

PreCheck

0.3

Généré par Doxygen 1.8.4

Jeudi Juin 20 2013 23 :51 :14

Table des matières

1	Index hiérarchique	1
1.1	Hiérarchie des classes	1
2	Index des classes	3
2.1	Liste des classes	3
3	Index des fichiers	5
3.1	Liste des fichiers	5
4	Documentation des classes	9
4.1	Référence de la classe ApplicationWindow	9
4.2	Référence de la classe Button	10
4.3	Référence de la classe FocusScope	11
4.4	Référence de la classe GridLayout	12
4.5	Référence de la classe Item	13
4.6	Référence de la classe QAbstractListModel	14
4.7	Référence de la classe QObject	15
4.8	Référence de la classe QQuickItem	17
4.9	Référence de la classe QSortFilterProxyModel	18
4.10	Référence de la classe QState	19
4.11	Référence de la classe QStateMachine	20
4.12	Référence de la classe Rectangle	21
4.13	Référence de la classe SH_AdaptDatabaseState	21
4.13.1	Description détaillée	24
4.13.2	Documentation des constructeurs et destructeur	24
4.13.2.1	SH_AdaptDatabaseState	24
4.13.3	Documentation des fonctions membres	24
4.13.3.1	insertUpdate	24
4.13.3.2	next	25
4.13.3.3	onEntry	25
4.13.3.4	onExit	26
4.13.3.5	toString	26

4.14 Référence de la classe SH_AddressCreationStateMachine	27
4.14.1 Description détaillée	30
4.14.2 Documentation des constructeurs et destructeur	30
4.14.2.1 SH_AddressCreationStateMachine	30
4.14.3 Documentation des fonctions membres	30
4.14.3.1 addChildrenNextTransition	30
4.14.3.2 addIOState	32
4.14.3.3 addIOSStateMachine	33
4.14.3.4 cancelReplacement	34
4.14.3.5 clearAll	34
4.14.3.6 confirmInput	35
4.14.3.7 displayCalendar	35
4.14.3.8 displayFileDialog	36
4.14.3.9 getContentValue	36
4.14.3.10 historyValue	37
4.14.3.11 ioContent	37
4.14.3.12 ioStatesHistory	38
4.14.3.13 next	38
4.14.3.14 receiveInput	39
4.14.3.15 replaceInput	39
4.14.3.16 resendText	39
4.14.3.17 sendText	40
4.14.3.18 setContentValue	40
4.14.3.19 setIOcontent	41
4.14.3.20 setIOSStateHistory	41
4.14.3.21 setTableName	42
4.14.3.22 tableName	42
4.14.3.23 toString	43
4.14.3.24 validateInput	44
4.14.4 Documentation des données membres	44
4.14.4.1 m_ioContent	44
4.14.4.2 m_ioStatesHistory	44
4.14.4.3 m_tableName	44
4.15 Référence de la classe SH_app	45
4.15.1 Description détaillée	46
4.15.2 Documentation des fonctions membres	46
4.15.2.1 reload	46
4.15.3 Documentation des propriétés	46
4.15.3.1 pages	46
4.16 Référence de la classe SH_ApplicationCore	46

4.16.1	Description détaillée	49
4.16.2	Documentation des énumérations membres	49
4.16.2.1	AppMode	49
4.16.3	Documentation des constructeurs et destructeur	50
4.16.3.1	SH_ApplicationCore	50
4.16.4	Documentation des fonctions membres	50
4.16.4.1	balanceLogRoutine	50
4.16.4.2	cancelReplacement	50
4.16.4.3	cancelRunningThread	50
4.16.4.4	clearAll	51
4.16.4.5	connectRunningThread	51
4.16.4.6	currentFSMchanged	52
4.16.4.7	displayCalendar	52
4.16.4.8	init	52
4.16.4.9	launchBillingsThread	53
4.16.4.10	launchBillThread	53
4.16.4.11	launchBookingsThread	54
4.16.4.12	mode	54
4.16.4.13	modeChanged	55
4.16.4.14	openTab	55
4.16.4.15	receiveConfirmation	55
4.16.4.16	receiveInput	55
4.16.4.17	receiveValidation	55
4.16.4.18	replacedInput	55
4.16.4.19	resendText	56
4.16.4.20	sendText	56
4.16.4.21	setMode	56
4.16.4.22	setUser	57
4.16.4.23	user	57
4.16.4.24	userChanged	58
4.16.4.25	userExists	58
4.16.4.26	userLogOut	58
4.16.5	Documentation des données membres	59
4.16.5.1	m_currentFSM	59
4.16.5.2	m_currentUser	59
4.16.5.3	m_mode	59
4.16.6	Documentation des propriétés	59
4.16.6.1	currentMode	59
4.16.6.2	currentUser	59
4.17	Référence de la classe SH_BillingCreationStateMachine	60

4.17.1	Description détaillée	62
4.17.2	Documentation des constructeurs et destructeur	62
4.17.2.1	SH_BillingCreationStateMachine	62
4.17.3	Documentation des fonctions membres	65
4.17.3.1	addChildrenNextTransition	65
4.17.3.2	addIOState	66
4.17.3.3	addIOStateMachine	68
4.17.3.4	cancelReplacement	69
4.17.3.5	clearAll	69
4.17.3.6	confirmInput	70
4.17.3.7	displayCalendar	70
4.17.3.8	displayFileDialog	71
4.17.3.9	getContentValue	71
4.17.3.10	historyValue	72
4.17.3.11	ioContent	72
4.17.3.12	ioStatesHistory	73
4.17.3.13	next	73
4.17.3.14	receiveInput	74
4.17.3.15	replaceInput	74
4.17.3.16	resendText	74
4.17.3.17	sendText	75
4.17.3.18	setContentValue	75
4.17.3.19	setIOcontent	76
4.17.3.20	setIOStateHistory	76
4.17.3.21	setTableName	77
4.17.3.22	tableName	77
4.17.3.23	toString	78
4.17.3.24	validateInput	79
4.17.4	Documentation des données membres	79
4.17.4.1	m_ioContent	79
4.17.4.2	m_ioStatesHistory	79
4.17.4.3	m_tableName	79
4.18	Référence de la classe SH_BillingsDelegate	80
4.18.1	Description détaillée	81
4.18.2	Documentation des propriétés	81
4.18.2.1	fontSize	81
4.18.2.2	value	81
4.19	Référence de la classe SH_BillingsTableModel	82
4.19.1	Description détaillée	84
4.19.2	Documentation des constructeurs et destructeur	84

4.19.2.1	SH_BillingsTableModel	84
4.19.3	Documentation des fonctions membres	85
4.19.3.1	addFilterKeyColumn	85
4.19.3.2	containsFilterKeyColumn	85
4.19.3.3	currentSortKeyColumn	85
4.19.3.4	data	86
4.19.3.5	data	86
4.19.3.6	fetch	87
4.19.3.7	field	87
4.19.3.8	fields	88
4.19.3.9	fieldsCount	88
4.19.3.10	fillModel	89
4.19.3.11	filterAcceptsRow	89
4.19.3.12	flags	90
4.19.3.13	invalidateFilter	90
4.19.3.14	isEmpty	90
4.19.3.15	lastError	90
4.19.3.16	removeFilterKeyColumn	91
4.19.3.17	roleNames	91
4.19.3.18	setBooleanColumns	91
4.19.3.19	setData	91
4.19.3.20	setNotNullColumns	92
4.19.3.21	setNullColumns	92
4.19.3.22	setPasswordColumns	93
4.19.3.23	setReadOnlyColumns	93
4.19.3.24	setSortKeyColumn	94
4.19.3.25	sort	94
4.19.3.26	sortChanged	95
4.19.3.27	tableName	95
4.19.4	Documentation des données membres	96
4.19.4.1	model	96
4.19.5	Documentation des propriétés	96
4.19.5.1	empty	96
4.19.5.2	fieldsList	96
4.19.5.3	lastError	96
4.19.5.4	sortKeyColumn	96
4.19.5.5	table	96
4.20	Référence de la classe SH_BillsTableModel	97
4.20.1	Description détaillée	99
4.20.2	Documentation des constructeurs et destructeur	99

4.20.2.1	SH_BillsTableModel	99
4.20.3	Documentation des fonctions membres	100
4.20.3.1	addFilterKeyColumn	100
4.20.3.2	containsFilterKeyColumn	100
4.20.3.3	currentSortKeyColumn	100
4.20.3.4	data	101
4.20.3.5	data	101
4.20.3.6	fetch	102
4.20.3.7	field	102
4.20.3.8	fields	103
4.20.3.9	fieldsCount	103
4.20.3.10	fillModel	104
4.20.3.11	filterAcceptsRow	104
4.20.3.12	flags	104
4.20.3.13	invalidateFilter	105
4.20.3.14	isEmpty	105
4.20.3.15	lastError	105
4.20.3.16	removeFilterKeyColumn	105
4.20.3.17	roleNames	105
4.20.3.18	setBooleanColumns	106
4.20.3.19	setData	106
4.20.3.20	setNotNullColumns	107
4.20.3.21	setNullColumns	107
4.20.3.22	setPasswordColumns	108
4.20.3.23	setReadOnlyColumns	108
4.20.3.24	setSortKeyColumn	108
4.20.3.25	sort	109
4.20.3.26	sortChanged	110
4.20.3.27	tableName	110
4.20.4	Documentation des données membres	110
4.20.4.1	model	110
4.20.5	Documentation des propriétés	111
4.20.5.1	empty	111
4.20.5.2	fieldsList	111
4.20.5.3	lastError	111
4.20.5.4	sortKeyColumn	111
4.20.5.5	table	111
4.21	Référence de la classe SH_BookingsDelegate	112
4.21.1	Description détaillée	113
4.21.2	Documentation des propriétés	113

4.21.2.1	fontSize	113
4.21.2.2	value	113
4.22	Référence de la classe SH_BookingsTableModel	114
4.22.1	Description détaillée	116
4.22.2	Documentation des constructeurs et destructeur	116
4.22.2.1	SH_BookingsTableModel	116
4.22.3	Documentation des fonctions membres	117
4.22.3.1	addFilterKeyColumn	117
4.22.3.2	containsFilterKeyColumn	117
4.22.3.3	currentSortKeyColumn	117
4.22.3.4	data	118
4.22.3.5	data	118
4.22.3.6	fetch	119
4.22.3.7	field	119
4.22.3.8	fields	120
4.22.3.9	fieldsCount	120
4.22.3.10	fillModel	121
4.22.3.11	filterAcceptsRow	121
4.22.3.12	flags	122
4.22.3.13	invalidateFilter	122
4.22.3.14	isEmpty	122
4.22.3.15	lastError	122
4.22.3.16	removeFilterKeyColumn	123
4.22.3.17	roleNames	123
4.22.3.18	setBooleanColumns	123
4.22.3.19	setData	123
4.22.3.20	setNotNullColumns	124
4.22.3.21	setNullColumns	124
4.22.3.22	setPasswordColumns	125
4.22.3.23	setReadOnlyColumns	125
4.22.3.24	setSortKeyColumn	126
4.22.3.25	sort	126
4.22.3.26	sortChanged	127
4.22.3.27	tableName	127
4.22.4	Documentation des données membres	128
4.22.4.1	model	128
4.22.5	Documentation des propriétés	128
4.22.5.1	empty	128
4.22.5.2	fieldsList	128
4.22.5.3	lastError	128

4.22.5.4 sortKeyColumn	128
4.22.5.5 table	128
4.23 Référence de la classe SH_CalendarDialog	129
4.23.1 Description détaillée	130
4.23.2 Documentation des fonctions membres	131
4.23.2.1 clicked	131
4.23.2.2 closed	131
4.23.2.3 opened	131
4.23.2.4 refresh	131
4.23.2.5 selected	131
4.23.3 Documentation des propriétés	131
4.23.3.1 currentDate	131
4.23.3.2 currentDay	131
4.23.3.3 currentMonth	131
4.23.3.4 currentMonthLength	131
4.23.3.5 currentWeekday	131
4.23.3.6 currentYear	131
4.23.3.7 firstWeekdayIndex	131
4.23.3.8 monthsList	131
4.23.3.9 text	131
4.23.3.10 weekdaysList	132
4.24 Référence de la classe SH_ClientCreationStateMachine	132
4.24.1 Description détaillée	135
4.24.2 Documentation des constructeurs et destructeur	135
4.24.2.1 SH_ClientCreationStateMachine	135
4.24.3 Documentation des fonctions membres	136
4.24.3.1 addChildrenNextTransition	136
4.24.3.2 addIOState	137
4.24.3.3 addIOStateMachine	139
4.24.3.4 cancelReplacement	140
4.24.3.5 clearAll	140
4.24.3.6 confirmInput	141
4.24.3.7 displayCalendar	141
4.24.3.8 displayFileDialog	142
4.24.3.9 getContentValue	142
4.24.3.10 historyValue	143
4.24.3.11 ioContent	143
4.24.3.12 ioStatesHistory	144
4.24.3.13 next	144
4.24.3.14 receiveInput	145

4.24.3.15 replaceInput	145
4.24.3.16 resendText	145
4.24.3.17 sendText	146
4.24.3.18 setContentValue	146
4.24.3.19 setIOcontent	147
4.24.3.20 setIOSStateHistory	147
4.24.3.21 setTableName	148
4.24.3.22 tableName	148
4.24.3.23 toString	149
4.24.3.24 validateInput	150
4.24.4 Documentation des données membres	150
4.24.4.1 m_ioContent	150
4.24.4.2 m_ioStatesHistory	150
4.24.4.3 m_tableName	150
4.25 Référence de la classe SH_ClientsTableModel	151
4.25.1 Description détaillée	153
4.25.2 Documentation des constructeurs et destructeur	153
4.25.2.1 SH_ClientsTableModel	153
4.25.3 Documentation des fonctions membres	154
4.25.3.1 addFilterKeyColumn	154
4.25.3.2 containsFilterKeyColumn	154
4.25.3.3 currentSortKeyColumn	154
4.25.3.4 data	155
4.25.3.5 data	155
4.25.3.6 fetch	156
4.25.3.7 field	156
4.25.3.8 fields	157
4.25.3.9 fieldsCount	157
4.25.3.10 fillModel	158
4.25.3.11 filterAcceptsRow	158
4.25.3.12 flags	158
4.25.3.13 invalidateFilter	159
4.25.3.14 isEmpty	159
4.25.3.15 lastError	159
4.25.3.16 removeFilterKeyColumn	159
4.25.3.17 roleNames	159
4.25.3.18 setBooleanColumns	160
4.25.3.19 setData	160
4.25.3.20 setNotNullColumns	161
4.25.3.21 setNullColumns	161

4.25.3.22 setPasswordColumns	162
4.25.3.23 setReadOnlyColumns	162
4.25.3.24 setSortKeyColumn	162
4.25.3.25 sort	163
4.25.3.26 sortChanged	164
4.25.3.27 tableName	164
4.25.4 Documentation des données membres	164
4.25.4.1 model	164
4.25.5 Documentation des propriétés	165
4.25.5.1 empty	165
4.25.5.2 fieldsList	165
4.25.5.3 lastError	165
4.25.5.4 sortKeyColumn	165
4.25.5.5 table	165
4.26 Référence de la classe SH_CommonPage	166
4.26.1 Description détaillée	167
4.26.2 Documentation des fonctions membres	167
4.26.2.1 cancelProcess	167
4.26.2.2 cancelReplace	167
4.26.2.3 confirm	167
4.26.2.4 keySelected	167
4.26.2.5 quit	167
4.26.2.6 reload	167
4.26.2.7 replace	167
4.26.2.8 selected	167
4.26.2.9 validate	167
4.26.3 Documentation des propriétés	167
4.26.3.1 streamBuffer	167
4.27 Référence de la classe SH_Company	167
4.27.1 Description détaillée	168
4.28 Référence de la classe SH_ConfirmationState	169
4.28.1 Description détaillée	171
4.28.2 Documentation des constructeurs et destructeur	171
4.28.2.1 SH_ConfirmationState	171
4.28.3 Documentation des fonctions membres	171
4.28.3.1 confirmInput	171
4.28.3.2 display	172
4.28.3.3 input	172
4.28.3.4 next	173
4.28.3.5 onEntry	173

4.28.3.6 <code>onExit</code>	174
4.28.3.7 <code>output</code>	174
4.28.3.8 <code>rawInput</code>	174
4.28.3.9 <code>resendInput</code>	175
4.28.3.10 <code>sendOutput</code>	175
4.28.3.11 <code>setInput</code>	176
4.28.3.12 <code>setOutput</code>	176
4.28.3.13 <code>setVisibility</code>	177
4.28.3.14 <code>toString</code>	177
4.28.3.15 <code>visibility</code>	178
4.29 Référence de la classe <code>SH_ConexionPage</code>	179
4.29.1 Description détaillée	180
4.29.2 Documentation des fonctions membres	180
4.29.2.1 <code>checkUsername</code>	180
4.29.2.2 <code>loggedin</code>	180
4.29.2.3 <code>logIn</code>	180
4.29.2.4 <code>reload</code>	180
4.30 Référence de la classe <code>SH_ContentView</code>	180
4.30.1 Description détaillée	181
4.30.2 Documentation des fonctions membres	182
4.30.2.1 <code>addFilterIndex</code>	182
4.30.2.2 <code>filter</code>	182
4.30.2.3 <code>removeFilterIndex</code>	182
4.30.2.4 <code>selected</code>	182
4.30.2.5 <code>sort</code>	182
4.30.3 Documentation des propriétés	182
4.30.3.1 <code>activeFilterIndicatorIndexes</code>	182
4.30.3.2 <code>emptyDelegate</code>	182
4.30.3.3 <code>model</code>	182
4.30.3.4 <code>sectionDelegate</code>	182
4.30.3.5 <code>sectionIndex</code>	182
4.30.3.6 <code>SH_DataDelegate</code>	182
4.31 Référence de la classe <code>SH_DatabaseContentQuestionState</code>	182
4.31.1 Description détaillée	185
4.31.2 Documentation des constructeurs et destructeur	185
4.31.2.1 <code>SH_DatabaseContentQuestionState</code>	185
4.31.3 Documentation des fonctions membres	186
4.31.3.1 <code>answerInvalid</code>	186
4.31.3.2 <code>answerValid</code>	186
4.31.3.3 <code>checkValidity</code>	187

4.31.3.4	choiceList	187
4.31.3.5	display	188
4.31.3.6	displayChoiceList	188
4.31.3.7	givenAnswer	188
4.31.3.8	input	189
4.31.3.9	isValid	189
4.31.3.10	next	189
4.31.3.11	onEntry	190
4.31.3.12	onExit	190
4.31.3.13	output	191
4.31.3.14	rawInput	191
4.31.3.15	resendInput	192
4.31.3.16	sendOutput	192
4.31.3.17	setGivenAnswer	193
4.31.3.18	setInput	193
4.31.3.19	setOutput	194
4.31.3.20	setVisibility	194
4.31.3.21	toString	195
4.31.3.22	visibility	196
4.31.4	Documentation des données membres	196
4.31.4.1	m_choices	196
4.31.4.2	m_choicesDisplayed	196
4.31.4.3	m_condition	196
4.31.4.4	m_field	196
4.31.4.5	m_table	197
4.32	Référence de la classe SH_DatabaseManager	197
4.32.1	Description détaillée	199
4.32.2	Documentation des constructeurs et destructeur	199
4.32.2.1	SH_DatabaseManager	199
4.32.2.2	~SH_DatabaseManager	200
4.32.3	Documentation des fonctions membres	200
4.32.3.1	dataCount	201
4.32.3.2	dbConnect	201
4.32.3.3	dbDisconnect	202
4.32.3.4	divideQVariantMap	203
4.32.3.5	execInsertReturningQuery	203
4.32.3.6	execReplaceQuery	204
4.32.3.7	execSelectQuery	205
4.32.3.8	getDbConnection	205
4.32.3.9	getInstance	205

4.32.3.10 isConnected	206
4.32.3.11 tableExists	206
4.32.4 Documentation des données membres	207
4.32.4.1 _instance	207
4.32.4.2 dbConnection	207
4.33 Référence de la classe SH_DataDelegate	207
4.33.1 Description détaillée	208
4.33.2 Documentation des propriétés	208
4.33.2.1 fontSize	208
4.33.2.2 value	208
4.34 Référence de la classe SH_DateQuestionState	208
4.34.1 Description détaillée	211
4.34.2 Documentation des constructeurs et destructeur	211
4.34.2.1 SH_DateQuestionState	211
4.34.3 Documentation des fonctions membres	212
4.34.3.1 answerInvalid	212
4.34.3.2 answerValid	212
4.34.3.3 checkValidity	212
4.34.3.4 display	213
4.34.3.5 getFuture	213
4.34.3.6 getPast	214
4.34.3.7 givenAnswer	214
4.34.3.8 input	214
4.34.3.9 isAnswerValid	215
4.34.3.10 next	215
4.34.3.11 onEntry	215
4.34.3.12 onExit	216
4.34.3.13 output	216
4.34.3.14 rawInput	217
4.34.3.15 resendInput	217
4.34.3.16 sendOutput	217
4.34.3.17 setFuture	218
4.34.3.18 setGivenAnswer	218
4.34.3.19 setInput	219
4.34.3.20 setOutput	219
4.34.3.21 setPast	220
4.34.3.22 setVisibility	220
4.34.3.23 toString	221
4.34.3.24 visibility	221
4.34.4 Documentation des données membres	222

4.34.4.1	m_future	222
4.34.4.2	m_past	222
4.35	Référence de la classe SH_DecimalQuestionState	222
4.35.1	Description détaillée	225
4.35.2	Documentation des constructeurs et destructeur	225
4.35.2.1	SH_DecimalQuestionState	225
4.35.3	Documentation des fonctions membres	226
4.35.3.1	answerInvalid	226
4.35.3.2	answerValid	226
4.35.3.3	checkValidity	226
4.35.3.4	display	227
4.35.3.5	givenAnswer	227
4.35.3.6	input	228
4.35.3.7	isValid	228
4.35.3.8	max	228
4.35.3.9	min	229
4.35.3.10	next	229
4.35.3.11	onEntry	230
4.35.3.12	onExit	230
4.35.3.13	output	231
4.35.3.14	rawInput	231
4.35.3.15	resendInput	232
4.35.3.16	sendOutput	232
4.35.3.17	setGivenAnswer	233
4.35.3.18	setInput	233
4.35.3.19	setMax	234
4.35.3.20	setMin	234
4.35.3.21	setOutput	235
4.35.3.22	setVisibility	235
4.35.3.23	toString	236
4.35.3.24	visibility	237
4.35.4	Documentation des données membres	237
4.35.4.1	m_max	237
4.35.4.2	m_min	237
4.36	Référence de la classe SH_ExtendedProxyModel	237
4.36.1	Description détaillée	240
4.36.2	Documentation des constructeurs et destructeur	241
4.36.2.1	SH_ExtendedProxyModel	241
4.36.3	Documentation des fonctions membres	241
4.36.3.1	addFilterKeyColumn	241

4.36.3.2	containsFilterKeyColumn	241
4.36.3.3	currentSortKeyColumn	241
4.36.3.4	data	242
4.36.3.5	data	242
4.36.3.6	fetch	243
4.36.3.7	field	243
4.36.3.8	fields	244
4.36.3.9	fieldsCount	244
4.36.3.10	fillModel	244
4.36.3.11	filterAcceptsRow	245
4.36.3.12	flags	245
4.36.3.13	invalidateFilter	246
4.36.3.14	isEmpty	246
4.36.3.15	lastError	246
4.36.3.16	removeFilterKeyColumn	246
4.36.3.17	replaceSet	247
4.36.3.18	roleNames	247
4.36.3.19	setBooleanColumns	248
4.36.3.20	setData	248
4.36.3.21	setNotNullColumns	249
4.36.3.22	setNullColumns	249
4.36.3.23	setPasswordColumns	250
4.36.3.24	setReadOnlyColumns	250
4.36.3.25	setSortKeyColumn	250
4.36.3.26	sort	251
4.36.3.27	sortChanged	252
4.36.3.28	tableName	252
4.36.4	Documentation des données membres	252
4.36.4.1	booleanSet	252
4.36.4.2	filters	253
4.36.4.3	model	253
4.36.4.4	notNullSet	253
4.36.4.5	nullSet	253
4.36.4.6	passwordSet	253
4.36.4.7	readonlySet	253
4.36.4.8	sortIndex	253
4.36.5	Documentation des propriétés	254
4.36.5.1	empty	254
4.36.5.2	fieldsList	254
4.36.5.3	lastError	254

4.36.5.4 sortKeyColumn	254
4.36.5.5 table	254
4.37 Référence de la classe SH_ExtendedQQmlAction	254
4.37.1 Description détaillée	257
4.37.2 Documentation des constructeurs et destructeur	257
4.37.2.1 SH_ExtendedQQmlAction	257
4.37.2.2 ~SH_ExtendedQQmlAction	258
4.37.3 Documentation des fonctions membres	258
4.37.3.1 enabledChanged	258
4.37.3.2 event	258
4.37.3.3 icon	259
4.37.3.4 iconChanged	259
4.37.3.5 iconName	260
4.37.3.6 iconNameChanged	260
4.37.3.7 iconSource	260
4.37.3.8 iconSourceChanged	261
4.37.3.9 iconVariant	261
4.37.3.10 isEnabled	261
4.37.3.11 keyShortcut	261
4.37.3.12 keyShortcutChanged	262
4.37.3.13 setEnabled	262
4.37.3.14 setIcon	262
4.37.3.15 setIconName	262
4.37.3.16 setIconSource	263
4.37.3.17 setKeySequence	263
4.37.3.18 setKeyShortcut	264
4.37.3.19 setMnemonicFromText	264
4.37.3.20 setShortcut	265
4.37.3.21 setText	265
4.37.3.22 setTooltip	266
4.37.3.23 shortcut	266
4.37.3.24 shortcutChanged	266
4.37.3.25 text	267
4.37.3.26 textChanged	267
4.37.3.27 toggled	267
4.37.3.28 tooltip	267
4.37.3.29 tooltipChanged	267
4.37.3.30 trigger	268
4.37.3.31 triggered	268
4.37.4 Documentation des données membres	268

