128
  - SlotForEachCategoryIter, 128
  - SlotForEachModelItem, 128
  - SlotForEachModelItemIter, 129
  - UNCHECKED, 129
- KitParser, 76
  - ~KitParser, 77
  - KitParser, 77
  - m\_category, 80
  - m\_cdata, 80
  - m\_item, 80
  - m\_model, 80
  - on\_characters, 79
  - on\_comment, 79
  - on\_end\_document, 78
  - on\_end\_element, 78
  - on\_error, 79
  - on\_fatal\_error, 79
  - on\_start\_document, 78
  - on\_start\_element, 78
  - on\_warning, 79
  - process\_category, 78
  - process\_category\_item, 78
  - process\_item, 77
- kitparser.cpp, 130
- kitparser.hpp, 131
  - KIT\_PARSER\_H, 131
- list\_categories
  - KitList, 37
- list\_item
  - KitList, 35
- list\_items
  - KitList, 35, 36
- list\_items\_end
  - KitList, 35
- list\_items\_start
  - KitList, 34
- load\_model
  - Service, 93
- load\_resource\_glade\_file
  - kitlistgui.cpp, 121
- m\_added\_children
  - ModelCategory, 84
- m\_cat\_items\_node
  - XmlDao, 111
- m\_categories\_node
  - XmlDao, 111
- m\_category
  - KitParser, 80
- m\_category\_cols
  - KitListGui, 66
- m\_category\_combo
  - KitListGui, 66
- m\_category\_map
  - KitModel, 75
- m\_cdata
  - KitParser, 80
- m\_changed
  - TickItem, 100
- m\_checkbutton\_add\_item
  - KitListGui, 66
- m\_checked
  - Item, 27
- m\_clipboard\_items
  - KitListGui, 64
- m\_col\_checked
  - ModelItemColumns, 88
- m\_col\_num
  - ModelCategoryColumns, 85
  - ModelItemColumns, 88
- m\_col\_text
  - ModelCategoryColumns, 85
  - ModelItemColumns, 88
- m\_current\_cat\_id
  - KitListGui, 68
- m\_dao
  - KitList, 39

- Service, 99
- m\_deleted
  - GuiState, 24
- m\_desc
  - Item, 27
- m\_dirty
  - GuiState, 24
  - KitModel, 74
- m\_entry\_add\_category
  - KitListGui, 65
- m\_entry\_add\_item
  - KitListGui, 65
- m\_file\_save\_menu\_item
  - KitListGui, 65
- m\_file\_save\_tool\_button
  - KitListGui, 66
- m\_filename
  - KitListGui, 64
  - XmlDao, 110
- m\_id
  - Category, 19
  - Item, 27
- m\_ignore\_list\_events
  - KitListGui, 64
- m\_item
  - KitParser, 80
- m\_item\_cols
  - KitListGui, 67
- m\_item\_filter
  - KitModel, 75
- m\_item\_map
  - KitModel, 75
- m\_item\_tree\_view
  - KitListGui, 67
- m\_items
  - Category, 19
- m\_items\_node
  - XmlDao, 110
- m\_kit
  - KitListGui, 65
- m\_max\_category\_id
  - XmlDao, 111
- m\_max\_item\_id
  - XmlDao, 111
- m\_model
  - KitParser, 80
  - Service, 99
- m\_name
  - Category, 19
- m\_new
  - GuiState, 24
- m\_paste\_menu\_item
  - KitListGui, 66
- m\_recent\_files\_menu\_item
  - KitListGui, 66
- m\_ref\_category\_list\_store
  - KitListGui, 66
- m\_ref\_item\_tree\_model
  - KitListGui, 67
- m\_removed\_children
  - ModelCategory, 84
- m\_service
  - KitListGui, 67
- m\_state
  - KitListGui, 67
- m\_status\_bar
  - KitListGui, 67
- m\_verbose\_flag
  - KitListDao, 46
- m\_window
  - KitListGui, 65
- m\_window\_add\_category
  - KitListGui, 65
- m\_window\_add\_item
  - KitListGui, 65
- main
  - main.cpp, 132
- main.cpp, 132
  - html\_flag, 133
  - main, 132
  - verbose\_flag, 133
- ModelCategory, 81
  - ~ModelCategory, 82
  - add\_item, 83
  - get\_added\_children, 82
  - get\_items, 82
  - get\_model\_items, 82
  - get\_removed\_children, 82
  - KitModel, 84
  - m\_added\_children, 84
  - m\_removed\_children, 84
  - ModelCategory, 82
  - purge, 83
  - remove\_item, 83
  - remove\_items, 83
  - reset, 83
- ModelCategoryColumns, 85
  - m\_col\_num, 85
  - m\_col\_text, 85
  - ModelCategoryColumns, 85
- ModelCategoryContainer
  - kitmodel.hpp, 128
- ModelCategoryIter
  - kitmodel.hpp, 128
- ModelItem, 86
  - ModelItem, 86
  - ModelItemCompareId, 87
  - set\_checked, 86