4.37.4.1	<code>m_enabled</code>	268
4.37.4.2	<code>m_icon</code>	269
4.37.4.3	<code>m_iconName</code>	269
4.37.4.4	<code>m_iconSource</code>	269
4.37.4.5	<code>m_mnemonic</code>	269
4.37.4.6	<code>m_shortcut</code>	269
4.37.4.7	<code>m_text</code>	269
4.37.4.8	<code>m_tooltip</code>	269
4.37.5	Documentation des propriétés	270
4.37.5.1	<code>__icon</code>	270
4.37.5.2	<code>enabled</code>	270
4.37.5.3	<code>iconName</code>	270
4.37.5.4	<code>iconSource</code>	270
4.37.5.5	<code>keyShortcut</code>	270
4.37.5.6	<code>shortcut</code>	270
4.37.5.7	<code>text</code>	270
4.37.5.8	<code>tooltip</code>	270
4.38	Référence de la classe <code>SH_FileSelectionState</code>	270
4.38.1	Description détaillée	273
4.38.2	Documentation des constructeurs et destructeur	273
4.38.2.1	<code>SH_FileSelectionState</code>	273
4.38.3	Documentation des fonctions membres	273
4.38.3.1	<code>display</code>	273
4.38.3.2	<code>input</code>	274
4.38.3.3	<code>next</code>	274
4.38.3.4	<code>onEntry</code>	275
4.38.3.5	<code>onExit</code>	275
4.38.3.6	<code>output</code>	276
4.38.3.7	<code>rawInput</code>	276
4.38.3.8	<code>resendInput</code>	277
4.38.3.9	<code>sendOutput</code>	277
4.38.3.10	<code>setInput</code>	277
4.38.3.11	<code>setOutput</code>	278
4.38.3.12	<code>setVisibility</code>	279
4.38.3.13	<code>toString</code>	279
4.38.3.14	<code>visibility</code>	280
4.39	Référence de la classe <code>SH_GenericState</code>	281
4.39.1	Description détaillée	283
4.39.2	Documentation des constructeurs et destructeur	283
4.39.2.1	<code>SH_GenericState</code>	283

4.39.3 Documentation des fonctions membres	283
4.39.3.1 name	283
4.39.3.2 next	284
4.39.3.3 onEntry	284
4.39.3.4 onExit	285
4.39.3.5 onMachineStarted	286
4.39.3.6 onTransitionTriggered	286
4.39.3.7 ptraddress	287
4.39.3.8 setName	287
4.39.3.9 toString	287
4.40 Référence de la classe SH_GroupsTableModel	288
4.40.1 Description détaillée	291
4.40.2 Documentation des constructeurs et destructeur	291
4.40.2.1 SH_GroupsTableModel	291
4.40.3 Documentation des fonctions membres	292
4.40.3.1 addFilterKeyColumn	292
4.40.3.2 containsFilterKeyColumn	292
4.40.3.3 currentSortKeyColumn	292
4.40.3.4 data	293
4.40.3.5 data	293
4.40.3.6 fetch	294
4.40.3.7 field	294
4.40.3.8 fields	295
4.40.3.9 fieldsCount	295
4.40.3.10 fillModel	296
4.40.3.11 filterAcceptsRow	296
4.40.3.12 flags	296
4.40.3.13 invalidateFilter	297
4.40.3.14 isEmpty	297
4.40.3.15 lastError	297
4.40.3.16 removeFilterKeyColumn	297
4.40.3.17 roleNames	297
4.40.3.18 setBooleanColumns	298
4.40.3.19 setData	298
4.40.3.20 setNotNullColumns	299
4.40.3.21 setNullColumns	299
4.40.3.22 setPasswordColumns	300
4.40.3.23 setReadOnlyColumns	300
4.40.3.24 setSortKeyColumn	300
4.40.3.25 sort	301

4.40.3.26 sortChanged	302
4.40.3.27 tableName	302
4.40.4 Documentation des données membres	302
4.40.4.1 model	302
4.40.5 Documentation des propriétés	303
4.40.5.1 empty	303
4.40.5.2 fieldsList	303
4.40.5.3 lastError	303
4.40.5.4 sortKeyColumn	303
4.40.5.5 table	303
4.41 Référence de la classe SH_HeaderView	304
4.41.1 Description détaillée	305
4.41.2 Documentation des fonctions membres	305
4.41.2.1 checked	305
4.41.2.2 up	305
4.41.3 Documentation des propriétés	305
4.41.3.1 delegateModel	305
4.41.3.2 model	305
4.42 Référence de la classe SH_InOutState	305
4.42.1 Description détaillée	308
4.42.2 Documentation des constructeurs et destructeur	308
4.42.2.1 SH_InOutState	308
4.42.3 Documentation des fonctions membres	308
4.42.3.1 display	308
4.42.3.2 input	309
4.42.3.3 next	309
4.42.3.4 onEntry	310
4.42.3.5 onExit	310
4.42.3.6 output	311
4.42.3.7 rawInput	311
4.42.3.8 resendInput	312
4.42.3.9 sendOutput	312
4.42.3.10 setInput	312
4.42.3.11 setOutput	313
4.42.3.12 setVisibility	314
4.42.3.13 toString	314
4.42.3.14 visibility	315
4.42.4 Documentation des données membres	315
4.42.4.1 m_display	315
4.42.4.2 m_input	316

4.42.4.3	<code>m_isVisible</code>	316
4.42.4.4	<code>m_output</code>	316
4.43	Référence de la classe <code>SH_InOutStateMachine</code>	316
4.43.1	Description détaillée	318
4.43.2	Documentation des constructeurs et destructeur	318
4.43.2.1	<code>SH_InOutStateMachine</code>	319
4.43.3	Documentation des fonctions membres	319
4.43.3.1	<code>addChildsNextTransition</code>	319
4.43.3.2	<code>addIOState</code>	320
4.43.3.3	<code>addIOStateMachine</code>	322
4.43.3.4	<code>cancelReplacement</code>	323
4.43.3.5	<code>clearAll</code>	323
4.43.3.6	<code>confirmInput</code>	323
4.43.3.7	<code>displayCalendar</code>	324
4.43.3.8	<code>displayFileDialog</code>	324
4.43.3.9	<code>getContentValue</code>	325
4.43.3.10	<code>historyValue</code>	325
4.43.3.11	<code>ioContent</code>	326
4.43.3.12	<code>ioStatesHistory</code>	326
4.43.3.13	<code>name</code>	327
4.43.3.14	<code>next</code>	327
4.43.3.15	<code>ptraddress</code>	328
4.43.3.16	<code>receiveInput</code>	328
4.43.3.17	<code>replaceInput</code>	328
4.43.3.18	<code>resendText</code>	329
4.43.3.19	<code>sendText</code>	329
4.43.3.20	<code>setContentValue</code>	330
4.43.3.21	<code>setIOcontent</code>	330
4.43.3.22	<code>setIOStateHistory</code>	331
4.43.3.23	<code>setIOStatesHistory</code>	331
4.43.3.24	<code>setName</code>	332
4.43.3.25	<code>setTableName</code>	332
4.43.3.26	<code>tableName</code>	333
4.43.3.27	<code>toString</code>	333
4.43.3.28	<code>validateInput</code>	334
4.43.4	Documentation des données membres	334
4.43.4.1	<code>m_ioContent</code>	334
4.43.4.2	<code>m_ioStatesHistory</code>	334
4.43.4.3	<code>m_tableName</code>	335
4.44	Référence de la classe <code>SH_Keyboard</code>	335

4.44.1 Description détaillée	336
4.44.2 Documentation des propriétés	336
4.44.2.1 actionsList	336
4.45 Référence de la classe Sh_LoopingInOutStateMachine	336
4.45.1 Description détaillée	339
4.45.2 Documentation des constructeurs et destructeur	339
4.45.2.1 Sh_LoopingInOutStateMachine	339
4.45.3 Documentation des fonctions membres	339
4.45.3.1 addChildsNextTransition	339
4.45.3.2 addIOState	341
4.45.3.3 addIOSStateMachine	343
4.45.3.4 cancelReplacement	344
4.45.3.5 clearAll	344
4.45.3.6 confirmInput	345
4.45.3.7 current	345
4.45.3.8 displayCalendar	346
4.45.3.9 displayFileDialog	346
4.45.3.10 getContentValue	347
4.45.3.11 historyValue	347
4.45.3.12 ioContent	348
4.45.3.13 ioStatesHistory	348
4.45.3.14 limit	349
4.45.3.15 limitChanged	349
4.45.3.16 next	349
4.45.3.17 receiveInput	350
4.45.3.18 replaceInput	350
4.45.3.19 resendText	351
4.45.3.20 sendText	351
4.45.3.21 setContentValue	351
4.45.3.22 setCurrent	352
4.45.3.23 setIOcontent	352
4.45.3.24 setIOSStateMachineHistory	353
4.45.3.25 setLimit	353
4.45.3.26 setPersistentContentValue	354
4.45.3.27 setTableName	354
4.45.3.28 stopLooping	355
4.45.3.29 tableName	355
4.45.3.30 toString	356
4.45.3.31 validateInput	357
4.45.4 Documentation des données membres	357

4.45.4.1	<code>m_contents</code>	357
4.45.4.2	<code>m_current</code>	357
4.45.4.3	<code>m_ioContent</code>	357
4.45.4.4	<code>m_ioStatesHistory</code>	357
4.45.4.5	<code>m_limit</code>	358
4.45.4.6	<code>m_persistentContent</code>	358
4.45.4.7	<code>m_tableName</code>	358
4.45.5	Documentation des propriétés	358
4.45.5.1	<code>limit</code>	358
4.46	Référence de la classe <code>SH_MessageManager</code>	358
4.46.1	Description détaillée	360
4.46.2	Documentation des énumérations membres	360
4.46.2.1	<code>ErrorMode</code>	360
4.46.3	Documentation des fonctions membres	360
4.46.3.1	<code>errorMessage</code>	360
4.46.3.2	<code>infoMessage</code>	361
4.46.3.3	<code>successMessage</code>	361
4.47	Référence de la classe <code>SH_NamedObject</code>	362
4.47.1	Description détaillée	363
4.47.2	Documentation des constructeurs et destructeur	363
4.47.2.1	<code>SH_NamedObject</code>	363
4.47.3	Documentation des fonctions membres	364
4.47.3.1	<code>name</code>	364
4.47.3.2	<code>ptraddress</code>	364
4.47.3.3	<code>setName</code>	364
4.47.3.4	<code>toString</code>	365
4.47.4	Documentation des données membres	365
4.47.4.1	<code>m_name</code>	365
4.47.4.2	<code>m_ptraddress</code>	365
4.48	Référence de la classe <code>SH_NumericQuestionState</code>	366
4.48.1	Description détaillée	368
4.48.2	Documentation des constructeurs et destructeur	368
4.48.2.1	<code>SH_NumericQuestionState</code>	368
4.48.3	Documentation des fonctions membres	369
4.48.3.1	<code>answerInvalid</code>	369
4.48.3.2	<code>answerValid</code>	369
4.48.3.3	<code>checkValidity</code>	369
4.48.3.4	<code>display</code>	370
4.48.3.5	<code>givenAnswer</code>	370
4.48.3.6	<code>input</code>	371

4.48.3.7	isAnswerValid	371
4.48.3.8	max	371
4.48.3.9	min	372
4.48.3.10	next	372
4.48.3.11	onEntry	373
4.48.3.12	onExit	373
4.48.3.13	output	374
4.48.3.14	rawInput	374
4.48.3.15	resendInput	375
4.48.3.16	sendOutput	375
4.48.3.17	setGivenAnswer	376
4.48.3.18	setInput	376
4.48.3.19	setMax	377
4.48.3.20	setMin	377
4.48.3.21	setOutput	378
4.48.3.22	setVisibility	378
4.48.3.23	toString	379
4.48.3.24	visibility	380
4.48.4	Documentation des données membres	380
4.48.4.1	m_max	380
4.48.4.2	m_min	380
4.49	Référence de la classe SH_OutputZone	381
4.49.1	Description détaillée	382
4.49.2	Documentation des fonctions membres	382
4.49.2.1	clear	383
4.49.2.2	clearAll	383
4.49.2.3	display	383
4.49.2.4	displayCalendar	383
4.49.2.5	displayNew	383
4.49.2.6	displayNewFixed	383
4.49.2.7	displaySqlDatas	383
4.49.2.8	displaySqlDetail	383
4.49.2.9	displayText	383
4.49.2.10	replace	383
4.49.2.11	selected	383
4.49.3	Documentation des propriétés	383
4.49.3.1	lastVisibleRow	383
4.50	Référence de la classe SH_PrintingState	383
4.50.1	Description détaillée	386
4.50.2	Documentation des constructeurs et destructeur	386

4.50.2.1	SH_PrintingState	386
4.50.3	Documentation des fonctions membres	386
4.50.3.1	next	386
4.50.3.2	onEntry	386
4.50.3.3	onExit	387
4.50.3.4	printFinished	388
4.50.3.5	printStarted	388
4.50.3.6	toString	388
4.51	Référence de la classe SH_QuestionState	389
4.51.1	Description détaillée	391
4.51.2	Documentation des constructeurs et destructeur	391
4.51.2.1	SH_QuestionState	391
4.51.3	Documentation des fonctions membres	391
4.51.3.1	answerInvalid	391
4.51.3.2	answerValid	392
4.51.3.3	checkValidity	392
4.51.3.4	display	393
4.51.3.5	givenAnswer	393
4.51.3.6	input	394
4.51.3.7	isAnswerValid	394
4.51.3.8	next	394
4.51.3.9	onEntry	395
4.51.3.10	onExit	395
4.51.3.11	output	396
4.51.3.12	rawInput	396
4.51.3.13	resendInput	397
4.51.3.14	sendOutput	397
4.51.3.15	setGivenAnswer	397
4.51.3.16	setInput	398
4.51.3.17	setOutput	399
4.51.3.18	setVisibility	399
4.51.3.19	toString	400
4.51.3.20	visibility	401
4.51.4	Documentation des données membres	401
4.51.4.1	m_givenAnswer	401
4.52	Référence de la classe SH_RegExpQuestionState	401
4.52.1	Description détaillée	404
4.52.2	Documentation des constructeurs et destructeur	404
4.52.2.1	SH_RegExpQuestionState	404
4.52.3	Documentation des fonctions membres	404

4.52.3.1	answerInvalid	405
4.52.3.2	answerValid	405
4.52.3.3	checkValidity	405
4.52.3.4	display	406
4.52.3.5	givenAnswer	406
4.52.3.6	input	407
4.52.3.7	isAnswerValid	407
4.52.3.8	maxLen	407
4.52.3.9	minLen	408
4.52.3.10	next	408
4.52.3.11	onEntry	409
4.52.3.12	onExit	409
4.52.3.13	output	410
4.52.3.14	rawInput	410
4.52.3.15	regexp	411
4.52.3.16	resendInput	411
4.52.3.17	sendOutput	412
4.52.3.18	setGivenAnswer	412
4.52.3.19	setInput	413
4.52.3.20	setMaxLen	413
4.52.3.21	setMinLen	414
4.52.3.22	setOutput	414
4.52.3.23	setRegexp	415
4.52.3.24	setVisibility	415
4.52.3.25	toString	416
4.52.3.26	visibility	417
4.52.4	Documentation des données membres	417
4.52.4.1	m_regexp	417
4.53	Référence de la classe SH_RoomsDelegate	418
4.53.1	Description détaillée	419
4.53.2	Documentation des propriétés	419
4.53.2.1	fontSize	419
4.53.2.2	value	419
4.54	Référence de la classe SH_RoomsSectionsDelegate	420
4.54.1	Description détaillée	421
4.54.2	Documentation des propriétés	421
4.54.2.1	value	421
4.55	Référence de la classe SH_RoomsTableModel	421
4.55.1	Description détaillée	424
4.55.2	Documentation des constructeurs et destructeur	424

4.55.2.1	SH_RoomsTableModel	424
4.55.3	Documentation des fonctions membres	425
4.55.3.1	addFilterKeyColumn	425
4.55.3.2	containsFilterKeyColumn	425
4.55.3.3	currentSortKeyColumn	425
4.55.3.4	data	426
4.55.3.5	data	426
4.55.3.6	fetch	427
4.55.3.7	field	427
4.55.3.8	fields	428
4.55.3.9	fieldsCount	428
4.55.3.10	fillModel	429
4.55.3.11	filterAcceptsRow	429
4.55.3.12	flags	430
4.55.3.13	invalidateFilter	430
4.55.3.14	isEmpty	430
4.55.3.15	lastError	431
4.55.3.16	removeFilterKeyColumn	431
4.55.3.17	roleNames	431
4.55.3.18	setBooleanColumns	431
4.55.3.19	setData	432
4.55.3.20	setNotNullColumns	432
4.55.3.21	setNullColumns	432
4.55.3.22	setPasswordColumns	433
4.55.3.23	setReadOnlyColumns	433
4.55.3.24	setSortKeyColumn	434
4.55.3.25	sort	434
4.55.3.26	sortChanged	435
4.55.3.27	tableName	435
4.55.4	Documentation des données membres	436
4.55.4.1	model	436
4.55.5	Documentation des propriétés	436
4.55.5.1	empty	436
4.55.5.2	fieldsList	436
4.55.5.3	lastError	436
4.55.5.4	sortKeyColumn	436
4.55.5.5	table	436
4.56	Référence de la classe SH_ServiceCharging	437
4.56.1	Description détaillée	439
4.56.2	Documentation des constructeurs et destructeur	439

4.56.2.1	SH_ServiceCharging	439
4.56.3	Documentation des fonctions membres	441
4.56.3.1	addChildrenNextTransition	441
4.56.3.2	addIOState	443
4.56.3.3	addIOMachine	445
4.56.3.4	cancelReplacement	446
4.56.3.5	clearAll	446
4.56.3.6	confirmInput	446
4.56.3.7	current	447
4.56.3.8	displayCalendar	447
4.56.3.9	displayFileDialog	448
4.56.3.10	getContentValue	448
4.56.3.11	historyValue	449
4.56.3.12	ioContent	449
4.56.3.13	ioStatesHistory	450
4.56.3.14	limit	450
4.56.3.15	limitChanged	450
4.56.3.16	next	451
4.56.3.17	receiveInput	451
4.56.3.18	replaceInput	451
4.56.3.19	resendText	452
4.56.3.20	sendText	452
4.56.3.21	setContentValue	453
4.56.3.22	setCurrent	454
4.56.3.23	setIOcontent	454
4.56.3.24	setIOMachine	454
4.56.3.25	setLimit	455
4.56.3.26	setPersistentContentValue	455
4.56.3.27	setTableName	456
4.56.3.28	stopLooping	456
4.56.3.29	tableName	457
4.56.3.30	toString	457
4.56.3.31	validateInput	458
4.56.4	Documentation des données membres	458
4.56.4.1	m_ioContent	458
4.56.4.2	m_ioStatesHistory	459
4.56.4.3	m_priceMin	459
4.56.4.4	m_tableName	459
4.56.4.5	m_vat	459
4.56.5	Documentation des propriétés	459

4.56.5.1	limit	459
4.57	Référence de la classe SH_ServicesDelegate	460
4.57.1	Description détaillée	461
4.57.2	Documentation des propriétés	461
4.57.2.1	fontSize	461
4.57.2.2	value	461
4.58	Référence de la classe SH_ServicesTableModel	462
4.58.1	Description détaillée	464
4.58.2	Documentation des constructeurs et destructeur	464
4.58.2.1	SH_ServicesTableModel	464
4.58.3	Documentation des fonctions membres	465
4.58.3.1	addFilterKeyColumn	465
4.58.3.2	containsFilterKeyColumn	465
4.58.3.3	currentSortKeyColumn	465
4.58.3.4	data	466
4.58.3.5	data	466
4.58.3.6	fetch	467
4.58.3.7	field	467
4.58.3.8	fields	468
4.58.3.9	fieldsCount	468
4.58.3.10	fillModel	469
4.58.3.11	filterAcceptsRow	469
4.58.3.12	flags	469
4.58.3.13	invalidateFilter	470
4.58.3.14	isEmpty	470
4.58.3.15	lastError	470
4.58.3.16	removeFilterKeyColumn	470
4.58.3.17	roleNames	471
4.58.3.18	setBooleanColumns	471
4.58.3.19	setData	471
4.58.3.20	setNotNullColumns	472
4.58.3.21	setNullColumns	472
4.58.3.22	setPasswordColumns	473
4.58.3.23	setReadOnlyColumns	473
4.58.3.24	setSortKeyColumn	473
4.58.3.25	sort	474
4.58.3.26	sortChanged	475
4.58.3.27	tableName	475
4.58.4	Documentation des données membres	475
4.58.4.1	model	475

4.58.5 Documentation des propriétés	476
4.58.5.1 empty	476
4.58.5.2 fieldsList	476
4.58.5.3 lastError	476
4.58.5.4 sortKeyColumn	476
4.58.5.5 table	476
4.59 Référence de la classe SH_SqlDataFields	476
4.59.1 Description détaillée	479
4.59.2 Documentation des constructeurs et destructeur	479
4.59.2.1 SH_SqlDataFields	479
4.59.3 Documentation des fonctions membres	479
4.59.3.1 name	479
4.59.3.2 nameChanged	479
4.59.3.3 role	480
4.59.3.4 roleChanged	480
4.59.3.5 setName	480
4.59.3.6 setSortOrder	481
4.59.3.7 setText	481
4.59.3.8 sortOrder	482
4.59.3.9 sortOrderChanged	482
4.59.3.10 text	482
4.59.3.11 textChanged	482
4.59.4 Documentation des données membres	483
4.59.4.1 m_name	483
4.59.4.2 m_sortOrder	483
4.59.4.3 m_text	483
4.59.5 Documentation des propriétés	483
4.59.5.1 name	483
4.59.5.2 role	483
4.59.5.3 sortOrder	483
4.59.5.4 text	483
4.60 Référence de la classe SH_SqlDataModel	483
4.60.1 Description détaillée	486
4.60.2 Documentation des constructeurs et destructeur	486
4.60.2.1 SH_SqlDataModel	486
4.60.3 Documentation des fonctions membres	486
4.60.3.1 applyRoles	486
4.60.3.2 data	487
4.60.3.3 datas	488
4.60.3.4 fetch	489

4.60.3.5 field	491
4.60.3.6 fieldFromRole	491
4.60.3.7 fieldsChanged	492
4.60.3.8 fieldsCount	492
4.60.3.9 fieldsList	492
4.60.3.10 filter	493
4.60.3.11 filterChanged	493
4.60.3.12 isEmpty	493
4.60.3.13 lastError	494
4.60.3.14 lastErrorChanged	494
4.60.3.15 query	494
4.60.3.16 resetFieldsToAll	494
4.60.3.17 resetFilterCondition	495
4.60.3.18 roleForField	495
4.60.3.19 roleNames	495
4.60.3.20 rolesChanged	496
4.60.3.21 rowCount	496
4.60.3.22 setFields	496
4.60.3.23 setFilterCondition	497
4.60.3.24 setHeaderData	497
4.60.3.25 setOrderBy	498
4.60.3.26 setTable	498
4.60.3.27 tableChanged	499
4.60.3.28 tableName	500
4.60.4 Documentation des données membres	500
4.60.4.1 mDataFields	500
4.60.4.2 mFilter	500
4.60.4.3 mRecords	500
4.60.4.4 mRoles	500
4.60.4.5 mSort	501
4.60.4.6 mSqlQuery	501
4.60.4.7 mTable	501
4.60.5 Documentation des propriétés	501
4.60.5.1 filter	501
4.60.5.2 lastError	501
4.60.5.3 table	501
4.61 Référence de la classe SH_SqlDataView	502
4.61.1 Description détaillée	503
4.61.2 Documentation des fonctions membres	503
4.61.2.1 newItem	503

4.61.2.2	selected	503
4.61.3	Documentation des propriétés	504
4.61.3.1	emptyDelegate	504
4.61.3.2	filterIndicatorsVisible	504
4.61.3.3	filtersTitle	504
4.61.3.4	isEmpty	504
4.61.3.5	itemDelegate	504
4.61.3.6	sectionDelegate	504
4.61.3.7	sqlModel	504
4.62	Référence de la classe SH_SqlTableView	505
4.62.1	Description détaillée	506
4.62.2	Documentation des fonctions membres	506
4.62.2.1	selected	506
4.63	Référence de la classe SH_StatementState	506
4.63.1	Description détaillée	509
4.63.2	Documentation des constructeurs et destructeur	509
4.63.2.1	SH_StatementState	509
4.63.3	Documentation des fonctions membres	509
4.63.3.1	display	509
4.63.3.2	input	510
4.63.3.3	next	510
4.63.3.4	onEntry	511
4.63.3.5	onExit	511
4.63.3.6	output	512
4.63.3.7	rawInput	512
4.63.3.8	resendInput	513
4.63.3.9	sendOutput	513
4.63.3.10	setInput	513
4.63.3.11	setOutput	514
4.63.3.12	setVisibility	514
4.63.3.13	toString	515
4.63.3.14	visibility	516
4.64	Référence de la classe SH_StringQuestionState	516
4.64.1	Description détaillée	519
4.64.2	Documentation des constructeurs et destructeur	519
4.64.2.1	SH_StringQuestionState	519
4.64.3	Documentation des fonctions membres	520
4.64.3.1	answerInvalid	520
4.64.3.2	answerValid	520
4.64.3.3	checkValidity	520

4.64.3.4	display	521
4.64.3.5	givenAnswer	521
4.64.3.6	input	522
4.64.3.7	isValid	522
4.64.3.8	maxLen	523
4.64.3.9	minLen	523
4.64.3.10	next	523
4.64.3.11	onEntry	524
4.64.3.12	onExit	524
4.64.3.13	output	525
4.64.3.14	rawInput	525
4.64.3.15	resendInput	526
4.64.3.16	sendOutput	526
4.64.3.17	setGivenAnswer	527
4.64.3.18	setInput	527
4.64.3.19	setMaxLen	528
4.64.3.20	setMinLen	528
4.64.3.21	setOutput	529
4.64.3.22	setVisibility	529
4.64.3.23	toString	530
4.64.3.24	visibility	531
4.64.4	Documentation des données membres	531
4.64.4.1	m_maxLen	531
4.64.4.2	m_minLen	531
4.65	Référence de la classe SH_TabZone	532
4.65.1	Description détaillée	533
4.65.2	Documentation des fonctions membres	533
4.65.2.1	newBilling	533
4.65.2.2	newBooking	533
4.65.2.3	newSelling	533
4.65.2.4	openTab	533
4.65.2.5	reload	533
4.65.2.6	selected	533
4.65.2.7	selectedForDetail	533
4.65.3	Documentation des propriétés	533
4.65.3.1	stdKeyboard	533
4.66	Référence de la classe SH_Trainee	533
4.66.1	Description détaillée	536
4.66.2	Documentation des constructeurs et destructeur	536
4.66.2.1	SH_Trainee	536

4.66.3 Documentation des fonctions membres	536
4.66.3.1 exists	536
4.66.3.2 id	537
4.66.3.3 isAdministrator	537
4.66.3.4 isManagerX	538
4.66.3.5 isManagerZ	538
4.66.3.6 isReceptionist	539
4.66.3.7 isValid	539
4.66.3.8 logIn	539
4.66.3.9 name	541
4.66.3.10 nameChanged	541
4.66.3.11 roles	541
4.66.3.12 rolesChanged	541
4.66.3.13 traineeExists	541
4.66.3.14 userExists	542
4.66.3.15 validityChanged	542
4.66.4 Documentation des propriétés	542
4.66.4.1 administrator	542
4.66.4.2 id	542
4.66.4.3 managerX	542
4.66.4.4 managerZ	543
4.66.4.5 name	543
4.66.4.6 receptionist	543
4.66.4.7 roles	543
4.66.4.8 valid	543
4.67 Référence de la classe SH_TriStateCheckImage	544
4.67.1 Description détaillée	545
4.67.2 Documentation des fonctions membres	545
4.67.2.1 pressed	545
4.67.3 Documentation des propriétés	546
4.67.3.1 checked	546
4.67.3.2 imgDown	546
4.67.3.3 imgUp	546
4.67.3.4 innerHeight	546
4.67.3.5 innerWidth	546
4.67.3.6 labelSpacing	546
4.67.3.7 length	546
4.67.3.8 text	546
4.67.3.9 up	546
4.68 Référence de la classe SH_User	546

4.68.1	Description détaillée	549
4.68.2	Documentation des constructeurs et destructeur	549
4.68.2.1	SH_User	549
4.68.3	Documentation des fonctions membres	550
4.68.3.1	exists	551
4.68.3.2	id	551
4.68.3.3	isAdministrator	552
4.68.3.4	isManagerX	552
4.68.3.5	isManagerZ	552
4.68.3.6	isReceptionist	553
4.68.3.7	isValid	553
4.68.3.8	logIn	554
4.68.3.9	name	555
4.68.3.10	nameChanged	555
4.68.3.11	roles	555
4.68.3.12	rolesChanged	555
4.68.3.13	setID	555
4.68.3.14	setName	556
4.68.3.15	traineeExists	557
4.68.3.16	userExists	557
4.68.3.17	validityChanged	558
4.68.4	Documentation des données membres	558
4.68.4.1	m_administrator	558
4.68.4.2	m_id	558
4.68.4.3	m_managerX	558
4.68.4.4	m_managerZ	558
4.68.4.5	m_name	558
4.68.4.6	m_receptionist	559
4.68.5	Documentation des propriétés	559
4.68.5.1	administrator	559
4.68.5.2	id	559
4.68.5.3	managerX	559
4.68.5.4	managerZ	559
4.68.5.5	name	559
4.68.5.6	receptionist	559
4.68.5.7	roles	559
4.68.5.8	valid	559
4.69	Référence de la classe SH_ValidationState	560
4.69.1	Description détaillée	562
4.69.2	Documentation des constructeurs et destructeur	562

4.69.2.1	SH_ValidationState	562
4.69.3	Documentation des fonctions membres	562
4.69.3.1	confirmInput	562
4.69.3.2	display	563
4.69.3.3	input	563
4.69.3.4	next	564
4.69.3.5	onEntry	564
4.69.3.6	onExit	565
4.69.3.7	output	565
4.69.3.8	rawInput	565
4.69.3.9	resendInput	566
4.69.3.10	sendOutput	566
4.69.3.11	setInput	567
4.69.3.12	setOutput	567
4.69.3.13	setVisibility	568
4.69.3.14	toString	568
4.69.3.15	visibility	569
4.70	Référence de la classe SH_WelcomePage	570
4.70.1	Description détaillée	571
4.70.2	Documentation des fonctions membres	571
4.70.2.1	clicked	571
4.70.2.2	loggedOut	571
4.70.2.3	logOut	571
4.70.2.4	quit	571
4.70.2.5	reload	571
4.71	Référence de la classe TableView	571
4.72	Référence de la classe TabView	572
5	Documentation des fichiers	575
5.1	Référence du fichier logic/SH_AdaptDatabaseState.cpp	575
5.2	SH_AdaptDatabaseState.cpp	575
5.3	Référence du fichier logic/SH_AdaptDatabaseState.h	576
5.4	SH_AdaptDatabaseState.h	576
5.5	Référence du fichier logic/SH_AddressCreation.cpp	577
5.6	SH_AddressCreation.cpp	577
5.7	Référence du fichier logic/SH_AddressCreation.h	577
5.8	SH_AddressCreation.h	578
5.9	Référence du fichier logic/SH_BillingCreation.cpp	579
5.10	SH_BillingCreation.cpp	579
5.11	Référence du fichier logic/SH_BillingCreation.h	581

5.12 SH_BillingCreation.h	582
5.13 Référence du fichier logic/SH_ClientCreation.cpp	583
5.14 SH_ClientCreation.cpp	583
5.15 Référence du fichier logic/SH_ClientCreation.h	584
5.16 SH_ClientCreation.h	585
5.17 Référence du fichier logic/SH_ConfirmationState.cpp	585
5.18 SH_ConfirmationState.cpp	586
5.19 Référence du fichier logic/SH_ConfirmationState.h	587
5.20 SH_ConfirmationState.h	588
5.21 Référence du fichier logic/SH_DatabaseContentQuestionState.cpp	588
5.22 SH_DatabaseContentQuestionState.cpp	588
5.23 Référence du fichier logic/SH_DatabaseContentQuestionState.h	589
5.24 SH_DatabaseContentQuestionState.h	590
5.25 Référence du fichier logic/SH_DateQuestionState.cpp	591
5.26 SH_DateQuestionState.cpp	592
5.27 Référence du fichier logic/SH_DateQuestionState.h	593
5.28 SH_DateQuestionState.h	594
5.29 Référence du fichier logic/SH.DecimalQuestionState.cpp	595
5.30 SH.DecimalQuestionState.cpp	596
5.31 Référence du fichier logic/SH.DecimalQuestionState.h	597
5.32 SH.DecimalQuestionState.h	599
5.33 Référence du fichier logic/SH_FileSelectionState.cpp	599
5.34 SH_FileSelectionState.cpp	600
5.35 Référence du fichier logic/SH_FileSelectionState.h	600
5.36 SH_FileSelectionState.h	601
5.37 Référence du fichier logic/SH_GenericDebugableState.cpp	602
5.38 SH_GenericDebugableState.cpp	602
5.39 Référence du fichier logic/SH_GenericDebugableState.h	603
5.40 SH_GenericDebugableState.h	604
5.41 Référence du fichier logic/SH_IOState.cpp	604
5.42 SH_IOState.cpp	605
5.43 Référence du fichier logic/SH_IOState.h	606
5.44 SH_IOState.h	607
5.45 Référence du fichier logic/SH_IOStateMachine.cpp	607
5.46 SH_IOStateMachine.cpp	608
5.47 Référence du fichier logic/SH_IOStateMachine.h	611
5.48 SH_IOStateMachine.h	612
5.49 Référence du fichier logic/SH_LoopingIOStateMachine.cpp	613
5.50 SH_LoopingIOStateMachine.cpp	613
5.51 Référence du fichier logic/SH_LoopingIOStateMachine.h	615

5.52 SH_LoopingIOStateMachine.h	616
5.53 Référence du fichier logic/SH_NamedObject.cpp	616
5.54 SH_NamedObject.cpp	617
5.55 Référence du fichier logic/SH_NamedObject.h	617
5.56 SH_NamedObject.h	618
5.57 Référence du fichier logic/SH_NumericQuestionState.cpp	618
5.58 SH_NumericQuestionState.cpp	619
5.59 Référence du fichier logic/SH_NumericQuestionState.h	620
5.60 SH_NumericQuestionState.h	622
5.61 Référence du fichier logic/SH_PrintingState.cpp	622
5.62 SH_PrintingState.cpp	623
5.63 Référence du fichier logic/SH_PrintingState.h	623
5.64 SH_PrintingState.h	624
5.65 Référence du fichier logic/SH_QuestionState.cpp	625
5.66 SH_QuestionState.cpp	625
5.67 Référence du fichier logic/SH_QuestionState.h	626
5.68 SH_QuestionState.h	627
5.69 Référence du fichier logic/SH_RegExpQuestionState.cpp	628
5.70 SH_RegExpQuestionState.cpp	629
5.71 Référence du fichier logic/SH_RegExpQuestionState.h	630
5.72 SH_RegExpQuestionState.h	631
5.73 Référence du fichier logic/SH_ServiceCharging.cpp	631
5.74 SH_ServiceCharging.cpp	632
5.75 Référence du fichier logic/SH_ServiceCharging.h	633
5.76 SH_ServiceCharging.h	634
5.77 Référence du fichier logic/SH_StatementState.cpp	635
5.78 SH_StatementState.cpp	635
5.79 Référence du fichier logic/SH_StatementState.h	636
5.80 SH_StatementState.h	637
5.81 Référence du fichier logic/SH_StringQuestionState.cpp	637
5.82 SH_StringQuestionState.cpp	638
5.83 Référence du fichier logic/SH_StringQuestionState.h	639
5.84 SH_StringQuestionState.h	640
5.85 Référence du fichier logic/SH_ValidationState.cpp	640
5.86 SH_ValidationState.cpp	641
5.87 Référence du fichier logic/SH_ValidationState.h	642
5.88 SH_ValidationState.h	643
5.89 Référence du fichier main.cpp	643
5.89.1 Documentation des fonctions	644
5.89.1.1 enableLogging	644

5.89.1.2	exportlog	644
5.89.1.3	main	645
5.89.1.4	spin	646
5.89.1.5	statusChanged	646
5.89.2	Documentation des variables	647
5.89.2.1	iterations	647
5.90	main.cpp	647
5.91	Référence du fichier models/SH_BillingsTableModel.cpp	649
5.92	SH_BillingsTableModel.cpp	649
5.93	Référence du fichier models/SH_BillingsTableModel.h	650
5.94	SH_BillingsTableModel.h	651
5.95	Référence du fichier models/SH_BillsTableModel.cpp	651
5.96	SH_BillsTableModel.cpp	651
5.97	Référence du fichier models/SH_BillsTableModel.h	652
5.98	SH_BillsTableModel.h	652
5.99	Référence du fichier models/SH_BookingsTableModel.cpp	653
5.100	SH_BookingsTableModel.cpp	653
5.101	Référence du fichier models/SH_BookingsTableModel.h	654
5.102	SH_BookingsTableModel.h	655
5.103	Référence du fichier models/SH_ClientsTableModel.cpp	655
5.104	SH_ClientsTableModel.cpp	655
5.105	Référence du fichier models/SH_ClientsTableModel.h	656
5.106	SH_ClientsTableModel.h	656
5.107	Référence du fichier models/SH_Company.cpp	657
5.108	SH_Company.cpp	657
5.109	Référence du fichier models/SH_Company.h	657
5.110	SH_Company.h	658
5.111	Référence du fichier models/SH_ExtendedProxyModel.cpp	658
5.112	SH_ExtendedProxyModel.cpp	659
5.113	Référence du fichier models/SH_ExtendedSqlProxyModel.h	661
5.114	SH_ExtendedSqlProxyModel.h	662
5.115	Référence du fichier models/SH_GroupsTableModel.cpp	663
5.116	SH_GroupsTableModel.cpp	664
5.117	Référence du fichier models/SH_GroupsTableModel.h	664
5.118	SH_GroupsTableModel.h	665
5.119	Référence du fichier models/SH_RoomsTableModel.cpp	666
5.120	SH_RoomsTableModel.cpp	666
5.121	Référence du fichier models/SH_RoomsTableModel.h	667
5.122	SH_RoomsTableModel.h	667
5.123	Référence du fichier models/SH_ServicesTableModel.cpp	668

5.124SH_ServicesTableModel.cpp	668
5.125Référence du fichier models/SH_ServicesTableModel.h	669
5.126SH_ServicesTableModel.h	669
5.127Référence du fichier models/SH_SqlDataField.cpp	670
5.128SH_SqlDataField.cpp	670
5.129Référence du fichier models/SH_SqlDataField.h	671
5.130SH_SqlDataField.h	671
5.131Référence du fichier models/SH_SqlDataManager.cpp	672
5.132SH_SqlDataManager.cpp	672
5.133Référence du fichier models/SH_SqlDataManager.h	676
5.134SH_SqlDataManager.h	676
5.135Référence du fichier models/SH_Trainee.cpp	677
5.136SH_Trainee.cpp	678
5.137Référence du fichier models/SH_Trainee.h	678
5.138SH_Trainee.h	679
5.139Référence du fichier models/SH_User.cpp	680
5.140SH_User.cpp	680
5.141Référence du fichier models/SH_User.h	681
5.142SH_User.h	682
5.143Référence du fichier SH_ApplicationCore.cpp	683
5.144SH_ApplicationCore.cpp	684
5.145Référence du fichier SH_ApplicationCore.h	686
5.146SH_ApplicationCore.h	687
5.147Référence du fichier SH_DatabaseManager.cpp	687
5.148SH_DatabaseManager.cpp	688
5.149Référence du fichier SH_DatabaseManager.h	691
5.149.1 Documentation des variables	692
5.149.1.1 dbAliasNameStr	692
5.149.1.2 dbCannotOpenStr	692
5.149.1.3 dbDriverNotExistStr	692
5.149.1.4 dbDriverStr	692
5.149.1.5 dbFileNameStr	692
5.149.1.6 dbFilePathStr	692
5.149.1.7 dbFolderPathStr	692
5.149.1.8 dbPasswordStr	692
5.149.1.9 dbUsernameStr	693
5.150SH_DatabaseManager.h	693
5.151Référence du fichier SH_MessageManager.cpp	694
5.152SH_MessageManager.cpp	694
5.153Référence du fichier SH_MessageManager.h	695

5.154SH_MessageManager.h	695
5.155Référence du fichier views/qml/SH_app.qml	696
5.156SH_app.qml	696
5.157Référence du fichier views/qml/SH_BillingsDelegate.qml	697
5.158SH_BillingsDelegate.qml	697
5.159Référence du fichier views/qml/SH_BookingsDelegate.qml	697
5.160SH_BookingsDelegate.qml	698
5.161Référence du fichier views/qml/SH_CalendarDialog.qml	698
5.162SH_CalendarDialog.qml	698
5.163Référence du fichier views/qml/SH_CommonPage.qml	701
5.164SH_CommonPage.qml	701
5.165Référence du fichier views/qml/SH_ConexionPage.qml	711
5.166SH_ConexionPage.qml	711
5.167Référence du fichier views/qml/SH_ContentView.qml	712
5.168SH_ContentView.qml	712
5.169Référence du fichier views/qml/SH_DataDelegate.qml	716
5.170SH_DataDelegate.qml	716
5.171Référence du fichier views/qml/SH_HeaderView.qml	717
5.172SH_HeaderView.qml	717
5.173Référence du fichier views/qml/SH_Keyboard.qml	718
5.174SH_Keyboard.qml	718
5.175Référence du fichier views/qml/SH_OutputZone.qml	718
5.176SH_OutputZone.qml	719
5.177Référence du fichier views/qml/SH_RoomsDelegate.qml	721
5.178SH_RoomsDelegate.qml	721
5.179Référence du fichier views/qml/SH_RoomsSectionsDelegate.qml	722
5.180SH_RoomsSectionsDelegate.qml	722
5.181Référence du fichier views/qml/SH_ServicesDelegate.qml	722
5.182SH_ServicesDelegate.qml	722
5.183Référence du fichier views/qml/SH_SqlDataView.qml	722
5.184SH_SqlDataView.qml	723
5.185Référence du fichier views/qml/SH_SqlTableView.qml	724
5.186SH_SqlTableView.qml	724
5.187Référence du fichier views/qml/SH_TabZone.qml	725
5.188SH_TabZone.qml	725
5.189Référence du fichier views/qml/SH_TriStateCheckImage.qml	727
5.190SH_TriStateCheckImage.qml	727
5.191Référence du fichier views/qml/SH_WelcomePage.qml	728
5.192SH_WelcomePage.qml	728
5.193Référence du fichier views/SH_ExtendedQQmlAction.cpp	730

5.193.1 Documentation des fonctions	730
5.193.1.1 qShortcutContextMatcher	730
5.194 SH_ExtendedQQmlAction.cpp	731
5.195 Référence du fichier views/Series_ExtendedQQmlAction.h	733
5.196 SH_ExtendedQQmlAction.h	734
Index	736

Chapitre 1

Index hiérarchique

1.1 Hiérarchie des classes

Cette liste d'héritage est classée approximativement par ordre alphabétique :

ApplicationWindow	9
SH_app	45
Button	10
SH_DataDelegate	207
SH_BillingsDelegate	80
SH_BookingsDelegate	112
SH_RoomsDelegate	418
SH_ServicesDelegate	460
FocusScope	11
SH_TriStateCheckImage	544
GridLayout	12
SH_ContentView	180
SH_HeaderView	304
SH_Keyboard	335
Item	13
SH_CommonPage	166
SH_ConexionPage	179
SH_WelcomePage	570
QAbstractListModel	14
SH_SqlDataModel	483
QObject	15
SH_ApplicationCore	46
SH_Company	167
SH_DatabaseManager	197
SH_ExtendedQQmlAction	254
SH_MessageManager	358
SH_User	546
SH_Trainee	533
QQuickItem	17
SH_SqlDataFields	476
QSortFilterProxyModel	18
SH_ExtendedProxyModel	237
SH_BillingsTableModel	82
SH_BillsTableModel	97
SH_BookingsTableModel	114

SH_ClientsTableModel	151
SH_GroupsTableModel	288
SH_RoomsTableModel	421
SH_ServicesTableModel	462
QState	19
SH_GenericState	281
SH_AdaptDatabaseState	21
SH_InOutState	305
SH_FileSelectionState	270
SH_QuestionState	389
SH_DatabaseContentQuestionState	182
SH_DateQuestionState	208
SH.DecimalQuestionState	222
SH_NumericQuestionState	366
SH_StringQuestionState	516
SH_RegExpQuestionState	401
SH_StatementState	506
SH_ConfirmationState	169
SH_ValidationState	560
SH_PrintingState	383
QStateMachine	20
SH_InOutStateMachine	316
SH_AddressCreationStateMachine	27
SH_BillingCreationStateMachine	60
SH_ClientCreationStateMachine	132
Sh_LoopingInOutStateMachine	336
SH_ServiceCharging	437
Rectangle	21
SH_CalendarDialog	129
SH_OutputZone	381
SH_RoomsSectionsDelegate	420
SH_SqlDataView	502
SH_	??
SH_AppDatabase	??
SH_CheckableSortFilterProxyMode	??
SH_IOState	??
SH_IOStateMachine	??
SH_LoopingStateMachine	??
SH_NamedObject	362
SH_GenericState	281
SH_InOutStateMachine	316
SH_QQQuickAction	??
SH_RestrictiveApplication	??
TableView	571
SH_SqlTableView	505
TabView	572
SH_TabZone	532

Chapitre 2

Index des classes

2.1 Liste des classes

Liste des classes, structures, unions et interfaces avec une brève description :

ApplicationWindow	9
Button	10
FocusScope	11
GridLayout	12
Item	13
QAbstractListModel	14
QObject	15
QQuickItem	17
QSortFilterProxyModel	18
QState	19
QStateMachine	20
Rectangle	21
SH_AdaptDatabaseState	21
SH_AddressCreationStateMachine	
The SH_AddressCreationStateMachine class	27
SH_app	45
SH_ApplicationCore	46
SH_BillingCreationStateMachine	
The SH_BillingCreationStateMachine class	60
SH_BillingsDelegate	80
SH_BillingsTableModel	82
SH_BillsTableModel	97
SH_BookingsDelegate	112
SH_BookingsTableModel	114
SH_CalendarDialog	129
SH_ClientCreationStateMachine	
The SH_ClientCreationStateMachine class	132
SH_ClientsTableModel	151
SH_CommonPage	166
SH_Company	167
SH_ConfirmationState	
La class ConfirmationState représente un état dans lequel le système attend que l'utilisateur appuie sur une touche de confirmation	169
SH_ConexionPage	179
SH_ContentView	180
SH_DatabaseContentQuestionState	182
SH_DatabaseManager	197
SH_DataDelegate	207

SH_DateQuestionState	208
SH.DecimalQuestionState	222
SH.ExtendedProxyModel	237
SH.ExtendedQQmlAction	254
SH.FileSelectionState	
The SH_FileSelectionState class	270
SH.GenericState	281
SH.GroupsTableModel	288
SH.HeaderView	304
SH.InOutState	305
SH.InOutStateMachine	316
SH.Keyboard	335
Sh.LoopingInOutStateMachine	336
SH.MessageManager	358
SH.NamedObject	362
SH.NumericQuestionState	366
SH.OutputZone	381
SH.PrintingState	383
SH.QuestionState	389
SH.RegExpQuestionState	
The SH_RegExpQuestionState class	401
SH.RoomsDelegate	418
SH.RoomsSectionsDelegate	420
SH.RoomsTableModel	421
SH.ServiceCharging	
The SH_ServiceCharging class	437
SH.ServicesDelegate	460
SH.ServicesTableModel	462
SH.SqlDataFields	476
SH.SqlDataManager	483
SH.SqlDataView	502
SH.SqlTableView	505
SH.StatementState	506
SH.StringQuestionState	516
SH.TabZone	532
SH.Trainee	533
SH.TriStateCheckImage	544
SH.User	546
SH.ValidationState	560
SH.WelcomePage	570
TableView	571
TabView	572

Chapitre 3

Index des fichiers

3.1 Liste des fichiers

Liste de tous les fichiers avec une brève description :

main.cpp	643
SH_ApplicationCore.cpp	683
SH_ApplicationCore.h	686
SH_DatabaseManager.cpp	687
SH_DatabaseManager.h	691
SH_MessageManager.cpp	694
SH_MessageManager.h	695
logic/SH_AdaptDatabaseState.cpp	575
logic/SH_AdaptDatabaseState.h	576
logic/SH_AddressCreation.cpp	577
logic/SH_AddressCreation.h	578
logic/SH_BillingCreation.cpp	579
logic/SH_BillingCreation.h	582
logic/SH_ClientCreation.cpp	583
logic/SH_ClientCreation.h	585
logic/SH_ConfirmationState.cpp	586
logic/SH_ConfirmationState.h	588
logic/SH_DatabaseContentQuestionState.cpp	588
logic/SH_DatabaseContentQuestionState.h	590
logic/SH_DateQuestionState.cpp	592
logic/SH_DateQuestionState.h	594
logic/SH.DecimalQuestionState.cpp	596
logic/SH.DecimalQuestionState.h	599
logic/SH_FileSelectionState.cpp	600
logic/SH_FileSelectionState.h	601
logic/SH_GenericDebugableState.cpp	602
logic/SH_GenericDebugableState.h	604
logic/SH_IOSState.cpp	605
logic/SH_IOSState.h	607
logic/SH_IOSStateMachine.cpp	608
logic/SH_IOSStateMachine.h	612
logic/SH_LoopingIOSStateMachine.cpp	613
logic/SH_LoopingIOSStateMachine.h	616
logic/SH_NamedObject.cpp	617
logic/SH_NamedObject.h	618
logic/SH_NumericQuestionState.cpp	619
logic/SH_NumericQuestionState.h	622
logic/SH_PrintingState.cpp	623

logic/SH_PrintingState.h	624
logic/SH_QuestionState.cpp	625
logic/SH_QuestionState.h	627
logic/SH_RegExpQuestionState.cpp	629
logic/SH_RegExpQuestionState.h	631
logic/SH_ServiceCharging.cpp	632
logic/SH_ServiceCharging.h	634
logic/SH_StatementState.cpp	635
logic/SH_StatementState.h	637
logic/SH_StringQuestionState.cpp	638
logic/SH_StringQuestionState.h	640
logic/SH_ValidationState.cpp	641
logic/SH_ValidationState.h	643
models/SH_BillingsTableModel.cpp	649
models/SH_BillingsTableModel.h	651
models/SH_BillsTableModel.cpp	651
models/SH_BillsTableModel.h	652
models/SH_BookingsTableModel.cpp	653
models/SH_BookingsTableModel.h	655
models/SH_ClientsTableModel.cpp	655
models/SH_ClientsTableModel.h	656
models/SH_Company.cpp	657
models/SH_Company.h	658
models/SH_ExtendedProxyModel.cpp	659
models/SH_ExtendedSqlProxyModel.h	662
models/SH_GroupsTableModel.cpp	664
models/SH_GroupsTableModel.h	665
models/SH_RoomsTableModel.cpp	666
models/SH_RoomsTableModel.h	667
models/SH_ServicesTableModel.cpp	668
models/SH_ServicesTableModel.h	669
models/SH_SqlDataField.cpp	670
models/SH_SqlDataField.h	671
models/SH_SqlDataModel.cpp	672
models/SH_SqlDataModel.h	676
models/SH_Trainee.cpp	678
models/SH_Trainee.h	679
models/SH_User.cpp	680
models/SH_User.h	682
views/SH_ExtendedQQmlAction.cpp	731
views/SH_ExtendedQQmlAction.h	734
views/qml/SH_app.qml	696
views/qml/SH_BillingsDelegate.qml	697
views/qml/SH_BookingsDelegate.qml	698
views/qml/SH_CalendarDialog.qml	698
views/qml/SH_CommonPage.qml	701
views/qml/SH_ConexionPage.qml	711
views/qml/SH_ContentView.qml	712
views/qml/SH_DataDelegate.qml	716
views/qml/SH_HeaderView.qml	717
views/qml/SH_Keyboard.qml	718
views/qml/SH_OutputZone.qml	719
views/qml/SH_RoomsDelegate.qml	721
views/qml/SH_RoomsSectionsDelegate.qml	722
views/qml/SH_ServicesDelegate.qml	722
views/qml/SH_SqlDataView.qml	723
views/qml/SH_SqlTableView.qml	724
views/qml/SH_TabZone.qml	725

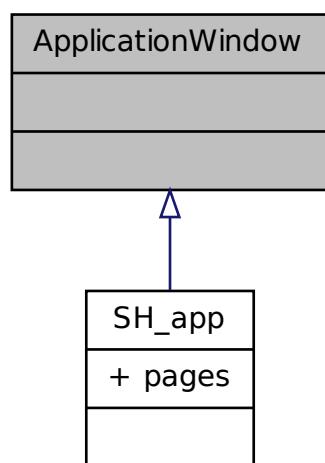
views/qml/SH_TriStateCheckImage.qml	727
views/qml/SH_WelcomePage.qml	728

Chapitre 4

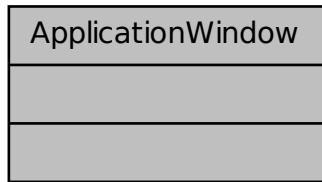
Documentation des classes

4.1 Référence de la classe ApplicationWindow

Graphe d'héritage de ApplicationWindow :



Graphe de collaboration de ApplicationWindow :