- ModellItemColumns, 88
  - m\_col\_checked, 88
  - m\_col\_num, 88
  - m\_col\_text, 88
  - ModellItemColumns, 88
- ModellItemCompareId, 90
  - ModellItem, 87
  - ModellItemCompareId, 90
  - operator(), 90
- ModellItemContainer
  - kitmodel.hpp, 128
- ModellItemIter
  - kitmodel.hpp, 128
- new\_category
  - KitList, 37
  - KitListDao, 44
  - XmlDao, 107
- NYI
  - xmldao.hpp, 137
- on\_category\_change
  - KitListGui, 60
- on\_cell\_edit
  - KitListGui, 61
- on\_characters
  - KitParser, 79
- on\_clipboard\_clear
  - KitListGui, 60
- on\_clipboard\_get
  - KitListGui, 60
- on\_clipboard\_received
  - KitListGui, 60
- on\_comment
  - KitParser, 79
- on\_delete\_event
  - KitListGui, 55
- on\_end\_document
  - KitParser, 78
- on\_end\_element
  - KitParser, 78
- on\_error
  - KitParser, 79
- on\_fatal\_error
  - KitParser, 79
- on\_list\_item
  - KitList, 35
- on\_menu\_add
  - KitListGui, 57
- on\_menu\_check\_selected
  - KitListGui, 58
- on\_menu\_copy
  - KitListGui, 57
- on\_menu\_create\_category
  - KitListGui, 59
- on\_menu\_cut
  - KitListGui, 57
- on\_menu\_delete
  - KitListGui, 57
- on\_menu\_delete\_category
  - KitListGui, 59
- on\_menu\_file\_new
  - KitListGui, 55
- on\_menu\_file\_open
  - KitListGui, 56
- on\_menu\_help\_about
  - KitListGui, 59
- on\_menu\_help\_contents
  - KitListGui, 60
- on\_menu\_paste
  - KitListGui, 57
- on\_menu\_quit
  - KitListGui, 55
- on\_menu\_recent\_file
  - KitListGui, 56
- on\_menu\_rename\_category
  - KitListGui, 59
- on\_menu\_save
  - KitListGui, 56
- on\_menu\_save\_as
  - KitListGui, 56
- on\_menu\_select\_all
  - KitListGui, 58
- on\_menu\_show\_all
  - KitListGui, 58
- on\_menu\_show\_checked
  - KitListGui, 58
- on\_menu\_show\_unchecked
  - KitListGui, 58
- on\_menu\_uncheck\_selected
  - KitListGui, 59
- on\_row\_changed
  - KitListGui, 63
- on\_start\_document
  - KitParser, 78
- on\_start\_element
  - KitParser, 78
- on\_warning
  - KitParser, 79
- open\_as\_xml
  - Service, 96
- open\_file
  - KitListGui, 56
- operator<<
  - Item, 27
- operator()
  - CategoryCompareId, 20
  - CategoryCompareName, 21

- ItemCompareId, 29
- ItemCompareName, 30
- ItemFunctor, 31
- ModelItemCompareId, 90
- TickItem, 100
- paste\_from\_xml
  - KitListGui, 62
- paste\_status\_received
  - KitListGui, 61
- process\_category
  - KitParser, 78
- process\_category\_item
  - KitParser, 78
- process\_item
  - KitParser, 77
- purge
  - KitModel, 74
  - ModelCategory, 83
- raise
  - KitListGui, 63
- refresh\_category\_list
  - KitListGui, 62
- refresh\_item\_list
  - KitListGui, 62
- remove\_item
  - Category, 17
  - ModelCategory, 83
- remove\_item\_from\_category
  - KitList, 38
  - KitListDao, 44
  - XmlDao, 108
- remove\_items
  - ModelCategory, 83
- RENAME\_CATEGORY
  - kitlistgui.hpp, 123
- require\_filename
  - KitListDao, 46
  - Service, 98
  - XmlDao, 110
- reset
  - GuiState, 23
  - KitModel, 74
  - ModelCategory, 83
- run
  - KitListGui, 64
- safe\_open\_file
  - KitListGui, 64
- save
  - Service, 96
- save\_as\_xml
  - Service, 96
- save\_model
  - KitListDao, 42
  - XmlDao, 105
- SB\_ITEM\_COUNT
  - anonymous\_namespace{kitlistgui.cpp}, 13
- SB\_MSG
  - anonymous\_namespace{kitlistgui.cpp}, 14
- SB\_SAVE
  - anonymous\_namespace{kitlistgui.cpp}, 14
- select\_items
  - Service, 98
- selected\_row\_callback
  - KitListGui, 62
- Service, 91
  - ~Service, 93
  - copy\_items, 94
  - create\_category, 95
  - create\_default\_model, 96
  - create\_item, 94
  - delete\_category, 94
  - delete\_item, 94
  - filter, 95
  - find\_category, 93
  - find\_item, 93
  - get\_categories, 97
  - get\_items, 97
  - get\_next\_category\_id, 93
  - get\_next\_item\_id, 93
  - is\_model\_dirty, 95
  - load\_model, 93
  - m\_dao, 99
  - m\_model, 99
  - open\_as\_xml, 96
  - require\_filename, 98
  - save, 96
  - save\_as\_xml, 96
  - select\_items, 98
  - Service, 93
  - set\_model\_dirty, 95
  - show\_all, 97
  - show\_checked\_only, 98
  - show\_unchecked\_only, 98
  - toggle\_selected\_items, 98
  - update\_item, 96
- service.cpp, 134
- service.hpp, 135
  - SERVICE\_H, 135
- SERVICE\_H
  - service.hpp, 135
- set\_all\_flags
  - KitList, 39
  - KitListDao, 46
  - XmlDao, 110
- set\_category\_flag