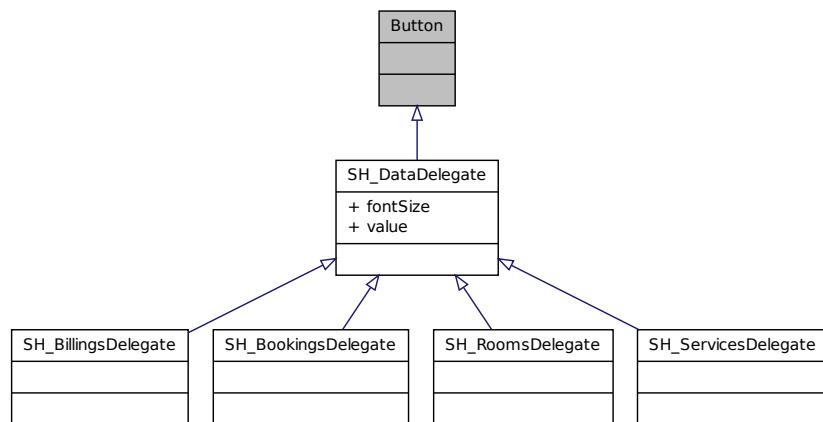


La documentation de cette classe a été générée à partir du fichier suivant :

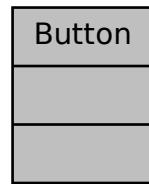
– [views/qml/SH_app.qml](#)

4.2 Référence de la classe Button

Graphe d'héritage de Button :



Graphe de collaboration de Button :

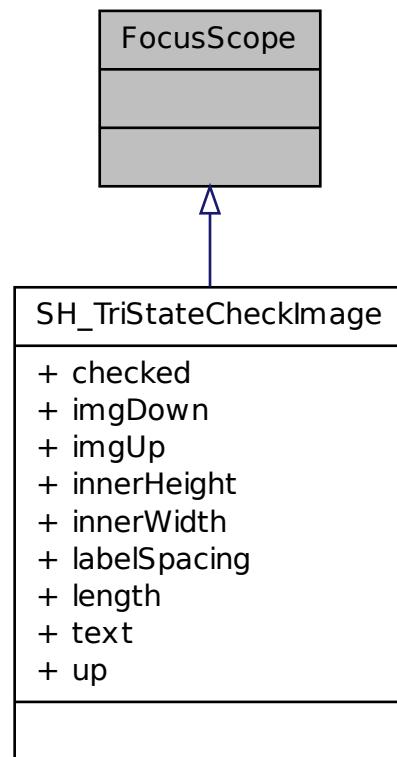


La documentation de cette classe a été générée à partir du fichier suivant :

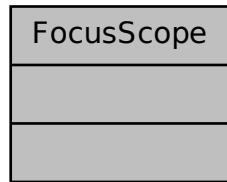
– [views/qml/SH_DataDelegate.qml](#)

4.3 Référence de la classe FocusScope

Graphe d'héritage de FocusScope :



Graphe de collaboration de FocusScope :

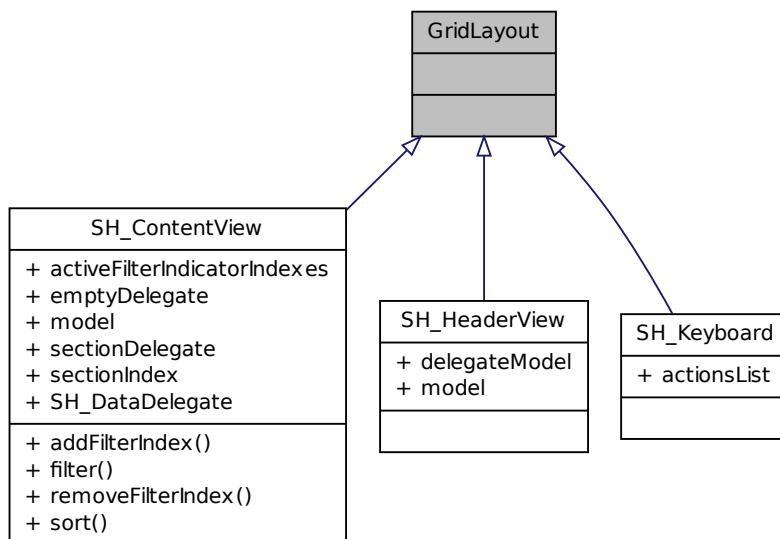


La documentation de cette classe a été générée à partir du fichier suivant :

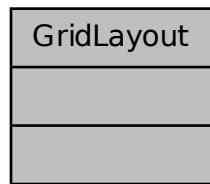
– [views/qml/SH_TriStateCheckImage.qml](#)

4.4 Référence de la classe GridLayout

Graphe d'héritage de GridLayout :



Graphe de collaboration de GridLayout :

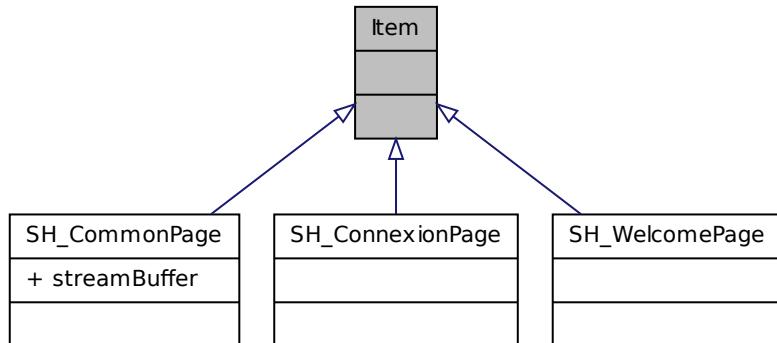


La documentation de cette classe a été générée à partir du fichier suivant :

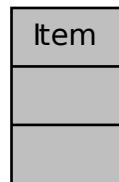
– views/qml/[SH_ContentView.qml](#)

4.5 Référence de la classe Item

Graphe d'héritage de Item :



Graphe de collaboration de Item :

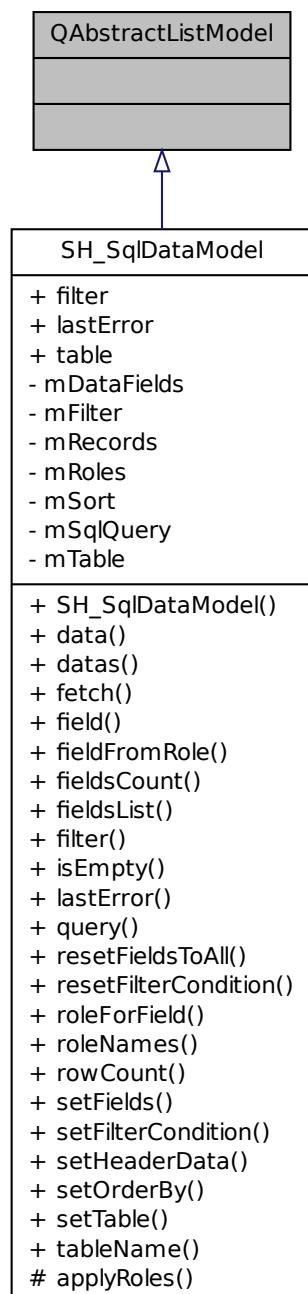


La documentation de cette classe a été générée à partir du fichier suivant :

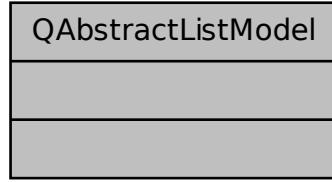
– views/qml/[SH_CommonPage.qml](#)

4.6 Référence de la classe QAbstractListModel

Graphe d'héritage de QAbstractListModel :



Graphe de collaboration de QAbstractListModel :

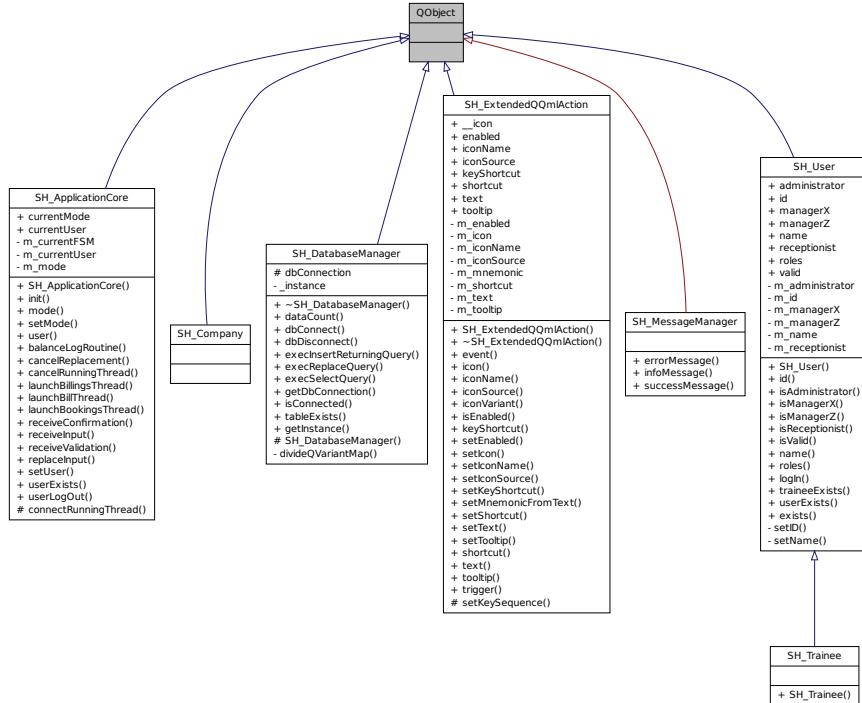


La documentation de cette classe a été générée à partir du fichier suivant :

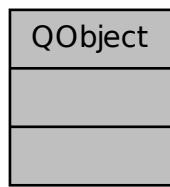
– [models/SH_SqlDataModel.h](#)

4.7 Référence de la classe QObject

Graphe d'héritage de QObject :



Graphe de collaboration de QObject :

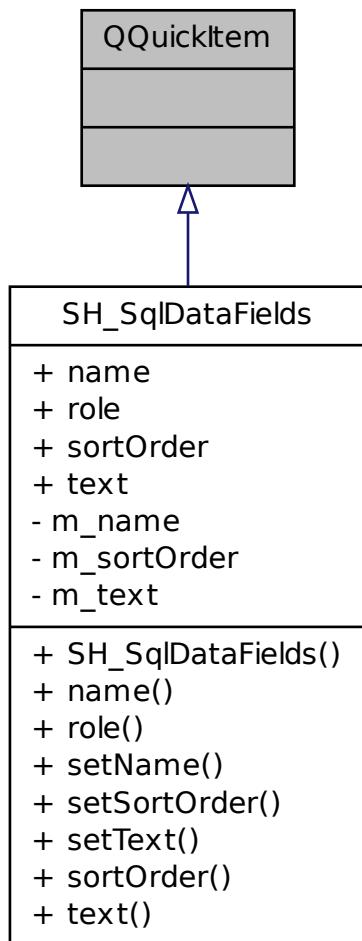


La documentation de cette classe a été générée à partir du fichier suivant :

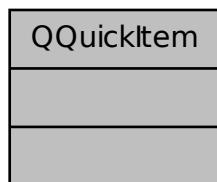
- [SH_MessageManager.h](#)

4.8 Référence de la classe QQuickItem

Graphe d'héritage de QQuickItem :



Graphe de collaboration de QQuickItem :

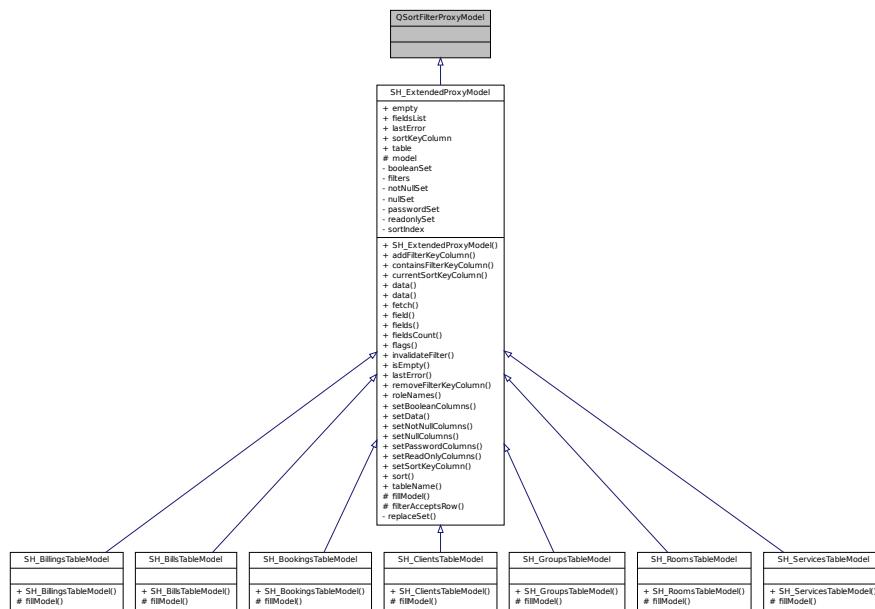


La documentation de cette classe a été générée à partir du fichier suivant :

- models/[SH_SqlDataField.h](#)

4.9 Référence de la classe QSortFilterProxyModel

Graphe d'héritage de QSortFilterProxyModel :



Graphe de collaboration de QSortFilterProxyModel :

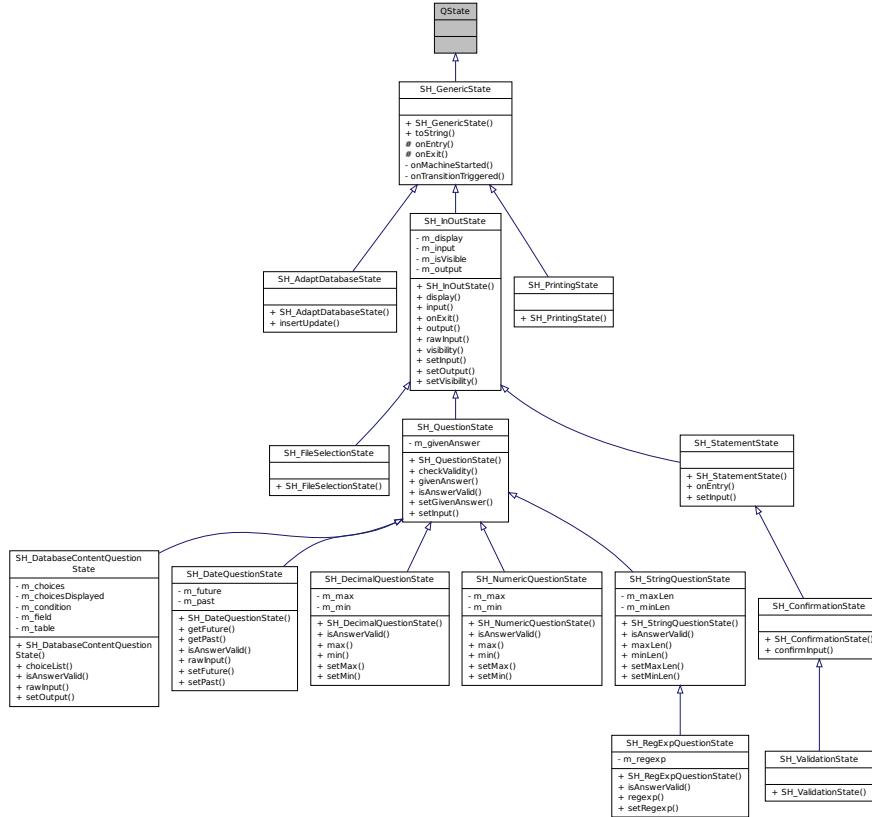


La documentation de cette classe a été générée à partir du fichier suivant :

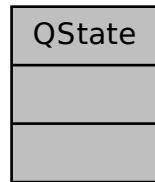
- models/[SH_ExtendedSqlProxyModel.h](#)

4.10 Référence de la classe QState

Graphe d'héritage de QState :



Graphe de collaboration de QState :

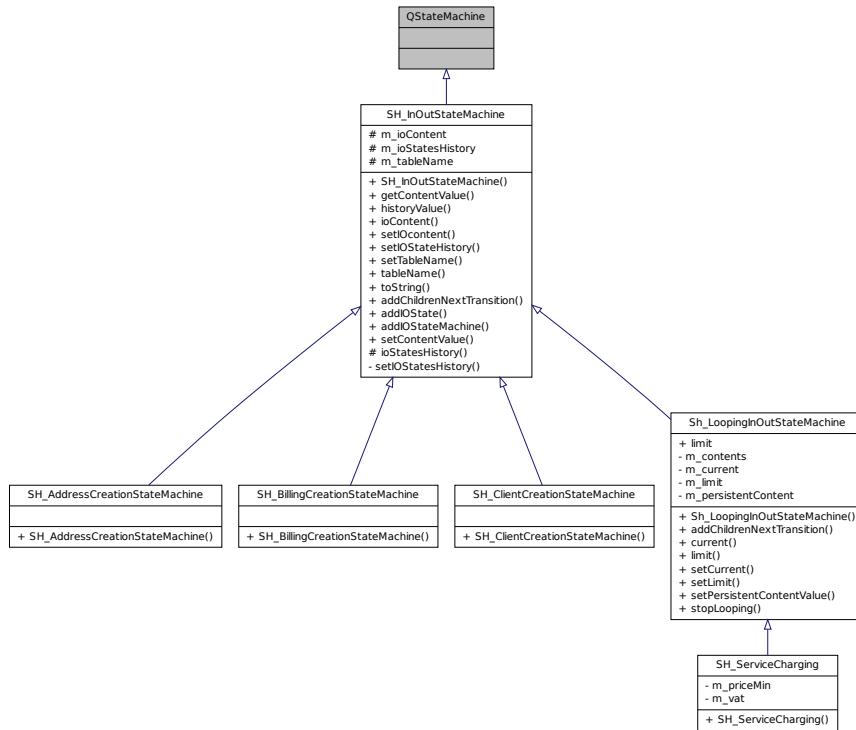


La documentation de cette classe a été générée à partir du fichier suivant :

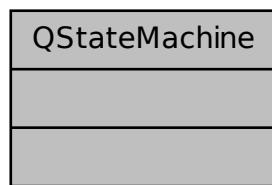
– logic/[SH_GenericDebugableState.h](#)

4.11 Référence de la classe QStateMachine

Graphe d'héritage de QStateMachine :



Graphe de collaboration de QStateMachine :

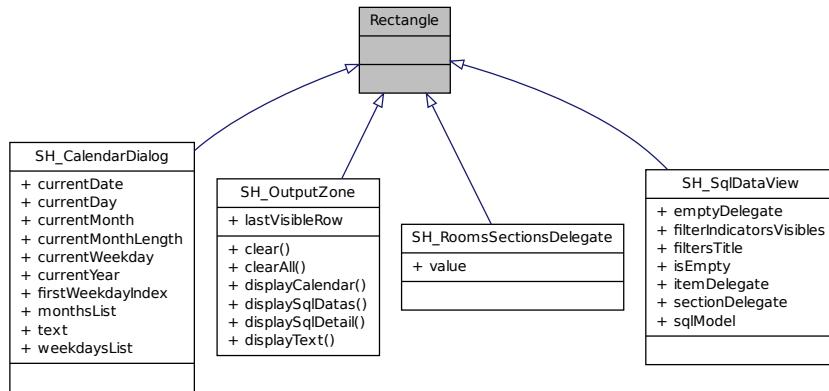


La documentation de cette classe a été générée à partir du fichier suivant :

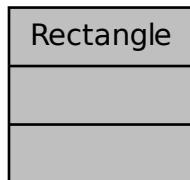
- logic/[SH_IOStateMachine.h](#)

4.12 Référence de la classe Rectangle

Graphe d'héritage de Rectangle :



Graphe de collaboration de Rectangle :



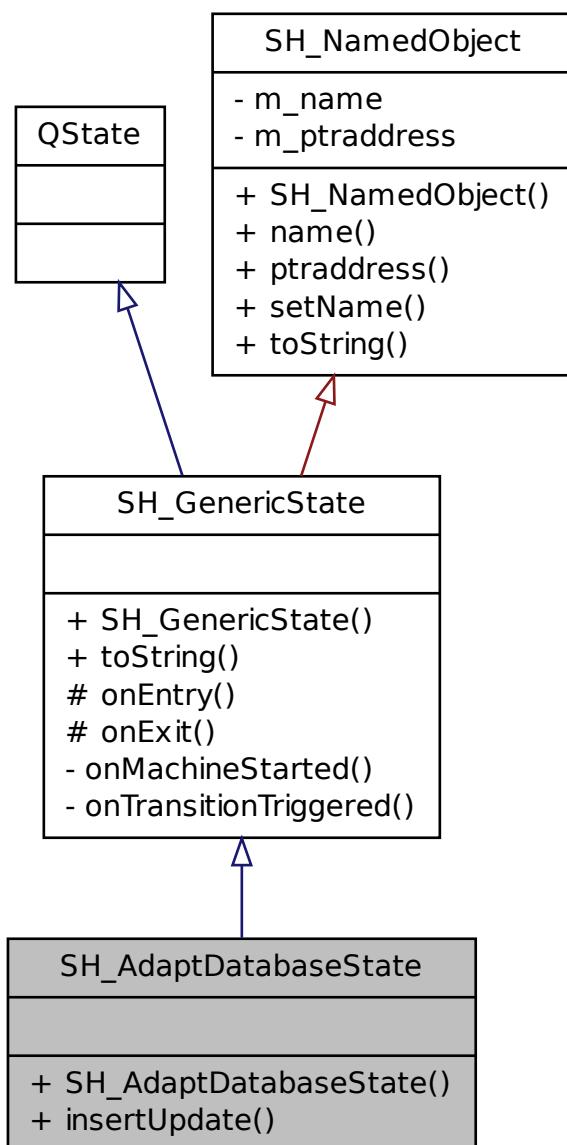
La documentation de cette classe a été générée à partir du fichier suivant :

– views/qml/[SH_CalendarDialog.qml](#)

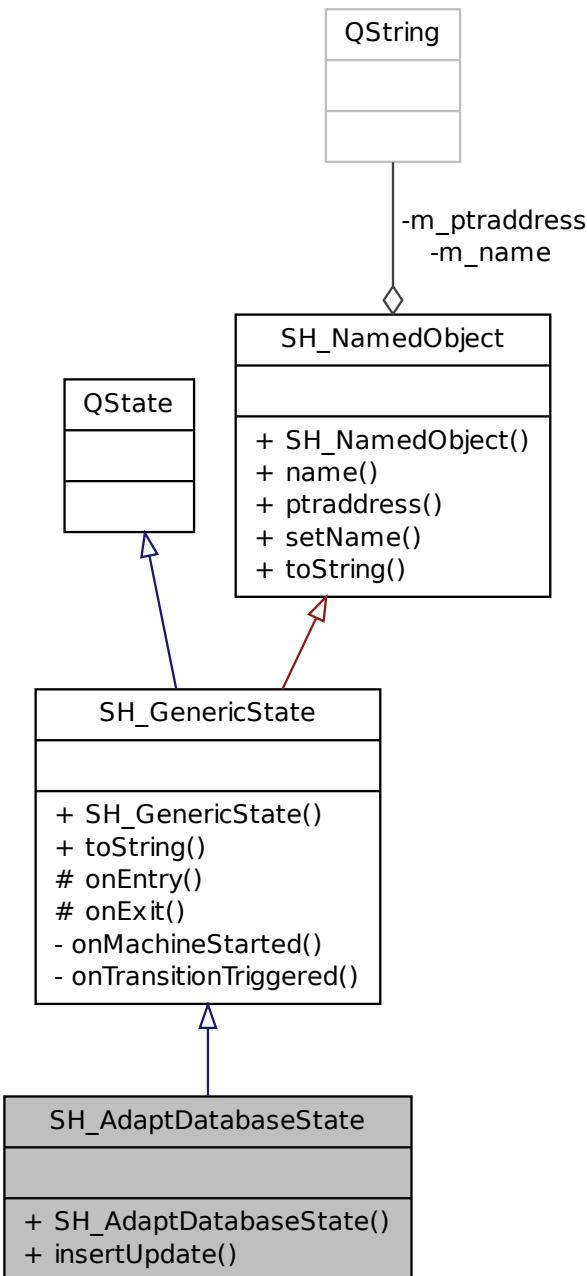
4.13 Référence de la classe SH_AdaptDatabaseState

```
#include <SH_AdaptDatabaseState.h>
```

Graphe d'héritage de SH_AdaptDatabaseState :



Graphe de collaboration de SH_AdaptDatabaseState :



Signaux

- void `next ()`

Fonctions membres publiques

- `SH_AdaptDatabaseState (QString name, QState *parent=0)`
- `QVariant insertUpdate (QString table, QVariantMap content)`

- `QString toString ()`

Fonctions membres protégées

- `void onEntry (QEvent *event)`
- `void onExit (QEvent *event)`

4.13.1 Description détaillée

Définition à la ligne 9 du fichier [SH_AdaptDatabaseState.h](#).

4.13.2 Documentation des constructeurs et destructeur

4.13.2.1 `SH_AdaptDatabaseState : :SH_AdaptDatabaseState (QString name, QState * parent = 0)`

Définition à la ligne 9 du fichier [SH_AdaptDatabaseState.cpp](#).

```
00009
00010     SH_GenericState(name, parent)
00011 {
00012 }
```

4.13.3 Documentation des fonctions membres

4.13.3.1 `QVariant SH_AdaptDatabaseState : :insertUpdate (QString table, QVariantMap content)`

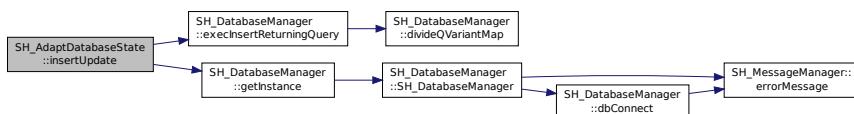
Définition à la ligne 18 du fichier [SH_AdaptDatabaseState.cpp](#).

Références `SH_DatabaseManager : :execInsertReturningQuery()`, `SH_DatabaseManager : :getInstance()`, et `SH_GenericState : :next()`.

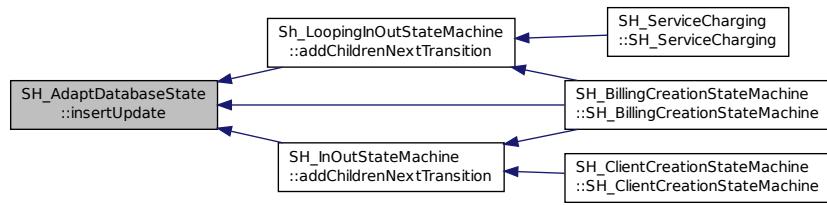
Référencé par `Sh_LoopingInOutStateMachine : :addChildsNextTransition()`, `SH_InOutStateMachine : :addChildsNextTransition()`, et `SH_BillingCreationStateMachine : :SH_BillingCreationStateMachine()`.

```
00019 {
00020     QVariant id = SH_DatabaseManager::getInstance () ->
00021         execInsertReturningQuery(table, content, "id");
00022     if(id.isValid ()) {
00023         emit next ();
00024     }
00025 }
```

Voici le graphe d'appel pour cette fonction :



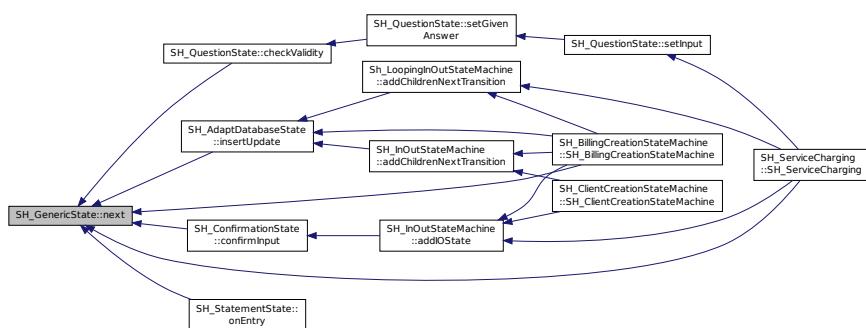
Voici le graphe des appelants de cette fonction :



4.13.3.2 void SH_GenericState ::next() [signal], [inherited]

Référencé par [SH_QuestionState ::checkValidity\(\)](#), [SH_ConfirmationState ::confirmInput\(\)](#), [insertUpdate\(\)](#), [SH_StatementState ::onEntry\(\)](#), [SH_BillingCreationStateMachine ::SH_BillingCreationStateMachine\(\)](#), et [SH_ServiceCharging ::SH_ServiceCharging\(\)](#).

Voici le graphe des appelants de cette fonction :



4.13.3.3 void SH_GenericState ::onEntry (QEvent * event) [protected], [inherited]

Définition à la ligne 62 du fichier [SH_GenericDebugableState.cpp](#).

Références [SH_NamedObject ::name\(\)](#).

Référencé par [SH_StatementState ::onEntry\(\)](#).

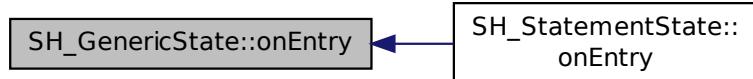
```

00063 {
00064     Q_UNUSED(event);
00065     qDebug() << "Machine: " << machine() ->objectName() << " entered " << name();
00066 }
  
```

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :



4.13.3.4 void SH_GenericState ::onExit (QEvent * event) [protected], [inherited]

Définition à la ligne 73 du fichier [SH_GenericDebugableState.cpp](#).

Références [SH_NamedObject ::name\(\)](#).

Référencé par [SH_InOutState ::onExit\(\)](#).

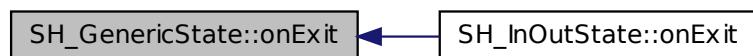
```

00074 {
00075     Q_UNUSED(event);
00076     qDebug() << "Machine: " << machine()->objectName() << " exited " << name();
00077 }
  
```

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :



4.13.3.5 QString SH_GenericState ::toString () [virtual], [inherited]

Réimplémentée à partir de [SH_NamedObject](#).

Définition à la ligne 21 du fichier [SH_GenericDebugableState.cpp](#).

Références [SH_NamedObject ::toString\(\)](#), et [SH_InOutStateMachine ::toString\(\)](#).

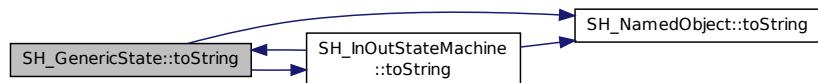
Référencé par [SH_InOutStateMachine ::addChildsNextTransition\(\)](#), [SH_DateQuestionState ::rawInput\(\)](#), et [SH_InOutStateMachine ::toString\(\)](#).

```

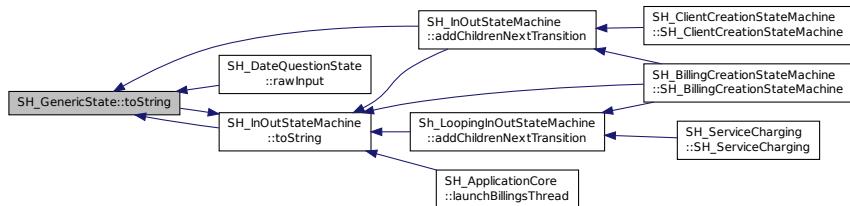
00022 {
00023     QStateMachine* machine = this->machine();
00024     SH_InOutStateMachine* mach = qobject_cast<
00025         SH_InOutStateMachine *>(machine);
00026     if(mach) {
00027         return SH_NamedObject::toString() + " [in "+mach->
00028             toString()+" ] ";
00029     } else {
00030         return SH_NamedObject::toString();
00031     }
00030 }

```

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appels de cette fonction :



La documentation de cette classe a été générée à partir des fichiers suivants :

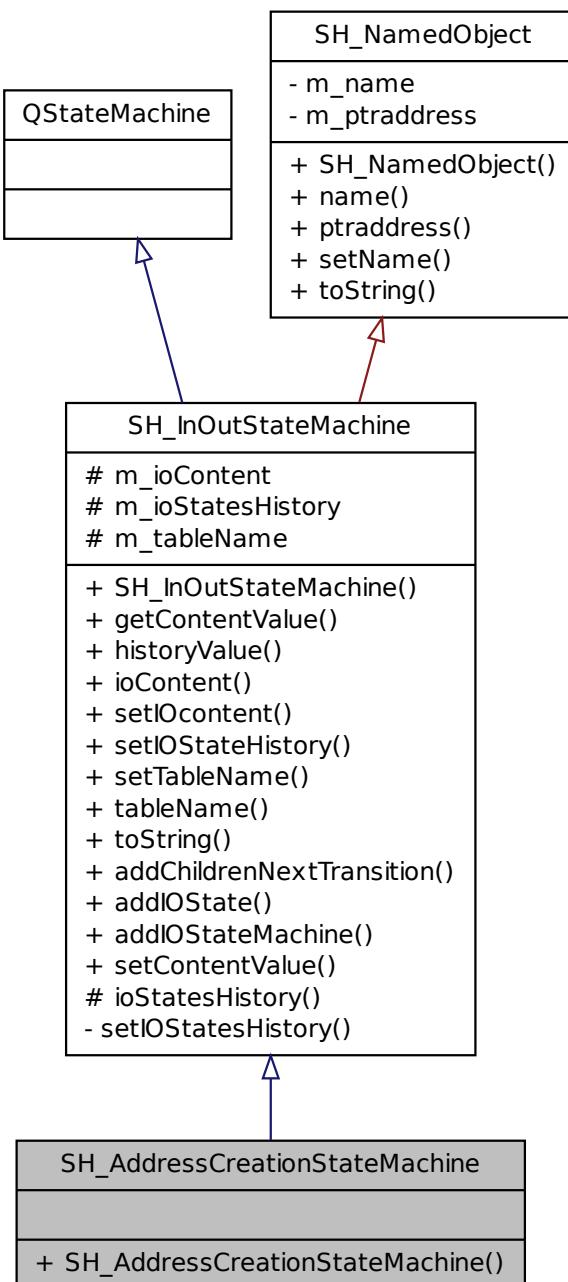
- logic/[SH_AdaptDatabaseState.h](#)
- logic/[SH_AdaptDatabaseState.cpp](#)

4.14 Référence de la classe SH_AddressCreationStateMachine

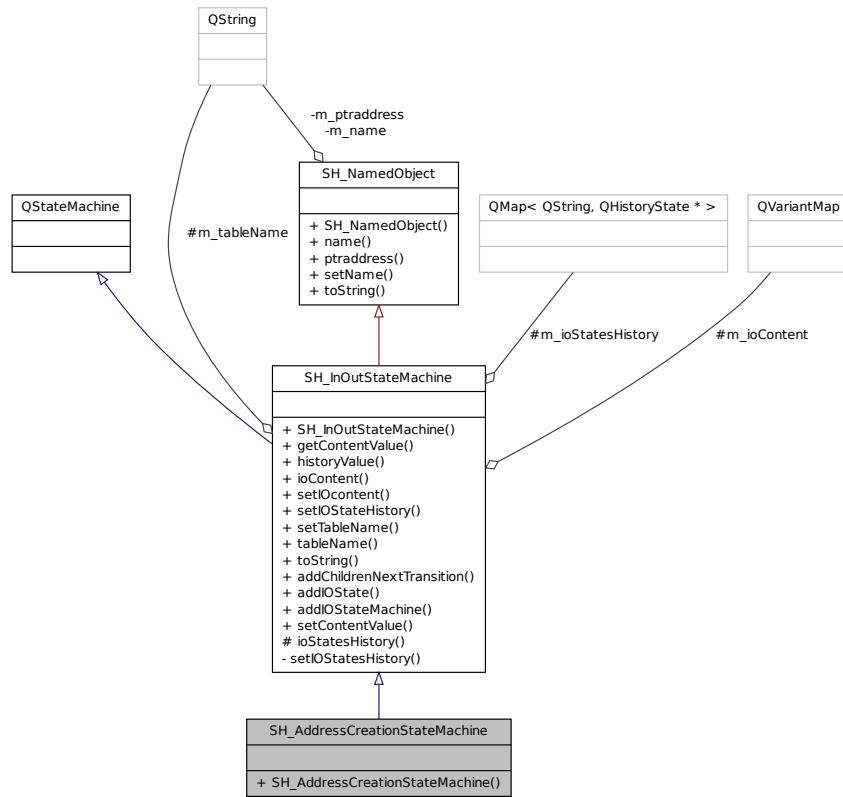
The [SH_AddressCreationStateMachine](#) class.

```
#include <SH_AddressCreation.h>
```

Graphe d'héritage de SH_AddressCreationStateMachine :



Graphe de collaboration de SH_AddressCreationStateMachine :



Connecteurs publics

- void `addChildsNextTransition` (QAbstractState *previousState, QAbstractState *nextState)
- void `addIOState` (SH_InOutState *state, QString field)
- void `addIOStateMachine` (SH_InOutStateMachine *fsm)
- void `setContentValue` (QVariant content, QString field)

Signaux

- void `cancelReplacement` ()
- void `clearAll` ()
- void `confirmInput` ()
- void `displayCalendar` ()
- void `displayFileDialog` ()
- void `next` ()
- void `receiveInput` (QString input)
- void `replaceInput` (QString field)
- void `resendText` (QString text, bool editable=false)
- void `sendText` (QString text, bool editable=false)
- void `validateInput` ()

Fonctions membres publiques

- `SH_AddressCreationStateMachine` (QString name, QObject *parent=0)
- QVariant `getContentValue` (QString field)
- QHistoryState * `historyValue` (QString field)
- QVariantMap `ioContent` () const
- void `setIOContent` (const QVariantMap &ioContent)
- void `setIOStateHistory` (QHistoryState *state, QString field)
- void `setTableName` (const QString &tableName)

- QString [tableName \(\) const](#)
- QString [toString \(\)](#)

Fonctions membres protégées

- QMap< QString, QHistoryState * > [ioStatesHistory \(\) const](#)

Attributs protégés

- QVariantMap [m_ioContent
 m_ioContent](#)
- QMap< QString, QHistoryState * > [m_ioStatesHistory
 m_ioStatesHistory](#)
- QString [m_tableName
 m_tableName](#)

4.14.1 Description détaillée

The [SH_AddressCreationStateMachine](#) class.

Définition à la ligne 7 du fichier [SH_AddressCreation.h](#).

4.14.2 Documentation des constructeurs et destructeur

4.14.2.1 SH_AddressCreationStateMachine : :SH_AddressCreationStateMachine (QString name, QObject * parent = 0)

Définition à la ligne 8 du fichier [SH_AddressCreation.cpp](#).

Références [SH_InOutStateMachine : :next\(\)](#).

```
00008
00009     SH_InOutStateMachine("ADRESSES", name, parent)
00010 {
00011 /*TODO: rue, numéro, code postal, ville, pays, destinataire*/
00012 emit next();
00013 }
```

4.14.3 Documentation des fonctions membres

4.14.3.1 void SH_InOutStateMachine : :addChildrenNextTransition (QAbstractState * previousState, QAbstractState * nextState) [slot], [inherited]

Définition à la ligne 250 du fichier [SH_IOStateMachine.cpp](#).

Références [SH_InOutStateMachine : :clearAll\(\)](#), [SH_InOutStateMachine : :historyValue\(\)](#), [SH_AdaptDatabaseState : :insertUpdate\(\)](#), [SH_InOutStateMachine : :m_ioContent](#), [SH_InOutStateMachine : :m_tableName](#), [SH_InOutStateMachine : :next\(\)](#), [SH_InOutStateMachine : :replaceInput\(\)](#), [SH_InOutStateMachine : :sendText\(\)](#), [SH_InOutStateMachine : :setContentValue\(\)](#), [SH_GenericState : :toString\(\)](#), et [SH_InOutStateMachine : :toString\(\)](#).

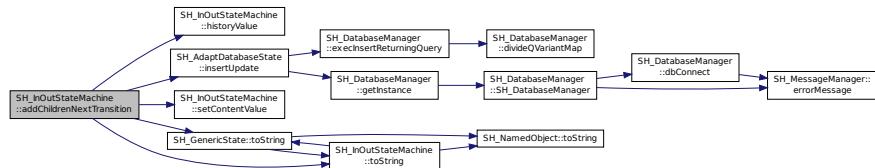
Référencé par [SH_BillingCreationStateMachine : :SH_BillingCreationStateMachine\(\)](#), et [SH_ClientCreationStateMachine : :SH_ClientCreationStateMachine\(\)](#).

```
00251 {
00252     SH_InOutStateMachine* fsmPreviousState = qobject_cast<
00253         SH_InOutStateMachine*>(previousState);
00254     SH_GenericState* genPreviousState = qobject_cast<
00255         SH_GenericState*>(previousState);
00256     QFinalState* final = qobject_cast<QFinalState*>(nextState);
00257     if(final) {
00258         SH_AdaptDatabaseState* saveState = new
00259             SH_AdaptDatabaseState("enregistrement de la machine "+
00260             toString());
00261         if(genPreviousState) {
```

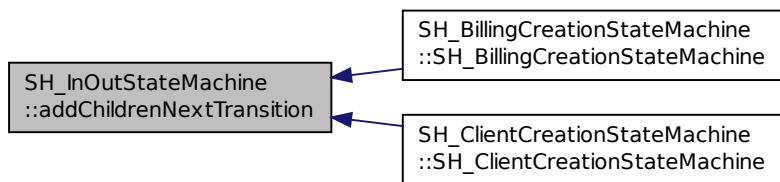
```

00258         genPreviousState->addTransition(genPreviousState, SIGNAL(next()), saveState);
00259     }
00260     if(fsmPreviousState) {
00261         fsmPreviousState->addTransition(fsmPreviousState, SIGNAL(next()), saveState);
00262     }
00263     if(genPreviousState || fsmPreviousState) {
00264         connect(previousState, &QAbstractState::exited, [=]() {
00265             connect(saveState, &QAbstractState::entered, [=]() {
00266                 emit this->sendText("Merci !");
00267                 setContentValue(saveState->insertUpdate(
00268                     m_tableName, m_ioContent), "ID");
00269                 emit this->clearAll();
00270             });
00271             saveState->addTransition(saveState, SIGNAL(next()), final);
00272         })
00273     } else {
00274         if(genPreviousState) {
00275             qDebug() << "next transition between " << genPreviousState->toString() << " and " <<
00276             nextState;
00277             genPreviousState->addTransition(genPreviousState, SIGNAL(next()), nextState);
00278         }
00279         if(fsmPreviousState) {
00280             qDebug() << "next transition between " << fsmPreviousState->toString() << " and " <<
00281             nextState;
00282             fsmPreviousState->addTransition(fsmPreviousState, SIGNAL(next()), nextState);
00283         }
00284         if(genPreviousState) {
00285             /*à faire au moment de l'entrée dans l'état previousState*/
00286             connect(genPreviousState, &QAbstractState::entered, [=]() {
00287                 connect(this, &SH_InOutStateMachine::replaceInput, [=](
00288                     QString field) {
00289                     /*après avoir demandé à revenir sur un état précédent, on attend la fin de l'état actuel
00290                     puis on retourne à l'historique de l'état désiré; celui-ci fini, on passe à l'état qui aurait du suivre celui
00291                     pendant lequel on a demandé à revenir sur un état précédent*/
00292                     QHistoryState* hState = historyValue(field);
00293                     if(hState) { /*si l'historique existe (on a déjà quitté l'état voulu)*/
00294                         hState->parentState()->addTransition(hState->parentState(), SIGNAL(
00295                             next()), nextState);
00296                         genPreviousState->addTransition(genPreviousState, SIGNAL(
00297                             next()), hState);
00298                     }
00299                 });
00300             });
00301         }
00302     }
00303 }
```

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appels de cette fonction :



4.14.3.2 void SH_InOutStateMachine ::addIOState (SH_InOutState * state, QString field) [slot], [inherited]

Définition à la ligne 110 du fichier [SH_IOStateMachine.cpp](#).

Références [SH_ConfirmationState ::confirmInput\(\)](#), [SH_InOutState ::display\(\)](#), [SH_InOutStateMachine ::displayCalendar\(\)](#), [SH_InOutStateMachine ::displayFileDialog\(\)](#), [SH_InOutState ::output\(\)](#), [SH_InOutState ::rawInput\(\)](#), [SH_InOutStateMachine ::receiveInput\(\)](#), [SH_InOutState ::resendInput\(\)](#), [SH_InOutStateMachine ::resendText\(\)](#), [SH_InOutState ::sendOutput\(\)](#), [SH_InOutStateMachine ::sendText\(\)](#), [SH_InOutStateMachine ::setContentValue\(\)](#), [SH_InOutState ::setInput\(\)](#), [SH_InOutStateMachine ::setIOStateHistory\(\)](#), [SH_InOutStateMachine ::validateInput\(\)](#), et [SH_InOutState ::visibility\(\)](#).

Référencé par [SH_BillingCreationStateMachine ::SH_BillingCreationStateMachine\(\)](#), [SH_ClientCreationStateMachine ::SH_ClientCreationStateMachine\(\)](#), et [SH_ServiceCharging ::SH_ServiceCharging\(\)](#).

```

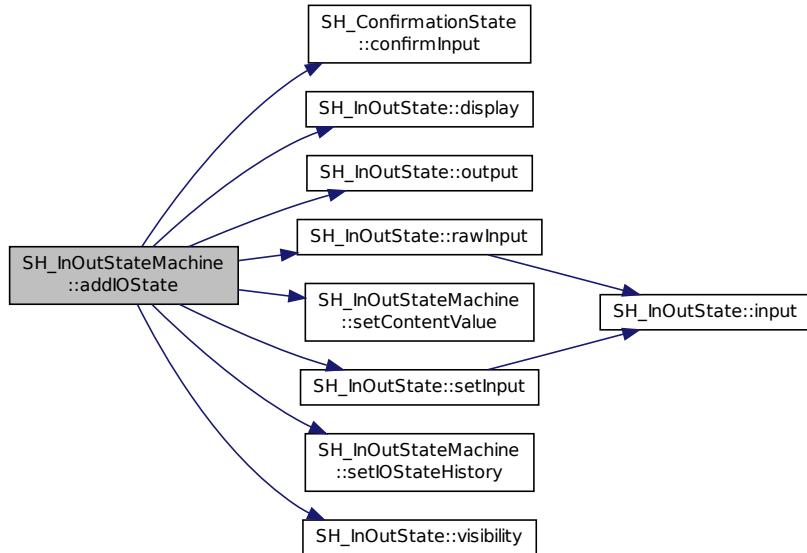
00111 {
00112     /*à faire au moment de l'entrée dans l'état state*/
00113     connect(state, &QState::entered, [=]() {
00114         qDebug() << "entered !";
00115         state->display(true);
00116         connect(this, &SH_InOutStateMachine::receiveInput, state, &
00117             SH_InOutState::setInput); /* la réception d'une valeur entraîne son enregistrement
00118             comme entrée de l'utilisateur auprès de l'état*/
00119         connect(this, &SH_InOutStateMachine::receiveInput, [=](QString in
00120             ) { qDebug() << "hello world !"; state->setInput(in);}); /* la réception d'une valeur entraîne son
00121             enregistrement comme entrée de l'utilisateur auprès de l'état*/
00122         connect(state, &SH_InOutState::sendOutput, [=](QVariant out) {qDebug() <<
00123             "connected !"; emit this->sendText(out.toString(), false);});
00124         connect(state, &SH_InOutState::resendInput, [=](QVariant in) {emit this->
00125             resendText(in.toString(), true);});
00126         if(state->visibility()) {
00127             state->sendOutput(QVariant(state->output()));
00128         } else {
00129             qDebug() << "invisible";
00130         }
00131     });
00132     SH_ValidationState *validationState = qobject_cast<
00133         SH_ValidationState*>(state);
00134     if(validationState) {
00135         /*à faire au moment de l'entrée dans l'état state*/
00136         connect(validationState, &QState::entered, [=]() {
00137             connect(this, &SH_InOutStateMachine::validateInput,
00138                 validationState, &SH_ValidationState::confirmInput);
00139         });
00140     }
00141     SH_ConfirmationState *confirmationState = qobject_cast<
00142         SH_ConfirmationState*>(state);
00143     if(confirmationState) {
00144         /*à faire au moment de l'entrée dans l'état state*/
00145         connect(confirmationState, &QState::entered, [=]() {
00146             connect(this, &SH_InOutStateMachine::validateInput,
00147                 confirmationState, &SH_ConfirmationState::confirmInput);
00148         });
00149     }
00150     /*à faire au moment de la sortie de l'état state*/
00151     connect(state, &QState::exited, [=]() {
00152         qDebug() << "exited !";
00153         if(!field.isEmpty()) {
00154             setContentValue(state->rawInput(), field);
00155             /*gestion de l'historique des états pour pouvoir revenir à l'état state après l'avoir quitté*/
00156             QHistoryState* hState = new QHistoryState(state);
00157             setIOStateHistory(hState, field);
00158         }
00159         state->disconnect(this); /*plus aucune action sur l'état ne pourra être provoquée par la machine*/
00160     });
00161
00162
00163     QAbstractState* astate = qobject_cast<QAbstractState *>(state);
00164     if(astate) {

```

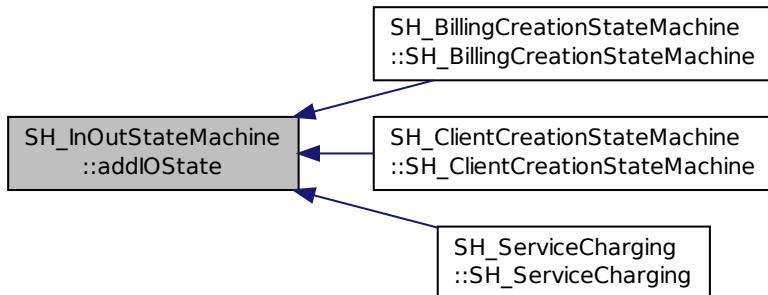
```

00165     addState(astate);
00166 }
00167 }
```

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :



4.14.3.3 void SH_InOutStateMachine ::addIOStateMachine (SH_InOutStateMachine * fsm) [slot], [inherited]

Définition à la ligne 175 du fichier [SH_IOStateMachine.cpp](#).

Références [SH_InOutStateMachine ::cancelReplacement\(\)](#), [SH_InOutStateMachine ::confirmInput\(\)](#), [SH_InOutStateMachine ::displayCalendar\(\)](#), [SH_InOutStateMachine ::receiveInput\(\)](#), [SH_InOutStateMachine ::replaceInput\(\)](#), [SH_InOutStateMachine ::resendText\(\)](#), [SH_InOutStateMachine ::sendText\(\)](#), et [SH_InOutStateMachine ::validateInput\(\)](#).

Référencé par [SH_BillingCreationStateMachine ::SH_BillingCreationStateMachine\(\)](#).

```

00176 {
00177     /*à faire au moment de l'entrée dans la machine d'état fsm*/
00178     connect(fsm, &QState::entered, [=]() {
00179         connect(this, &SH_InOutStateMachine::receiveInput, fsm, &
00180             SH_InOutStateMachine::receiveInput);
00181         connect(this, &SH_InOutStateMachine::sendText, fsm, &
00182             SH_InOutStateMachine::sendText);
00183         connect(this, &SH_InOutStateMachine::resendText, fsm, &
00184             SH_InOutStateMachine::resendText);
00185         connect(this, &SH_InOutStateMachine::confirmInput, fsm, &
00186             SH_InOutStateMachine::confirmInput);
00187         connect(this, &SH_InOutStateMachine::validateInput, fsm, &
00188             SH_InOutStateMachine::validateInput);
00189         connect(this, &SH_InOutStateMachine::replaceInput, fsm, &
00190             SH_InOutStateMachine::replaceInput);
00191         connect(this, &SH_InOutStateMachine::cancelReplacement, fsm,
00192             &SH_InOutStateMachine::cancelReplacement);
00193         connect(this, &SH_InOutStateMachine::displayCalendar, fsm, &
00194             SH_InOutStateMachine::displayCalendar);
00195     });
00196     /*à faire au moment de la sortie de la machine d'état fsm*/
00197     connect(fsm, &QState::exited, [=]() {
00198         fsm->disconnect(this); /*plus aucune action sur la machine d'état fille ne pourra être provoquée
00199         par la machine mère*/
00200     });
00201 };
00202 }
00203 }
```

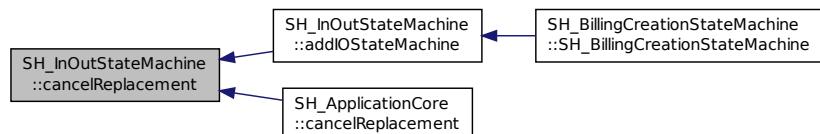
Voici le graphe des appelants de cette fonction :



4.14.3.4 void SH_InOutStateMachine ::cancelReplacement() [signal], [inherited]

Référencé par [SH_InOutStateMachine ::addIOStateMachine\(\)](#), et [SH_ApplicationCore ::cancelReplacement\(\)](#).