- KitList, 38
- KitListDao, 45
- XmlDao, 109
- set\_checked
  - Item, 26
  - ModelItem, 86
- set\_deleted
  - GuiState, 23
- set\_description
  - Item, 26
- set\_dirty
  - GuiState, 23
  - KitModel, 73
- set\_filename
  - XmlDao, 110
- set\_id
  - Category, 16
  - Item, 26
- set\_item\_flag
  - KitList, 38
  - KitListDao, 45
  - XmlDao, 109
- set\_model\_dirty
  - Service, 95
- set\_name
  - Category, 17
- set\_new\_flag
  - GuiState, 23
- set\_selected
  - KitListGui, 63
- set\_verbose
  - KitListDao, 42
- show\_all
  - KitModel, 73
  - Service, 97
- show\_checked\_only
  - KitModel, 74
  - Service, 98
- show\_unchecked\_only
  - KitModel, 74
  - Service, 98
- SlotForeachCategory
  - kitmodel.hpp, 128
- SlotForeachCategoryIter
  - kitmodel.hpp, 128
- SlotForeachItem
  - item.hpp, 116
- SlotForeachModelItem
  - kitmodel.hpp, 128
- SlotForeachModelItemIter
  - kitmodel.hpp, 129
- tick\_items
  - KitList, 36, 37
- TickItem, 100
  - m\_changed, 100
  - operator(), 100
  - TickItem, 100
- toggle\_selected
  - KitListGui, 63
- toggle\_selected\_items
  - Service, 98
- type\_children
  - kitlistgui.cpp, 120
- UNCHECKED
  - kitmodel.hpp, 129
- UNCHECKED\_ITEMS
  - kitlistdao.hpp, 117
- unset\_all\_flags
  - KitList, 39
  - KitListDao, 46
  - XmlDao, 110
- unset\_category\_flag
  - KitList, 38
  - KitListDao, 45
  - XmlDao, 109
- unset\_item\_flag
  - KitList, 38
  - KitListDao, 45
  - XmlDao, 109
- update\_item
  - Service, 96
- update\_item\_checked\_state
  - KitListDao, 44
  - XmlDao, 108
- update\_item\_count
  - KitListGui, 63
- update\_paste\_status
  - KitListGui, 61
- update\_recent\_files
  - KitListGui, 55
- update\_recent\_files\_menu
  - KitListGui, 54
- verbose\_flag
  - main.cpp, 133
- XML\_ELEMENT\_ID
  - anonymous\_namespace{kitlistgui.cpp}, 14
- XmlDao, 102
  - add\_category\_item\_to\_dom, 104
  - add\_category\_to\_dom, 104
  - add\_item, 106
  - add\_item\_to\_dom, 104
  - append\_items\_to\_category, 107
  - associate\_item\_with\_category, 107
  - delete\_category, 109

- delete\_item, 108
- get\_all\_items, 106
- get\_categories, 107
- get\_category, 106
- get\_model, 104, 105
- get\_next\_category\_id, 108
- get\_next\_item\_id, 108
- m\_cat\_items\_node, 111
- m\_categories\_node, 111
- m\_filename, 110
- m\_items\_node, 110
- m\_max\_category\_id, 111
- m\_max\_item\_id, 111
- new\_category, 107
- remove\_item\_from\_category, 108
- require\_filename, 110
- save\_model, 105
- set\_all\_flags, 110
- set\_category\_flag, 109
- set\_filename, 110
- set\_item\_flag, 109
- unset\_all\_flags, 110
- unset\_category\_flag, 109
- unset\_item\_flag, 109
- update\_item\_checked\_state, 108
- XmlDao, 104
- xmldao.cpp, 136
- xmldao.hpp, 137
  - NYI, 137
  - XMLDAO\_H, 137
- XMLDAO\_H
  - xmldao.hpp, 137