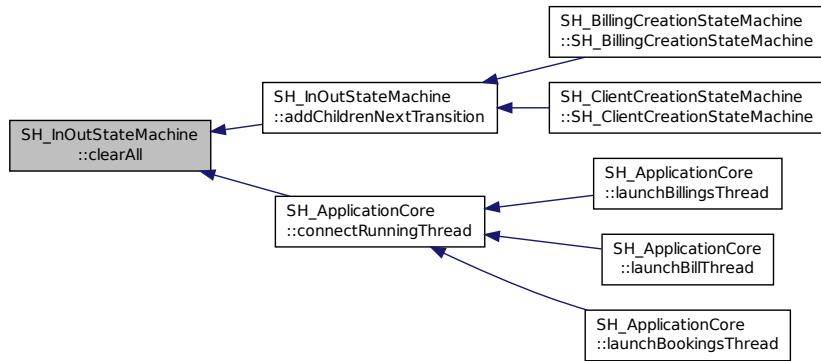
Voici le graphe des appelants de cette fonction :



4.14.3.5 void SH_InOutStateMachine ::clearAll() [signal], [inherited]

Référencé par [SH_InOutStateMachine ::addChildrenNextTransition\(\)](#), et [SH_ApplicationCore ::connectRunningThread\(\)](#).

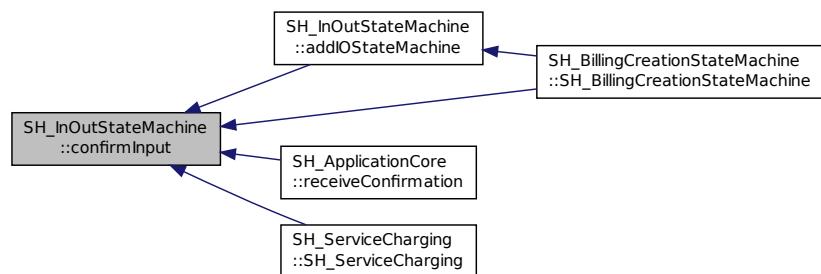
Voici le graphe des appelants de cette fonction :



4.14.3.6 void SH_InOutStateMachine ::confirmInput() [signal], [inherited]

Référencé par `SH_InOutStateMachine ::addIOStateMachine()`, `SH_ApplicationCore ::receiveConfirmation()`, `SH_BillingCreationStateMachine ::SH_BillingCreationStateMachine()`, et `SH_ServiceCharging ::SH_ServiceCharging()`.

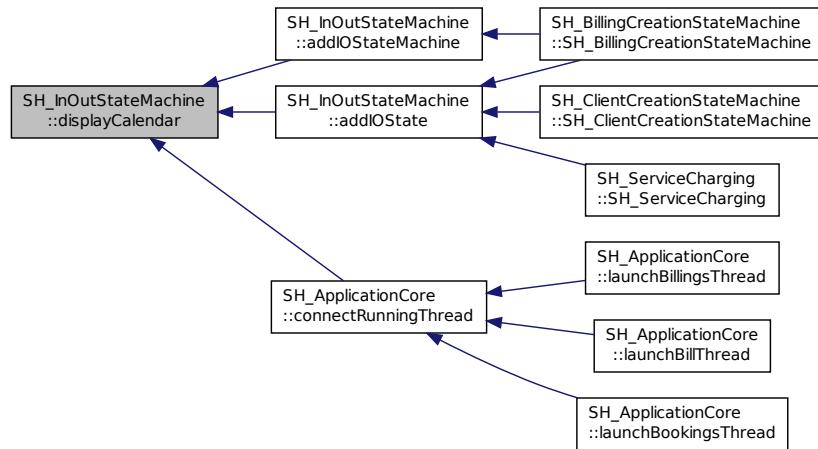
Voici le graphe des appelants de cette fonction :



4.14.3.7 void SH_InOutStateMachine ::displayCalendar() [signal], [inherited]

Référencé par `SH_InOutStateMachine ::addIOState()`, `SH_InOutStateMachine ::addIOStateMachine()`, et `SH_ApplicationCore ::connectRunningThread()`.

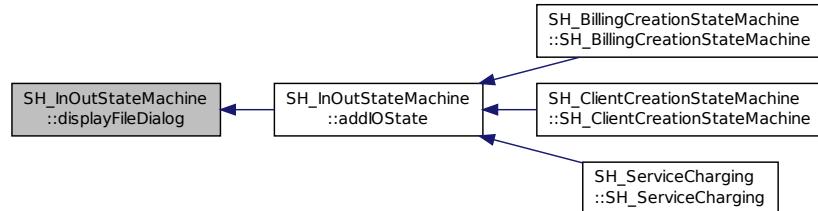
Voici le graphe des appelants de cette fonction :



4.14.3.8 void SH_InOutStateMachine ::displayFileDialog () [signal], [inherited]

Référencé par [SH_InOutStateMachine ::addIOState\(\)](#).

Voici le graphe des appelants de cette fonction :



4.14.3.9 QVariant SH_InOutStateMachine ::getContentValue (QString field) [inherited]

Définition à la ligne 65 du fichier [SH_IOStateMachine.cpp](#).

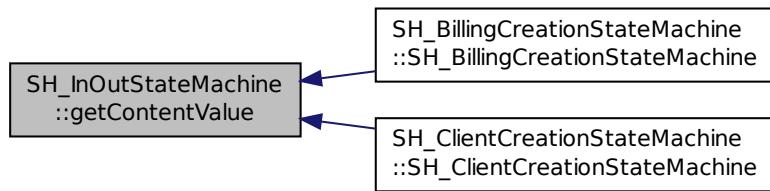
Références [SH_InOutStateMachine ::m_ioContent](#).

Référencé par [SH_BillingCreationStateMachine ::SH_BillingCreationStateMachine\(\)](#), et [SH_ClientCreationStateMachine ::SH_ClientCreationStateMachine\(\)](#).

```

00066 {
00067     return m_ioContent.value(field);
00068 }
  
```

Voici le graphe des appelants de cette fonction :



4.14.3.10 QHistoryState * SH_InOutStateMachine ::historyValue (QString field) [inherited]

Définition à la ligne 238 du fichier [SH_IOStateMachine.cpp](#).

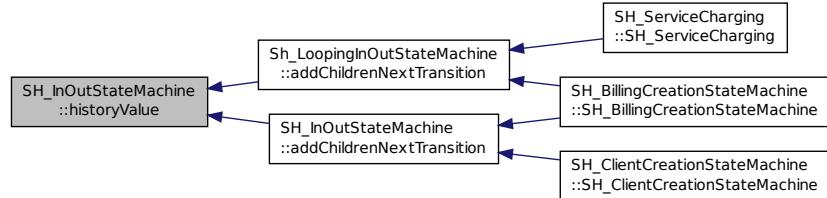
Références `SH_InOutStateMachine ::m_ioStatesHistory`.

Référencé par `Sh_LoopingInOutStateMachine ::addChildrenNextTransition()`, et `SH_InOutStateMachine ::addChildrenNextTransition()`.

```

00239 {
00240     return m_ioStatesHistory.value(field);
00241 }
  
```

Voici le graphe des appelants de cette fonction :



4.14.3.11 QVariantMap SH_InOutStateMachine ::ioContent () const [inherited]

Définition à la ligne 43 du fichier [SH_IOStateMachine.cpp](#).

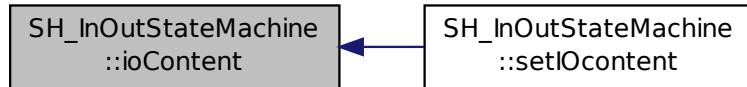
Références `SH_InOutStateMachine ::m_ioContent`.

Référencé par `SH_InOutStateMachine ::setIOcontent()`.

```

00044 {
00045     return m_ioContent;
00046 }
  
```

Voici le graphe des appelants de cette fonction :



4.14.3.12 QMap< QString, QHistoryState * > SH_InOutStateMachine ::ioStatesHistory() const [protected], [inherited]

Définition à la ligne 202 du fichier [SH_IOStateMachine.cpp](#).

Références [SH_InOutStateMachine ::m_ioStatesHistory](#).

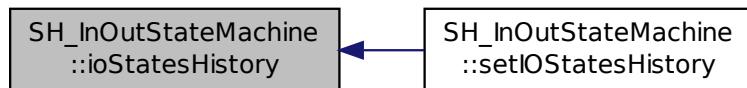
Référencé par [SH_InOutStateMachine ::setIOStatesHistory\(\)](#).

```

00203 {
00204     return m_ioStatesHistory;
00205 }

```

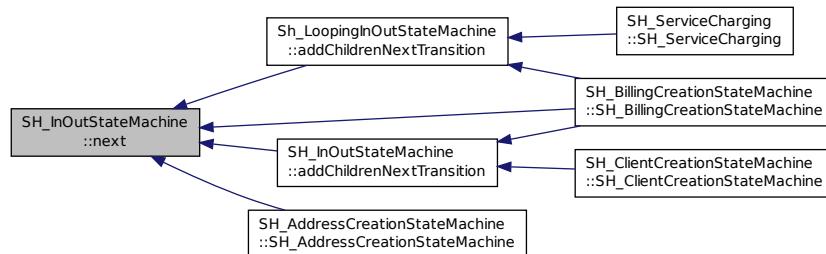
Voici le graphe des appelants de cette fonction :



4.14.3.13 void SH_InOutStateMachine ::next() [signal], [inherited]

Référencé par [Sh_LoopingInOutStateMachine ::addChildsNextTransition\(\)](#), [SH_InOutStateMachine ::addChildsNextTransition\(\)](#), [SH_AddressCreationStateMachine\(\)](#), et [SH_BillingCreationStateMachine ::SH_BillingCreationStateMachine\(\)](#).

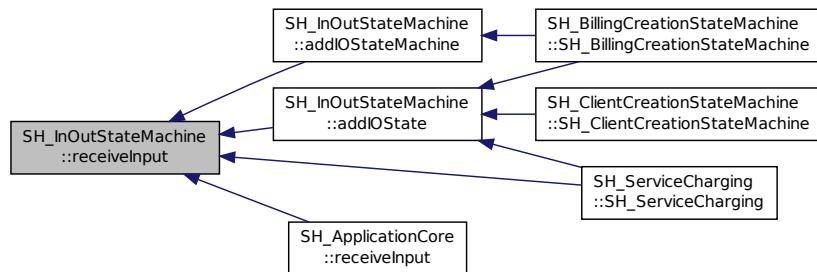
Voici le graphe des appelants de cette fonction :



4.14.3.14 void SH_InOutStateMachine ::receiveInput (QString *input*) [signal], [inherited]

Référencé par [SH_InOutStateMachine ::addIOState\(\)](#), [SH_InOutStateMachine ::addIOStateMachine\(\)](#), [SH_ApplicationCore ::receiveInput\(\)](#), et [SH_ServiceCharging ::SH_ServiceCharging\(\)](#).

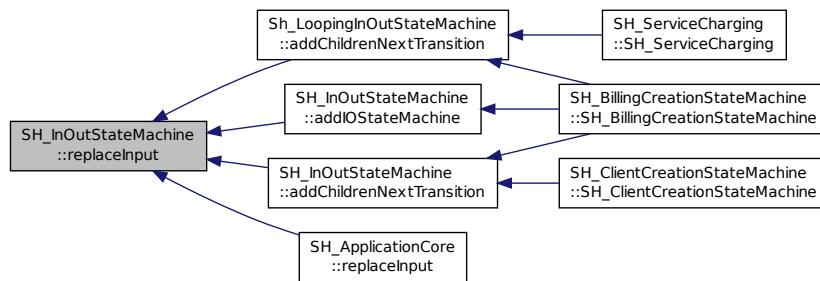
Voici le graphe des appelleurs de cette fonction :



4.14.3.15 void SH_InOutStateMachine ::replaceInput (QString *field*) [signal], [inherited]

Référencé par [Sh_LoopingInOutStateMachine ::addChildsNextTransition\(\)](#), [SH_InOutStateMachine ::addChildrenNextTransition\(\)](#), [SH_InOutStateMachine ::addIOStateMachine\(\)](#), et [SH_ApplicationCore ::replaceInput\(\)](#).

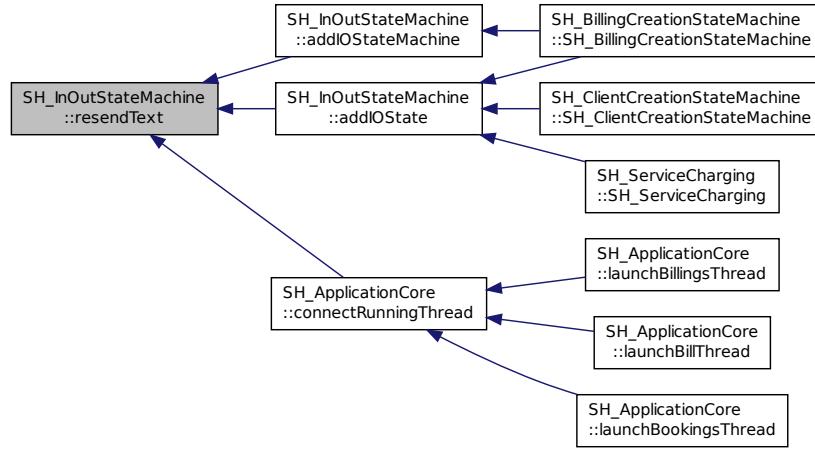
Voici le graphe des appelleurs de cette fonction :



4.14.3.16 void SH_InOutStateMachine ::resendText (QString *text*, bool *editable* = false) [signal], [inherited]

Référencé par [SH_InOutStateMachine ::addIOState\(\)](#), [SH_InOutStateMachine ::addIOStateMachine\(\)](#), et [SH_ApplicationCore ::connectRunningThread\(\)](#).

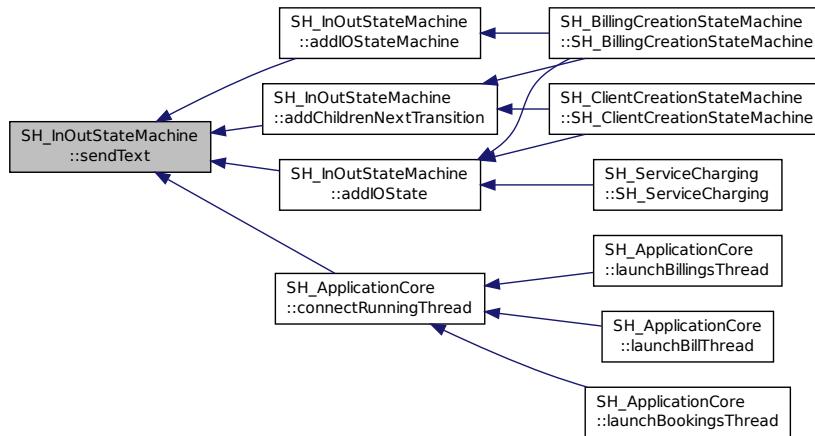
Voici le graphe des appelants de cette fonction :



4.14.3.17 void SH_InOutStateMachine ::sendText (QString text, bool editable = false) [signal], [inherited]

Référencé par [SH_InOutStateMachine ::addChildrenNextTransition\(\)](#), [SH_InOutStateMachine ::addIOState\(\)](#), [SH_InOutStateMachine ::addIOStateMachine\(\)](#), et [SH_ApplicationCore ::connectRunningThread\(\)](#).

Voici le graphe des appelants de cette fonction :



4.14.3.18 void SH_InOutStateMachine ::setContentValue (QVariant content, QString field) [slot], [inherited]

Définition à la ligne 99 du fichier [SH_IOSMachine.cpp](#).

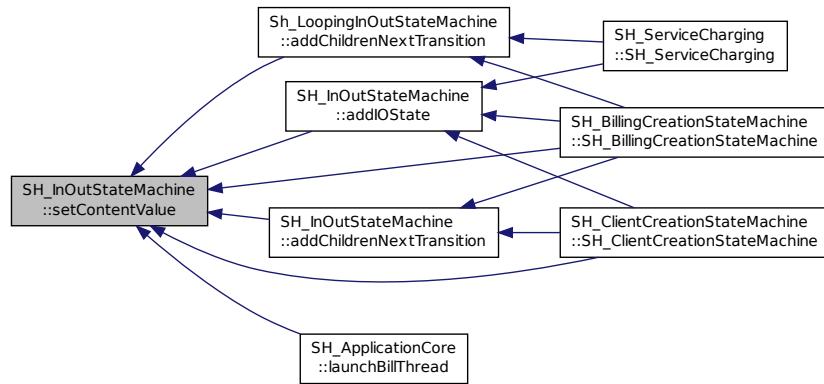
Références [SH_InOutStateMachine ::m_ioContent](#).

Référencé par [Sh_LoopingInOutStateMachine ::addChildrenNextTransition\(\)](#), [SH_InOutStateMachine ::addChildrenNextTransition\(\)](#), [SH_InOutStateMachine ::addIOState\(\)](#), [SH_ApplicationCore ::launchBillThread\(\)](#),

[SH_BillingCreationStateMachine](#) : `:SH_BillingCreationStateMachine()`, et [SH_ClientCreationStateMachine](#) : `:SH_ClientCreationStateMachine()`.

```
00100 {
00101     m_ioContent.insert(field, content);
00102 }
```

Voici le graphe des appels de cette fonction :



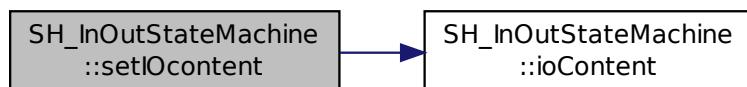
4.14.3.19 void SH_InOutStateMachine::setIOContent (const QVariantMap & ioContent) [inherited]

Définition à la ligne 54 du fichier [SH_IOSMachine.cpp](#).

Références [SH_InOutStateMachine::ioContent\(\)](#), et [SH_InOutStateMachine::m_ioContent](#).

```
00055 {
00056     m_ioContent = ioContent;
00057 }
```

Voici le graphe d'appel pour cette fonction :



4.14.3.20 void SH_InOutStateMachine::setIOStateHistory (QHistoryState * state, QString field) [inherited]

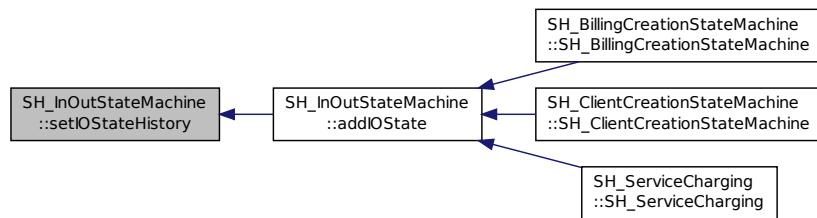
Définition à la ligne 226 du fichier [SH_IOSMachine.cpp](#).

Références [SH_InOutStateMachine::m_ioStatesHistory](#).

Référencé par [SH_InOutStateMachine::addIOState\(\)](#).

```
00227 {
00228     m_ioStatesHistory.insert(field, state); /*remplacement si plusieurs fois*/
00229 }
```

Voici le graphe des appels de cette fonction :



4.14.3.21 void SH_InOutStateMachine ::setTableName (const QString & tableName) [inherited]

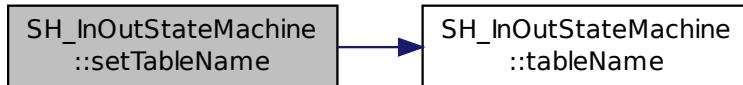
Définition à la ligne 87 du fichier [SH_IOStateMachine.cpp](#).

Références [SH_InOutStateMachine ::m_tableName](#), et [SH_InOutStateMachine ::tableName\(\)](#).

```

00088 {
00089     m_tableName = tableName;
00090 }
```

Voici le graphe d'appel pour cette fonction :



4.14.3.22 QString SH_InOutStateMachine ::tableName () const [inherited]

Définition à la ligne 76 du fichier [SH_IOStateMachine.cpp](#).

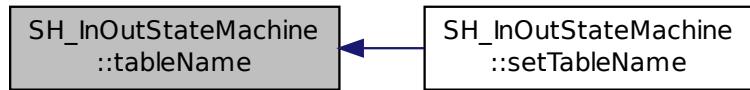
Références [SH_InOutStateMachine ::m_tableName](#).

Référencé par [SH_InOutStateMachine ::setTableName\(\)](#).

```

00077 {
00078     return m_tableName;
00079 }
```

Voici le graphe des appelants de cette fonction :



4.14.3.23 QString SH_InOutStateMachine ::toString() [virtual], [inherited]

Réimplémentée à partir de [SH_NamedObject](#).

Définition à la ligne 26 du fichier [SH_IOSMachine.cpp](#).

Références [SH_NamedObject ::toString\(\)](#), et [SH_GenericState ::toString\(\)](#).

Référencé par [Sh_LoopingInOutStateMachine ::addChildsNextTransition\(\)](#), [SH_InOutStateMachine ::addChildsNextTransition\(\)](#), [SH_ApplicationCore ::launchBillingsThread\(\)](#), [SH_BillingCreationStateMachine ::SH_BillingCreationStateMachine\(\)](#), et [SH_GenericState ::toString\(\)](#).

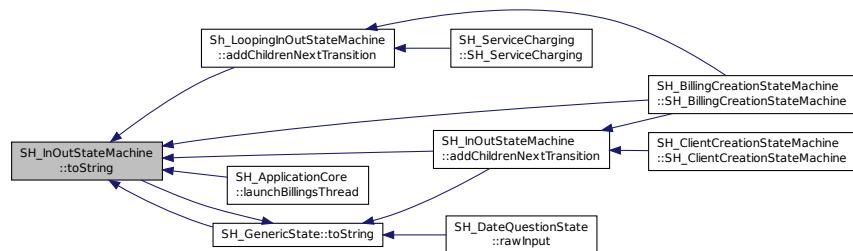
```

00027 {
00028     QObject* parent = this->parent();
00029     SH_GenericState* par = qobject_cast<SH_GenericState *>(parent);
00030     if(par) {
00031         return SH_NamedObject::toString() + " [descending from "+par->
00032             toString()+"] ";
00033     } else {
00034         return SH_NamedObject::toString();
00035     }
  
```

Voici le graphe d'appel pour cette fonction :



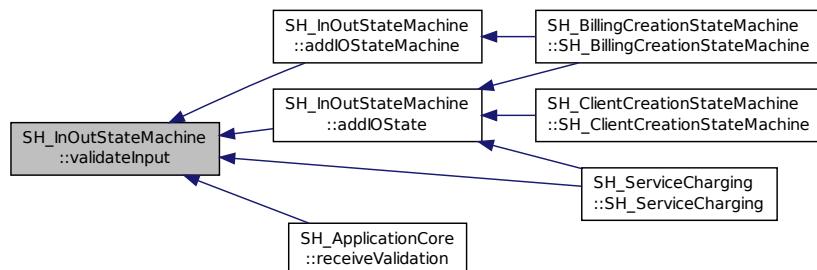
Voici le graphe des appelants de cette fonction :



4.14.3.24 void SH_InOutStateMachine ::validateInput() [signal], [inherited]

Référencé par [SH_InOutStateMachine ::addIOState\(\)](#), [SH_InOutStateMachine ::addIOStateMachine\(\)](#), [SH_ApplicationCore ::receiveValidation\(\)](#), et [SH_ServiceCharging ::SH_ServiceCharging\(\)](#).

Voici le graphe des appelleurs de cette fonction :



4.14.4 Documentation des données membres

4.14.4.1 QVariantMap SH_InOutStateMachine ::m_ioContent [protected], [inherited]

`m_ioContent`

Définition à la ligne 209 du fichier [SH_IOStateMachine.h](#).

Référencé par [Sh_LoopingInOutStateMachine ::addChildsNextTransition\(\)](#), [SH_InOutStateMachine ::addChildrenNextTransition\(\)](#), [SH_InOutStateMachine ::getContentValue\(\)](#), [SH_InOutStateMachine :::ioContent\(\)](#), [SH_InOutStateMachine ::setContentValue\(\)](#), [SH_InOutStateMachine ::setIOcontent\(\)](#), et [SH_BillingCreationStateMachine ::SH_BillingCreationStateMachine\(\)](#).

4.14.4.2 QMap<QString, QHistoryState*> SH_InOutStateMachine ::m_ioStatesHistory [protected], [inherited]

`m_ioStatesHistory`

Définition à la ligne 217 du fichier [SH_IOStateMachine.h](#).

Référencé par [SH_InOutStateMachine ::historyValue\(\)](#), [SH_InOutStateMachine :::ioStatesHistory\(\)](#), [SH_InOutStateMachine ::setIOStateHistory\(\)](#), et [SH_InOutStateMachine ::setIOStatesHistory\(\)](#).

4.14.4.3 QString SH_InOutStateMachine ::m_tableName [protected], [inherited]

`m_tableName`

Définition à la ligne 213 du fichier [SH_IOStateMachine.h](#).

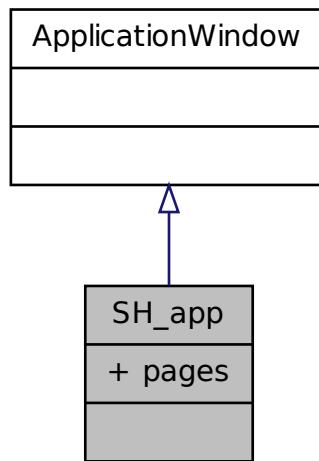
Référencé par [Sh_LoopingInOutStateMachine ::addChildsNextTransition\(\)](#), [SH_InOutStateMachine ::addChildrenNextTransition\(\)](#), [SH_InOutStateMachine :::setTableName\(\)](#), [SH_BillingCreationStateMachine ::SH_BillingCreationStateMachine\(\)](#), et [SH_InOutStateMachine ::tableName\(\)](#).

La documentation de cette classe a été générée à partir des fichiers suivants :

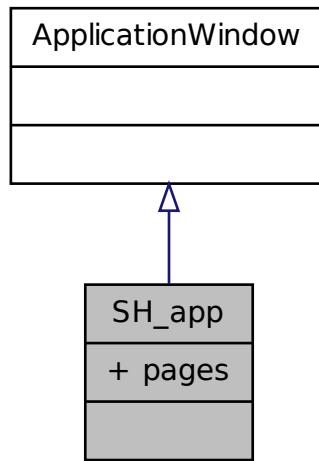
- logic/[SH_AddressCreation.h](#)
- logic/[SH_AddressCreation.cpp](#)

4.15 Référence de la classe SH_app

Graphe d'héritage de SH_app :



Graphe de collaboration de SH_app :



Signaux

- void `reload ()`

Propriétés

- alias [pages](#)

4.15.1 Description détaillée

Définition à la ligne [5](#) du fichier [SH_app.qml](#).

4.15.2 Documentation des fonctions membres

4.15.2.1 void SH_app ::reload() [signal]

4.15.3 Documentation des propriétés

4.15.3.1 alias SH_app ::pages

Définition à la ligne [8](#) du fichier [SH_app.qml](#).

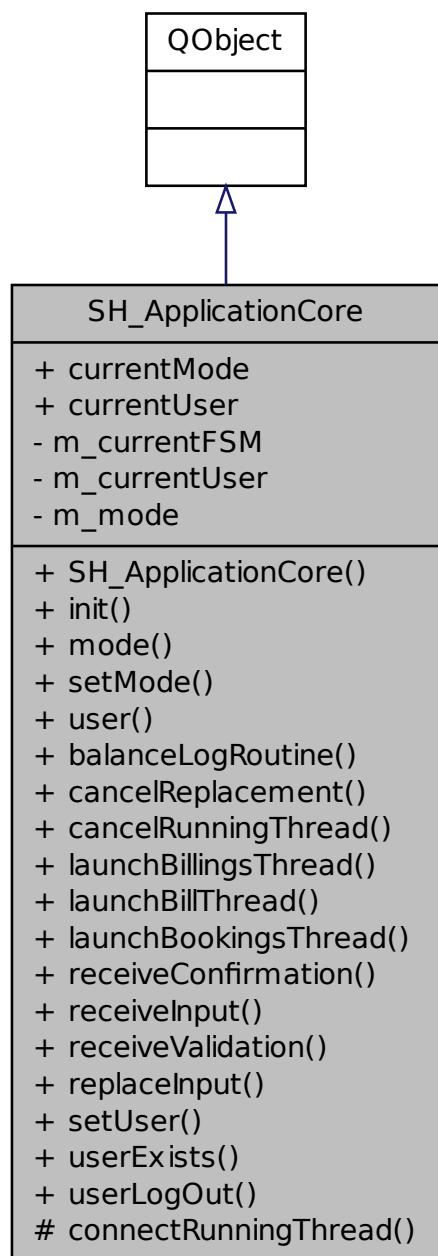
La documentation de cette classe a été générée à partir du fichier suivant :

- [views/qml/SH_app.qml](#)

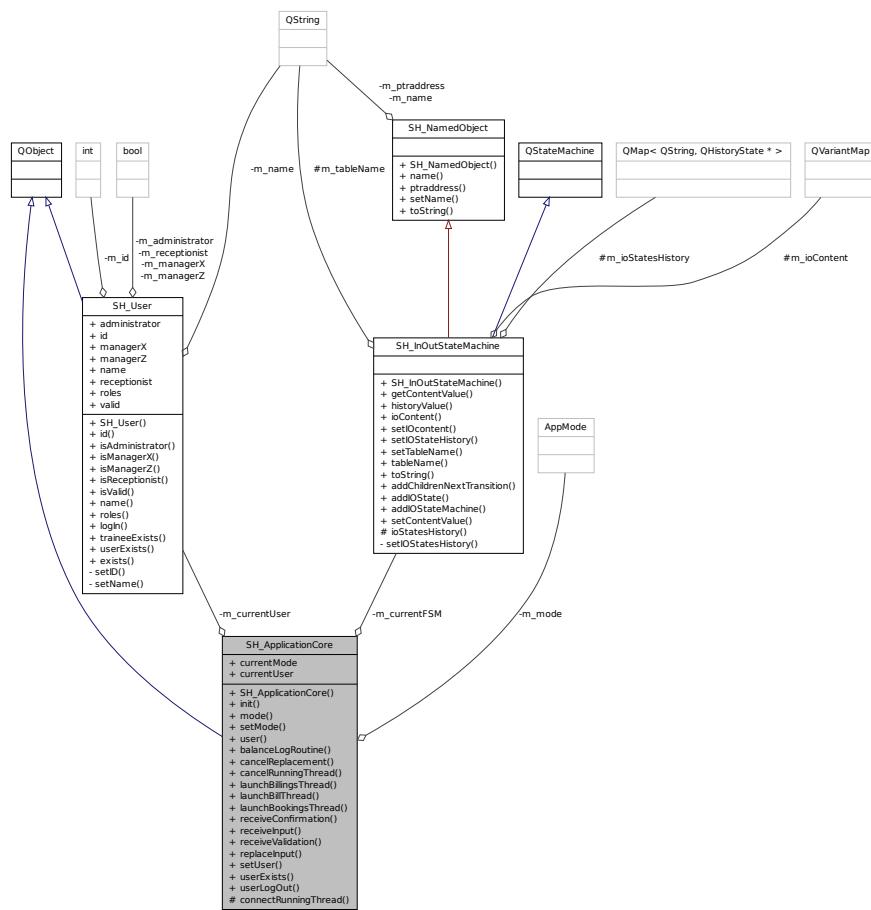
4.16 Référence de la classe SH_ApplicationCore

```
#include <SH_ApplicationCore.h>
```

Graphe d'héritage de SH_ApplicationCore :



Graphe de collaboration de SH_ApplicationCore :



Types publics

- enum [AppMode](#) {
 [CONNEXION](#), [ACCUEIL](#), [RECEPTION](#), [MANAGEMENT_X](#),
[MANAGEMENT_Z](#), [ADMINISTRATION](#) }

Connecteurs publics

- bool [balanceLogRoutine\(\)](#)
- void [cancelReplacement\(\)](#)
- bool [cancelRunningThread\(\)](#)
- Q_INVOKABLE bool [launchBillingsThread\(\)](#)
- bool [launchBillThread\(\)](#)
- bool [launchBookingsThread\(\)](#)
- void [receiveConfirmation\(\)](#)
- void [receiveInput\(QString\)](#)
- void [receiveValidation\(\)](#)
- void [replaceInput\(QString\)](#)
- bool [setUser\(QString, QString\)](#)
- bool [userExists\(QString\)](#)
- bool [userLogOut\(\)](#)

Signaux

- void [clearAll\(\)](#)
- void [currentFSMchanged\(\)](#)

- void [displayCalendar \(\)](#)
- void [modeChanged \(QVariant mode\)](#)
- void [openTab \(QVariant tabPos\)](#)
- void [resendText \(QString text\)](#)
- void [sendText \(QString text\)](#)
- void [userChanged \(QVariant name\)](#)

Fonctions membres publiques

- [SH_ApplicationCore \(QObject *parent=0\)](#)
- void [init \(\)](#)
- [AppMode mode \(\) const](#)
- void [setMode \(AppMode mode\)](#)
- [SH_User * user \(\) const](#)

Fonctions membres protégées

- bool [connectRunningThread \(\)](#)

Propriétés

- [AppMode currentMode](#)
- [SH_User currentUser](#)

Attributs privés

- [SH_InOutStateMachine * m_currentFSM
m_currentFSM](#)
- [SH_User * m_currentUser
m_currentUser](#)
- [AppMode m_mode
m_mode](#)

4.16.1 Description détaillée

Définition à la ligne 11 du fichier [SH_ApplicationCore.h](#).

4.16.2 Documentation des énumérations membres

4.16.2.1 enum SH_ApplicationCore : :AppMode

Valeurs énumérées

CONNEXION

ACCUEIL

RECEPTION

MANAGEMENT_X

MANAGEMENT_Z

ADMINISTRATION

Définition à la ligne 25 du fichier [SH_ApplicationCore.h](#).

```
00025 { CONNEXION, ACCUEIL, RECEPTION, MANAGEMENT_X,
          MANAGEMENT_Z, ADMINISTRATION };
```

4.16.3 Documentation des constructeurs et destructeur

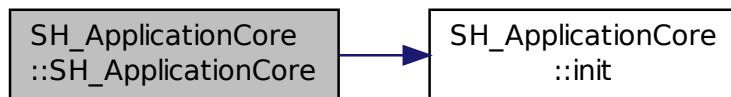
4.16.3.1 SH_ApplicationCore : :SH_ApplicationCore (QObject * parent = 0)

Définition à la ligne 14 du fichier [SH_ApplicationCore.cpp](#).

Références [init\(\)](#).

```
00014 :
00015     QObject (parent)
00016 {
00017     init ();
00018 }
```

Voici le graphe d'appel pour cette fonction :



4.16.4 Documentation des fonctions membres

4.16.4.1 bool SH_ApplicationCore : :balanceLogRoutine () [slot]

Définition à la ligne 112 du fichier [SH_ApplicationCore.cpp](#).

```
00112 {
00113     /*AppDatabase::getInstance () ->getDbConnection () .exec ("execute procedure logPeriodicBalance (H)");
00114     AppDatabase::getInstance () ->getDbConnection () .exec ("execute procedure logPeriodicBalance (D)");
00115     AppDatabase::getInstance () ->getDbConnection () .exec ("execute procedure logPeriodicBalance (W)");
00116     AppDatabase::getInstance () ->getDbConnection () .exec ("execute procedure logPeriodicBalance (M)");
00117     AppDatabase::getInstance () ->getDbConnection () .exec ("execute procedure logPeriodicBalance (Y)");*/
00118 }
```

4.16.4.2 void SH_ApplicationCore : :cancelReplacement () [slot]

Définition à la ligne 169 du fichier [SH_ApplicationCore.cpp](#).

Références [SH_InOutStateMachine : :cancelReplacement\(\)](#), et [m_currentFSM](#).

```
00170 {
00171     if (this->m_currentFSM) {
00172         emit this->m_currentFSM->cancelReplacement ();
00173     }
00174 }
```

4.16.4.3 bool SH_ApplicationCore : :cancelRunningThread () [slot]

Définition à la ligne 232 du fichier [SH_ApplicationCore.cpp](#).

Références [m_currentFSM](#).

```

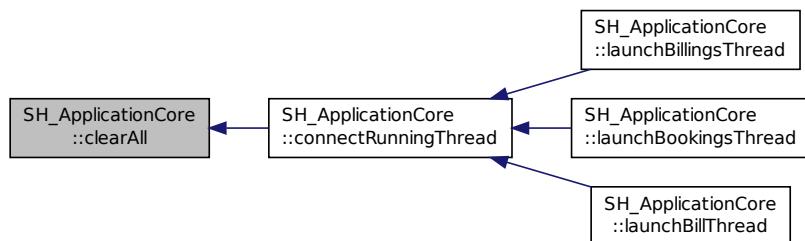
00233 {
00234     /*if(!this->m_currentFSM) {
00235         return true;
00236     }*/
00237     this->m_currentFSM->stop();
00238     bool ok = !this->m_currentFSM->isRunning();
00239     this->m_currentFSM = NULL;
00240     return ok;
00241 }

```

4.16.4.4 void SH_ApplicationCore ::clearAll() [signal]

Référencé par [connectRunningThread\(\)](#).

Voici le graphe des appels de cette fonction :



4.16.4.5 bool SH_ApplicationCore ::connectRunningThread() [protected]

Définition à la ligne [249](#) du fichier [SH_ApplicationCore.cpp](#).

Références [SH_InOutStateMachine ::clearAll\(\)](#), [clearAll\(\)](#), [SH_InOutStateMachine ::displayCalendar\(\)](#), [displayCalendar\(\)](#), [m_currentFSM](#), [SH_InOutStateMachine ::resendText\(\)](#), [resendText\(\)](#), [SH_InOutStateMachine ::sendText\(\)](#), et [sendText\(\)](#).

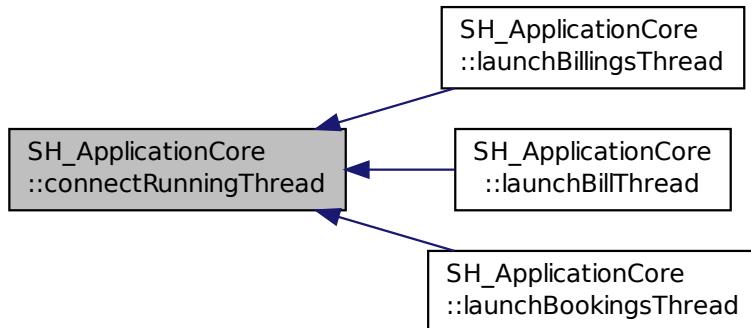
Référencé par [launchBillingsThread\(\)](#), [launchBillThread\(\)](#), et [launchBookingsThread\(\)](#).

```

00250 {
00251     /*if(!this->m_currentFSM) {
00252         return false;
00253     }*/
00254     qDebug() << "coucou";
00255     QObject::connect(this->m_currentFSM, &
00256                     SH_InOutStateMachine::sendText, this, &
00257                     SH_ApplicationCore::sendText, Qt::DirectConnection);
00258     QObject::connect(this->m_currentFSM, &
00259                     SH_InOutStateMachine::clearAll, this, &
00260                     SH_ApplicationCore::clearAll, Qt::DirectConnection);
00261     QObject::connect(this->m_currentFSM, &
00262                     SH_InOutStateMachine::resendText, this, &
00263                     SH_ApplicationCore::resendText, Qt::DirectConnection);
00264     QObject::connect(this->m_currentFSM, &
00265                     SH_InOutStateMachine::displayCalendar, this, &
00266                     SH_ApplicationCore::displayCalendar, Qt::DirectConnection);
00267     return this->m_currentFSM->isRunning();
00268 }

```

Voici le graphe des appelants de cette fonction :

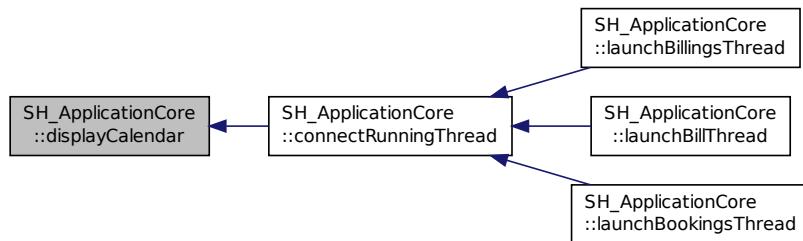


4.16.4.6 void SH_ApplicationCore ::currentFSMchanged() [signal]

4.16.4.7 void SH_ApplicationCore ::displayCalendar() [signal]

Référencé par [connectRunningThread\(\)](#).

Voici le graphe des appelants de cette fonction :



4.16.4.8 void SH_ApplicationCore ::init()

Définition à la ligne 35 du fichier [SH_ApplicationCore.cpp](#).

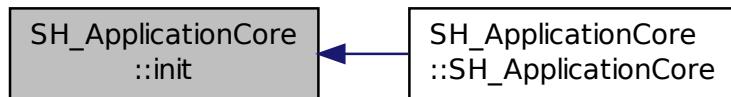
Références [m_currentUser](#).

Référencé par [SH_ApplicationCore\(\)](#).

```

00035
00036     this->m_currentUser = new SH_User();
00037 }
```

Voici le graphe des appelants de cette fonction :



4.16.4.9 bool SH_ApplicationCore ::launchBillingsThread() [slot]

Définition à la ligne 182 du fichier [SH_ApplicationCore.cpp](#).

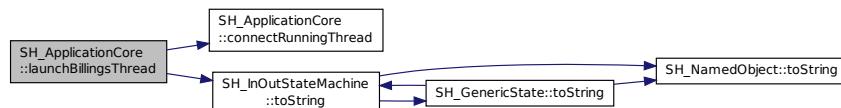
Références [connectRunningThread\(\)](#), [m_currentFSM](#), et [SH_InOutStateMachine ::toString\(\)](#).

```

00183 {
00184     qDebug() << "Hallo !";
00185     /*if(this->m_currentFSM) {
00186         return false;
00187     }*/
00188     qDebug() << "Hallo !";
00189     this->m_currentFSM= new SH_BillingCreationStateMachine("création facturation");
00190     this->m_currentFSM->start();
00191     qDebug() << this->m_currentFSM->toString() << " " << this->
00192     m_currentFSM->initialState();
00193     return this->connectRunningThread();
00194 }

```

Voici le graphe d'appel pour cette fonction :



4.16.4.10 bool SH_ApplicationCore ::launchBillThread() [slot]

Définition à la ligne 216 du fichier [SH_ApplicationCore.cpp](#).

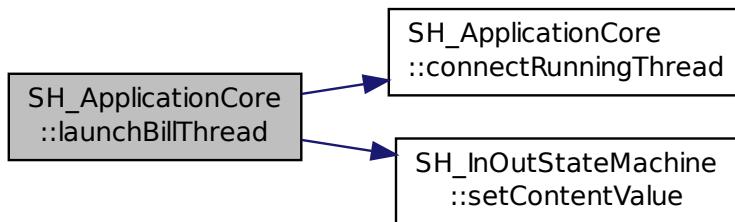
Références [connectRunningThread\(\)](#), [SH_User ::id](#), [m_currentFSM](#), [m_currentUser](#), et [SH_InOutStateMachine ::setContentValue\(\)](#).

```

00217 {
00218     /*if(this->m_currentFSM) {
00219         return false;
00220     }*/
00221     this->m_currentFSM= new SH_ServiceCharging("facturation prestation");
00222     this->m_currentFSM->setContentValue(QVariant(this->
00223     m_currentUser->id()), "BILL_ID");
00224     this->m_currentFSM->start();
00225     return this->connectRunningThread();
00226 }

```

Voici le graphe d'appel pour cette fonction :



4.16.4.11 bool SH_ApplicationCore ::launchBookingsThread() [slot]

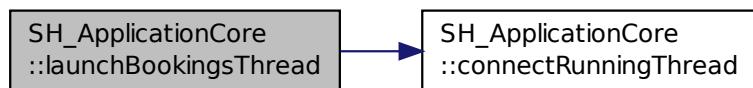
Définition à la ligne 201 du fichier [SH_ApplicationCore.cpp](#).

Références [connectRunningThread\(\)](#).

```

00202 {
00203     /*if(this->m_currentFSM) {
00204         return false;
00205     }*/
00206     /*this->m_currentFSM= new BookingCreationStateMachine("création facturation");*/
00207     /*this->m_currentFSM->start();*/
00208     return this->connectRunningThread();
00209 }
  
```

Voici le graphe d'appel pour cette fonction :



4.16.4.12 SH_ApplicationCore ::AppMode SH_ApplicationCore ::mode() const

Définition à la ligne 25 du fichier [SH_ApplicationCore.cpp](#).

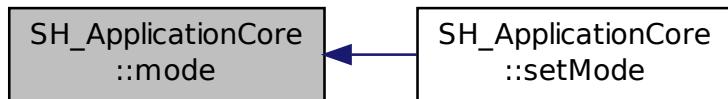
Références [m_mode](#).

Référencé par [setMode\(\)](#).

```

00026 {
00027     return m_mode;
00028 }
  
```

Voici le graphe des appels de cette fonction :



4.16.4.13 void SH_ApplicationCore ::modeChanged (QVariant mode) [signal]

4.16.4.14 void SH_ApplicationCore ::openTab (QVariant tabPos) [signal]

4.16.4.15 void SH_ApplicationCore ::receiveConfirmation () [slot]

Définition à la ligne 147 du fichier [SH_ApplicationCore.cpp](#).

Références [SH_InOutStateMachine ::confirmInput\(\)](#), et [m_currentFSM](#).

```

00148 {
00149
00150     emit this->m\_currentFSM->confirmInput\(\);
00151
00152 }
  
```

4.16.4.16 void SH_ApplicationCore ::receiveInput (QString in) [slot]

Définition à la ligne 125 du fichier [SH_ApplicationCore.cpp](#).

Références [m_currentFSM](#), et [SH_InOutStateMachine ::receiveInput\(\)](#).

```

00126 {
00127     qDebug() << "input received "<<in;
00128     emit this->m\_currentFSM->receiveInput\(in\);
00129
00130 }
  
```

4.16.4.17 void SH_ApplicationCore ::receiveValidation () [slot]

Définition à la ligne 136 du fichier [SH_ApplicationCore.cpp](#).

Références [m_currentFSM](#), et [SH_InOutStateMachine ::validateInput\(\)](#).

```

00137 {
00138
00139     emit this->m\_currentFSM->validateInput\(\);
00140
00141 }
  
```

4.16.4.18 void SH_ApplicationCore ::replaceInput (QString inputName) [slot]

Définition à la ligne 158 du fichier [SH_ApplicationCore.cpp](#).

Références [m_currentFSM](#), et [SH_InOutStateMachine ::replaceInput\(\)](#).

```

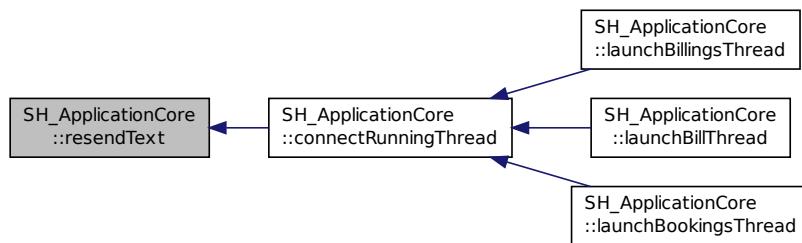
00159 {
00160     emit this->m_currentFSM->replaceInput(inputName);
00162
00163 }

```

4.16.4.19 void SH_ApplicationCore ::resendText (QString text) [signal]

Référencé par [connectRunningThread\(\)](#).

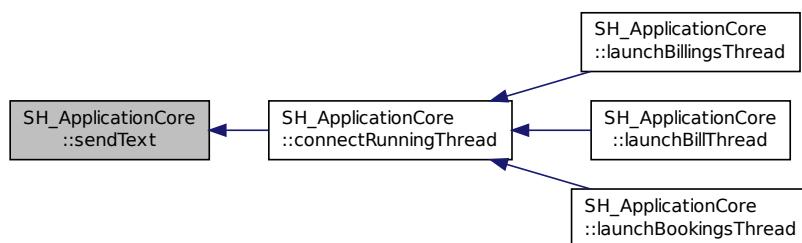
Voici le graphe des appelants de cette fonction :



4.16.4.20 void SH_ApplicationCore ::sendText (QString text) [signal]

Référencé par [connectRunningThread\(\)](#).

Voici le graphe des appelants de cette fonction :



4.16.4.21 void SH_ApplicationCore ::setMode (SH_ApplicationCore ::AppMode mode)

Définition à la ligne 44 du fichier [SH_ApplicationCore.cpp](#).

Références [ACCUEIL](#), [ADMINISTRATION](#), [CONNEXION](#), [SH_User ::exists\(\)](#), [SH_User ::isAdministrator\(\)](#), [SH_User ::isManagerX\(\)](#), [SH_User ::isManagerZ\(\)](#), [SH_User ::isReceptionist\(\)](#), [m_currentUser](#), [m_mode](#), [MANAGEMENT_X](#), [MANAGEMENT_Z](#), [mode\(\)](#), [SH_User ::name](#), et [RECEPTION](#).

```

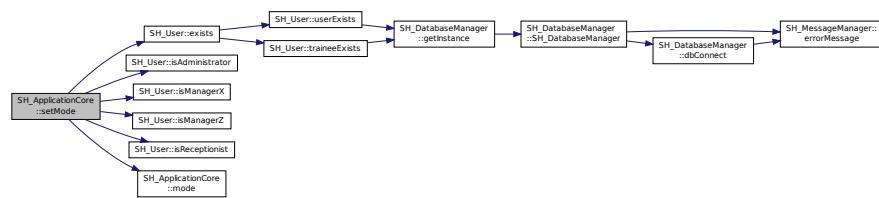
00045 {
00046     if(!this->m_currentUser || ! SH_User::exists(QVariant(this->
00047         m_currentUser->name())).toBool()) {
00047         this->m_mode = CONNEXION;

```

```

00048     } else {
00049         if((mode == ADMINISTRATION) && (!this->m_currentUser->
00050             isAdministrator()) || 
00051             ((mode == MANAGEMENT_X) && (!this->m_currentUser->
00052             isManagerX()) || 
00053             ((mode == MANAGEMENT_Z) && (!this->m_currentUser->
00054             isManagerZ()) || 
00055             ((mode == RECEPTION) && (!this->m_currentUser->
00056             isReceptionist())))) {
00057             this->m_mode = ACCUEIL;
00058         } else {
00059             this->m_mode = mode;
00060         }
00061     }
00062 }
```

Voici le graphe d'appel pour cette fonction :



4.16.4.22 bool SH_ApplicationCore :: setUser (QString login, QString pass) [slot]

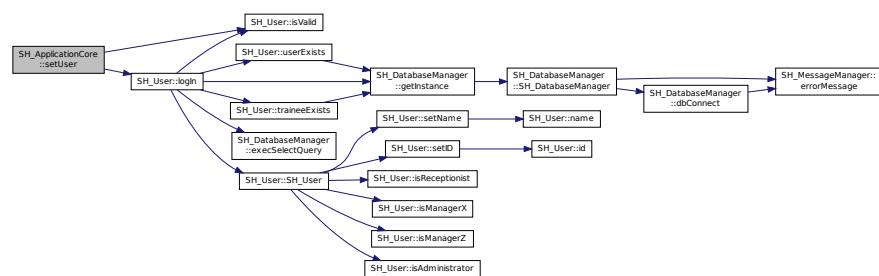
Définition à la ligne 86 du fichier [SH_ApplicationCore.cpp](#).

Références [SH_User ::isValid\(\)](#), [SH_User ::logIn\(\)](#), [m_currentUser](#), [SH_User ::name](#), et [userChanged\(\)](#).

```

00087 {
00088     this->m_currentUser = SH_User::logIn(login,pass);
00089     if(this->m_currentUser->isValid()) {
00090         emit userChanged(QVariant(this->m_currentUser->
00091             name()));
00092         return true;
00093     }
00094 }
```

Voici le graphe d'appel pour cette fonction :



4.16.4.23 SH_User * SH_ApplicationCore :: user () const

Définition à la ligne 65 du fichier [SH_ApplicationCore.cpp](#).

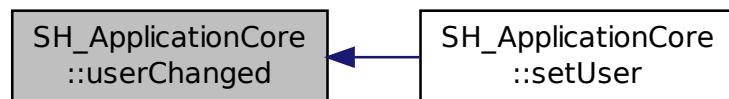
Références [m_currentUser](#).

```
00066 {
00067     return this->m_currentUser;
00068 }
```

4.16.4.24 void SH_ApplicationCore ::userChanged (QVariant name) [signal]

Référencé par [setUser\(\)](#).

Voici le graphe des appelants de cette fonction :



4.16.4.25 bool SH_ApplicationCore ::userExists (QString login) [slot]

Définition à la ligne [102](#) du fichier [SH_ApplicationCore.cpp](#).

Références [SH_User ::exists\(\)](#).

```
00103 {
00104     return SH_User::exists(login).toBool();
00105 }
```

Voici le graphe d'appel pour cette fonction :



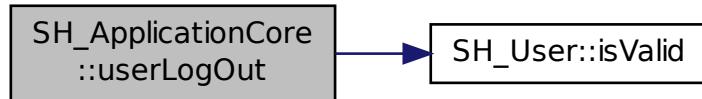
4.16.4.26 bool SH_ApplicationCore ::userLogOut() [slot]

Définition à la ligne [75](#) du fichier [SH_ApplicationCore.cpp](#).

Références [SH_User ::isValid\(\)](#), et [m_currentUser](#).

```
00076 {
00077     this->m_currentUser = new SH_User();
00078     return !this->m_currentUser->isValid();
00079 }
```

Voici le graphe d'appel pour cette fonction :



4.16.5 Documentation des données membres

4.16.5.1 SH_InOutStateMachine* SH_ApplicationCore::m_currentFSM [private]

m_currentFSM

Définition à la ligne 233 du fichier [SH_ApplicationCore.h](#).

Référencé par [cancelReplacement\(\)](#), [cancelRunningThread\(\)](#), [connectRunningThread\(\)](#), [launchBillingsThread\(\)](#), [launchBillThread\(\)](#), [receiveConfirmation\(\)](#), [receiveInput\(\)](#), [receiveValidation\(\)](#), et [replaceInput\(\)](#).

4.16.5.2 SH_User* SH_ApplicationCore::m_currentUser [private]

m_currentUser

Définition à la ligne 225 du fichier [SH_ApplicationCore.h](#).

Référencé par [init\(\)](#), [launchBillThread\(\)](#), [setMode\(\)](#), [setUser\(\)](#), [user\(\)](#), et [userLogOut\(\)](#).

4.16.5.3 AppMode SH_ApplicationCore::m_mode [private]

m_mode

Définition à la ligne 229 du fichier [SH_ApplicationCore.h](#).

Référencé par [mode\(\)](#), et [setMode\(\)](#).

4.16.6 Documentation des propriétés

4.16.6.1 AppMode SH_ApplicationCore::currentMode [read], [write]

Définition à la ligne 15 du fichier [SH_ApplicationCore.h](#).

4.16.6.2 SH_User SH_ApplicationCore::currentUser [read]

Définition à la ligne 14 du fichier [SH_ApplicationCore.h](#).

La documentation de cette classe a été générée à partir des fichiers suivants :

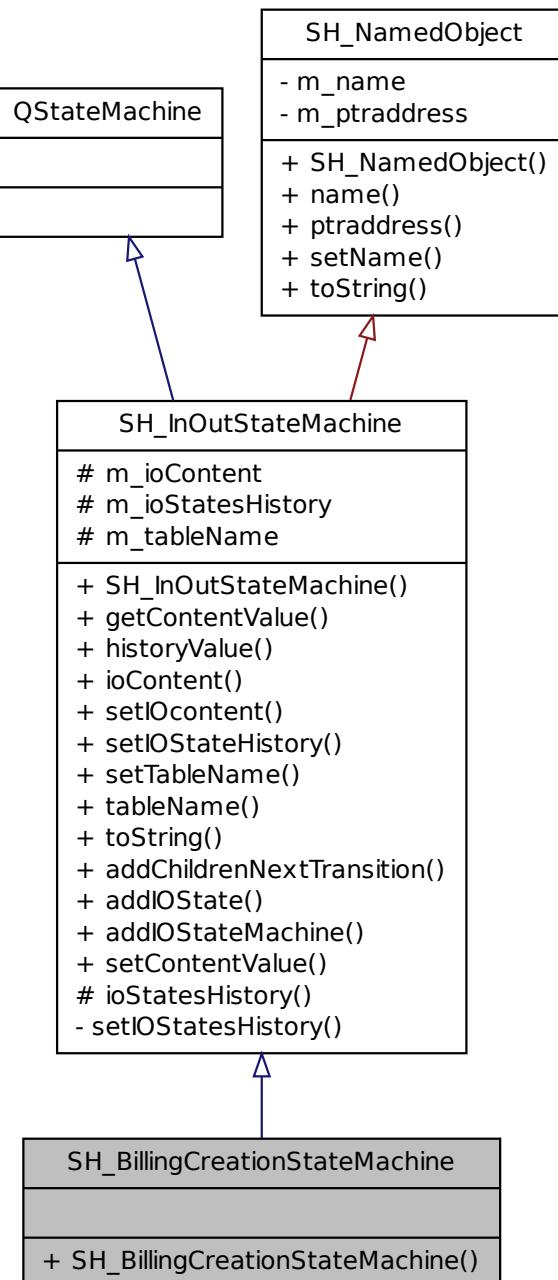
- [SH_ApplicationCore.h](#)
- [SH_ApplicationCore.cpp](#)

4.17 Référence de la classe SH_BillingCreationStateMachine

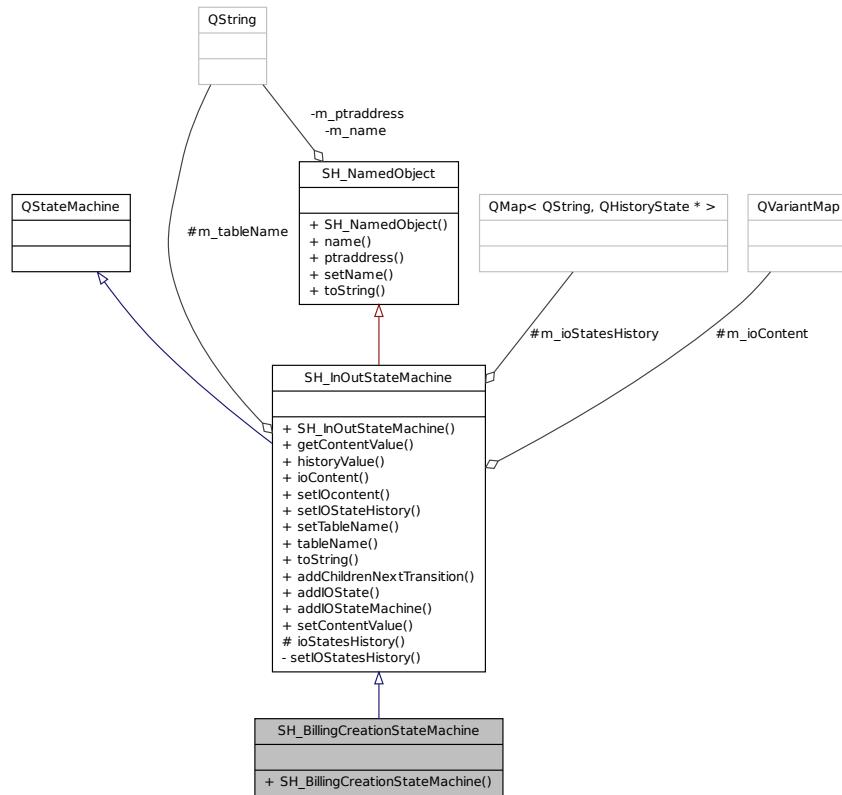
The [SH_BillingCreationStateMachine](#) class.

```
#include <SH_BillingCreation.h>
```

Graphe d'héritage de SH_BillingCreationStateMachine :



Graphe de collaboration de SH_BillingCreationStateMachine :



Connecteurs publics

- void `addChildsNextTransition` (QAbstractState *previousState, QAbstractState *nextState)
- void `addIOState` (SH_InOutState *state, QString field)
- void `addIOStateMachine` (SH_InOutStateMachine *fsm)
- void `setContentValue` (QVariant content, QString field)

Signaux

- void `cancelReplacement` ()
- void `clearAll` ()
- void `confirmInput` ()
- void `displayCalendar` ()
- void `displayFileDialog` ()
- void `next` ()
- void `receiveInput` (QString input)
- void `replaceInput` (QString field)
- void `resendText` (QString text, bool editable=false)
- void `sendText` (QString text, bool editable=false)
- void `validateInput` ()

Fonctions membres publiques

- `SH_BillingCreationStateMachine` (QString name, QObject *parent=0)
 SH_BillingCreationStateMachine
- QVariant `getContentValue` (QString field)
- QHistoryState * `historyValue` (QString field)
- QVariantMap `ioContent` () const
- void `setIOContent` (const QVariantMap &`ioContent`)
- void `setIOStateHistory` (QHistoryState *state, QString field)

- void `setTableName` (const QString &`tableName`)
- QString `tableName` () const
- QString `toString` ()

Fonctions membres protégées

- QMap< QString, QHistoryState * > `ioStatesHistory` () const

Attributs protégés

- QVariantMap `m_ioContent`
`m_ioContent`
- QMap< QString, QHistoryState * > `m_ioStatesHistory`
`m_ioStatesHistory`
- QString `m_tableName`
`m_tableName`

4.17.1 Description détaillée

The `SH_BillingCreationStateMachine` class.

Définition à la ligne 7 du fichier `SH_BillingCreation.h`.

4.17.2 Documentation des constructeurs et destructeur

4.17.2.1 SH_BillingCreationStateMachine : :SH_BillingCreationStateMachine (QString *name*, QObject * *parent* = 0)

`SH_BillingCreationStateMachine`.

Paramètres

<code>name</code>	
<code>parent</code>	<code>SH_BillingCreationStateMachine</code> : : <code>SH_BillingCreationStateMachine</code>

Définition à la ligne 14 du fichier `SH_BillingCreation.cpp`.

Références `Sh_LoopingInOutStateMachine` : :`addChildsNextTransition()`, `SH_InOutStateMachine` : :`addChildsNextTransition()`, `SH_InOutStateMachine` : :`addIOState()`, `SH_InOutStateMachine` : :`addIOStateMachine()`, `SH_QuestionState` : :`answerInvalid()`, `SH_InOutStateMachine` : :`confirmInput()`, `Sh_LoopingInOutStateMachine` : :`current()`, `SH_InOutStateMachine` : :`getContentValue()`, `SH_QuestionState` : :`givenAnswer()`, `SH_AdaptDatabaseState` : :`insertUpdate()`, `SH_InOutStateMachine` : :`m_ioContent`, `SH_InOutStateMachine` : :`m_tableName`, `SH_GenericState` : :`next()`, `SH_InOutStateMachine` : :`next()`, `SH_InOutStateMachine` : :`setContentValue()`, `Sh_LoopingInOutStateMachine` : :`setLimit()`, `Sh_LoopingInOutStateMachine` : :`setPersistentContentValue()`, `Sh_LoopingInOutStateMachine` : :`stopLooping()`, et `SH_InOutStateMachine` : :`toString()`.

```

00014
00015     SH_InOutStateMachine("BILLINGS",name, parent)
00016 {
00017     qDebug() << "facturation";
00018
00019     SH_StatementState* intro = new SH_StatementState("Création d'une
facturation", "intro billing creation");
00020     SH_NumericQuestionState* nbAdults = new
SH_NumericQuestionState("Veuillez entrer le nombre d'adultes", "adults billing
creation", 0);
00021     SH_NumericQuestionState* nbChildren = new
SH_NumericQuestionState("Veuillez entrer le nombre d'enfants", "children billing
creation", 0);
00022     SH_DateQuestionState* arrivingDate = new
SH_DateQuestionState("Veuillez entrer la date d'arrivée", "arriving date billing
creation", true,true);
00023     SH_DateQuestionState* departureDate = new
SH_DateQuestionState("Veuillez entrer la date de départ prévue", "departure date
billing creation", false,true);
00024     SH_DatabaseContentQuestionState* client = new
SH_DatabaseContentQuestionState("Veuillez entrer le nom du client à facturer
:
```

```

    ", "main client billing creation", "CLIENTS", "NAME");
00025     SH_ClientCreationStateMachine* clientCreation = new
SH_ClientCreationStateMachine("main client creation in billing creation");
00026     SH_NumericQuestionState* nbRooms = new
SH_NumericQuestionState("Veuillez entrer le nombre de chambres", "nb rooms billing
creation", 1);
00027     /*DatabaseContentQuestionState* type = new DatabaseContentQuestionState("Veuillez choisir le type de
facturation","billing type billing creation", "BILLINGSTYPES", "CODE");*/
00028     SH_DatabaseContentQuestionState* type = new
SH_DatabaseContentQuestionState("Veuillez choisir le type de facturation",
"billing type billing creation", "BILLINGSTYPES", "ID");
00029     Sh_LoopingInOutStateMachine* roomsAffectation = new
Sh_LoopingInOutStateMachine("ROOMSOCCUPATION", "rooms affectation billing
creation");
00030     Sh_LoopingInOutStateMachine* billsCreation = new
Sh_LoopingInOutStateMachine("BILLS", "bills creation billing creation");
00031     Sh_LoopingInOutStateMachine* clientList = new
Sh_LoopingInOutStateMachine("CLIENTS", "bills creation billing creation");
00032     SH_ConfirmationState* confirmPart1 = new
SH_ConfirmationState("Veuillez appuyer sur la touche \"CONFIRMER\" pour passer à
l'étape suivante", "confirm part 1");
00033     SH_AdaptDatabaseState* saveState = new
SH_AdaptDatabaseState("enregistrement de la machine "+

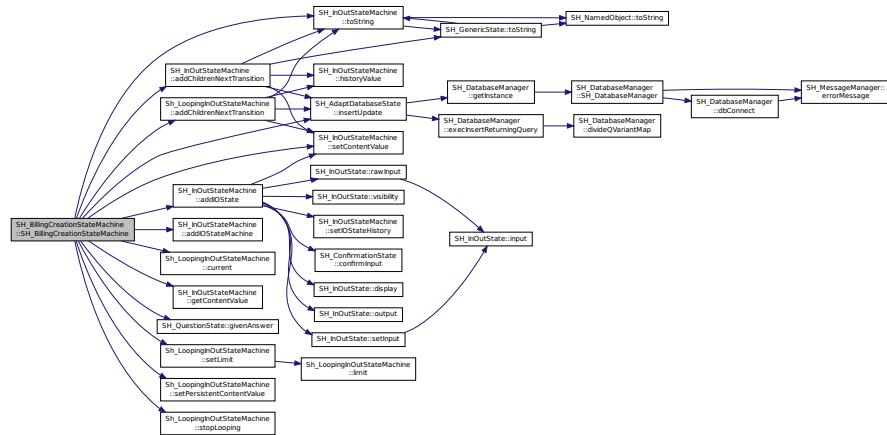
toString());
00034     SH_ConfirmationState* confirmAll = new
SH_ConfirmationState("Veuillez appuyer sur la touche \"CONFIRMER\" pour passer à
l'étape suivante", "confirm all");
00035     QFinalState* final = new QFinalState();
00036
00037
00038
00039     connect(nbAdults, &SH_GenericState::exited, [=]() {
00040         clientList->setLimit(getContentValue("NBADULTS").toInt()-1);
00041     });
00042
00043     connect(nbRooms, &SH_GenericState::exited, [=]() {
00044         roomsAffectation->setLimit(getContentValue("NBROOMS").toInt());
00045     });
00046
00047     connect(type, &SH_GenericState::exited, [=]() {
00048         billsCreation->setLimit(getContentValue("NBROOMS").toInt() * (
getContentValue("BILLINGTYPE_ID").toInt() % 3));
00049     });
00050
00051     connect(saveState, &SH_GenericState::exited, [=]() {
00052         roomsAffectation->setPersistentContentValue(
getContentValue("ID"), "BILLING_ID");
00053         billsCreation->setPersistentContentValue(
getContentValue("ID"), "BILLING_ID");
00054     });
00055
00056
00057
00058
00059
00060     SH_DatabaseContentQuestionState* rooms = new
SH_DatabaseContentQuestionState("Veuillez entrer un numéro de chambre", "room
billing creation", "ROOMS", "NUMBER");
00061     QFinalState* finalRooms = new QFinalState();
00062     roomsAffectation->addChildsNextTransition(rooms, finalRooms);
00063     roomsAffectation->addIOState(rooms, "ROOM_NUMBER");
00064     roomsAffectation->addState(finalRooms);
00065     roomsAffectation->setInitialState(rooms);
00066
00067
00068
00069     SH_DatabaseContentQuestionState* supplClient = new
SH_DatabaseContentQuestionState("Veuillez entrer le nom du client (adulte)
supplémentaire ou appuyer sur la touche \"CONFIRMER\" pour passer à la suite de la facturation", "other client
billing creation", "CLIENTS", "NAME");
00070     SH_ClientCreationStateMachine* supplClientCreation = new
SH_ClientCreationStateMachine("other client creation in billing creation");
00071     connect(clientList, &SH_InOutStateMachine::confirmInput, [=]() {
00072         clientList->stopLooping();
00073         supplClient->next();
00074     });
00075     QFinalState* finalClients = new QFinalState();
00076     clientList->addChildsNextTransition(supplClient, finalClients);
00077     connect(supplClient, &SH_QuestionState::answerInvalid, [=]() {
00078         supplClientCreation->setContentValue(supplClient->
givenAnswer(), "NAME");
00079         supplClient->addTransition(supplClient, SIGNAL(next()), supplClientCreation);
00080         emit supplClient->next();
00081     });
00082     clientList->addChildsNextTransition(supplClientCreation, finalClients);
00083     clientList->addState(finalClients);
00084     clientList->addState(supplClient);

```

```

00085     clientList->setInitialState(supplClient);
00086
00087
00088
00089
00090     QFinalState* finalBills = new QFinalState();
00091     SH_GenericState* bills = new SH_GenericState("bill id attribution");
00092     connect(bills, &SH_GenericState::entered, [=]() {
00093         this->setContentValue(QVariant(billsCreation->current()), "BILLINGBILL_ID");
00094         int billingType = getContentValue("BILLINGTYPE_ID").toInt();
00095         int billType;
00096         if(billingType <= 2) {
00097             billType = 1+billingType; /*nb facture par chambre*/
00098         } else {
00099             billType = (billsCreation->current() % (1+(billingType % 3)));
00100         }
00101         this->setContentValue(QVariant(billType), "BILLTYPE_ID");
00102         emit bills->next();
00103     });
00104     billsCreation->addChildrenNextTransition(bills, finalBills);
00105     billsCreation->addState(finalBills);
00106     billsCreation->addState(bills);
00107     billsCreation->setInitialState(bills);
00108
00109
00110
00111
00112
00113     this->addChildsNextTransition(intro, nbAdults);
00114     this->addChildsNextTransition(nbAdults, nbChildren);
00115     this->addChildsNextTransition(nbChildren, arrivingDate);
00116     this->addChildsNextTransition(arrivingDate, departureDate);
00117     this->addChildsNextTransition(departureDate, client);
00118     this->addChildsNextTransition(client, nbRooms);
00119     connect(client, &SH_QuestionState::answerInvalid, [=]() {
00120         clientCreation->setContentValue(client->givenAnswer(), "NAME");
00121         client->addTransition(client, SIGNAL(next()), clientCreation);
00122         emit client->next();
00123     });
00124     this->addChildsNextTransition(clientCreation, nbRooms);
00125     this->addChildsNextTransition(nbRooms, type);
00126     /*this->addChildsNextTransition(type, final);*/
00127     this->addChildsNextTransition(type, confirmPart1);
00128     confirmPart1->addTransition(confirmPart1, SIGNAL(next()), confirmPart1);
00129     connect(confirmPart1, &SH_GenericState::exited, [=]() {
00130         connect(saveState, &SH_GenericState::entered, [=]() {
00131             setContentValue(saveState->insertUpdate(
00132                 m_tableName, m_ioContent), "ID");
00133         });
00134         saveState->addTransition(saveState, SIGNAL(next()), confirmAll);
00135         saveState->addTransition(saveState, SIGNAL(next()), roomsAffection);
00136         this->addChildsNextTransition(roomsAffection, billsCreation);
00137         this->addChildsNextTransition(billsCreation, clientList);
00138         this->addChildsNextTransition(clientList, confirmAll);
00139         this->addChildsNextTransition(confirmAll, final);
00140
00141         this->addIOState(intro, "");
00142         this->addIOState(nbAdults, "NBADULTS");
00143         this->addIOState(nbChildren, "NBCHILDREN");
00144         this->addIOState(arrivingDate, "ARRIVINGDATE");
00145         this->addIOState(departureDate, "EXPECTEDDEPARTUREDATE");
00146         this->addIOState(client, "CLIENT_ID");
00147         this->addIOState(nbRooms, "NBROOMS");
00148         this->addIOState(type, "BILLINGTYPE_ID");
00149         this->addIOState(confirmPart1, "");
00150         this->addIOState(confirmAll, "");
00151         this->addIOStateMachine(billsCreation);
00152         this->addIOStateMachine(roomsAffection);
00153         this->addIOStateMachine(clientList);
00154         this->addState(final);
00155
00156     this->setInitialState(intro);
00157 }
```

Voici le graphe d'appel pour cette fonction :



4.17.3 Documentation des fonctions membres

4.17.3.1 void SH_InOutStateMachine :: addChildrenNextTransition (QAbstractState * previousState, QAbstractState * nextState) [slot], [inherited]

Définition à la ligne 250 du fichier [SH_IOStateMachine.cpp](#).

Références [SH_InOutStateMachine :: clearAll\(\)](#), [SH_InOutStateMachine :: historyValue\(\)](#), [SH_AdaptDatabaseState :: insertUpdate\(\)](#), [SH_InOutStateMachine :: m_ioContent](#), [SH_InOutStateMachine :: m_tableName](#), [SH_InOutStateMachine :: next\(\)](#), [SH_InOutStateMachine :: replaceInput\(\)](#), [SH_InOutStateMachine :: sendText\(\)](#), [SH_InOutStateMachine :: setContentValue\(\)](#), [SH_GenericState :: toString\(\)](#), et [SH_InOutStateMachine :: toString\(\)](#).

Référencé par [SH_BillingCreationStateMachine\(\)](#), et [SH_ClientCreationStateMachine :: SH_ClientCreationStateMachine\(\)](#).

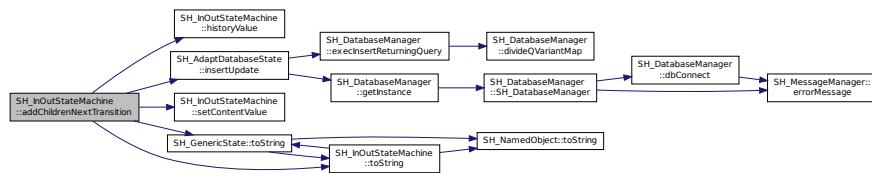
```

00251 {
00252     SH_InOutStateMachine* fsmPreviousState = qobject_cast<
00253         SH_InOutStateMachine*>(previousState);
00254     SH_GenericState* genPreviousState = qobject_cast<
00255         SH_GenericState*>(previousState);
00256     QFinalState* final = qobject_cast<QFinalState*>(nextState);
00257     if(final) {
00258         SH_AdaptDatabaseState* saveState = new
00259             SH_AdaptDatabaseState("enregistrement de la machine "+
00260             toString());
00261         if(genPreviousState) {
00262             genPreviousState->addTransition(genPreviousState, SIGNAL(next()), saveState);
00263         }
00264         if(fsmPreviousState) {
00265             fsmPreviousState->addTransition(fsmPreviousState, SIGNAL(next()), saveState);
00266         }
00267         if(genPreviousState || fsmPreviousState) {
00268             connect(previousState, &QAbstractState::exited, [=]() {
00269                 connect(saveState, &QAbstractState::entered, [=]() {
00270                     emit this->sendText("Merci !");
00271                     setContentType(saveState->insertUpdate(
00272                         m_tableName, m_ioContent), "ID");
00273                     emit this->clearAll();
00274                 });
00275             });
00276             saveState->addTransition(saveState, SIGNAL(next()), final);
00277         }
00278     } else {
00279         if(genPreviousState) {
00280             qDebug() << "next transition between " << genPreviousState->toString() << " and " <<
00281             nextState;
00282             genPreviousState->addTransition(genPreviousState, SIGNAL(next()), nextState);
00283         }
00284         if(fsmPreviousState) {
00285             qDebug() << "next transition between " << fsmPreviousState->toString() << " and " <<
00286             nextState;
  
```

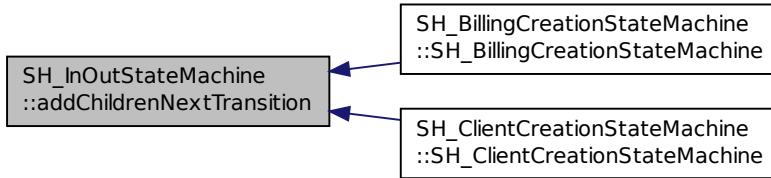
```

00280         fsmPreviousState->addTransition(fsmPreviousState, SIGNAL(next()), nextState);
00281     }
00282 }
00283 if(genPreviousState) {
00284     /*à faire au moment de l'entrée dans l'état previousState*/
00285     connect(genPreviousState, &QAbstractState::entered, [=]() {
00286         connect(this, &SH_InOutStateMachine::replaceInput, [=](
00287             QString field) {
00288             /*après avoir demandé à revenir sur un état précédent, on attend la fin de l'état actuel
00289             puis on retourne à l'historique de l'état désiré; celui-ci fini, on passe à l'état qui aurait du suivre celui
00290             pendant lequel on a demandé à revenir sur un état précédent*/
00291             QHistoryState* hState = historyValue(field);
00292             if(hState) /*si l'historique existe (on a déjà quitté l'état voulu)*/
00293                 hState->parentState()->addTransition(hState->parentState(), SIGNAL(
00294                     next()), nextState);
00295             genPreviousState->addTransition(genPreviousState, SIGNAL(
00296                 next()), hState);
00297         });
00298     });
00299 }
00300 }
00301 
```

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appels de cette fonction :



4.17.3.2 void SH_InOutStateMachine ::addIOState (SH_InOutState * state, QString field) [slot], [inherited]

Définition à la ligne 110 du fichier [SH_IOStateMachine.cpp](#).

Références [SH_ConfirmationState ::confirmInput\(\)](#), [SH_InOutState ::display\(\)](#), [SH_InOutStateMachine ::displayCalendar\(\)](#), [SH_InOutStateMachine ::displayFileDialog\(\)](#), [SH_InOutState ::output\(\)](#), [SH_InOutState ::rawInput\(\)](#), [SH_InOutStateMachine ::receiveInput\(\)](#), [SH_InOutState ::resendInput\(\)](#), [SH_InOutStateMachine ::resendText\(\)](#), [SH_InOutState ::sendOutput\(\)](#), [SH_InOutStateMachine ::sendText\(\)](#), [SH_InOutStateMachine ::setContentValue\(\)](#), [SH_InOutState ::setInput\(\)](#), [SH_InOutStateMachine ::setIOStateHistory\(\)](#), [SH_InOutStateMachine ::validateInput\(\)](#), et [SH_InOutState ::visibility\(\)](#).

Référencé par [SH_BillingCreationStateMachine\(\)](#), [SH_ClientCreationStateMachine ::SH_ClientCreationStateMachine\(\)](#), et [SH_ServiceCharging ::SH_ServiceCharging\(\)](#).

```

00111 {
00112     /*à faire au moment de l'entrée dans l'état state*/

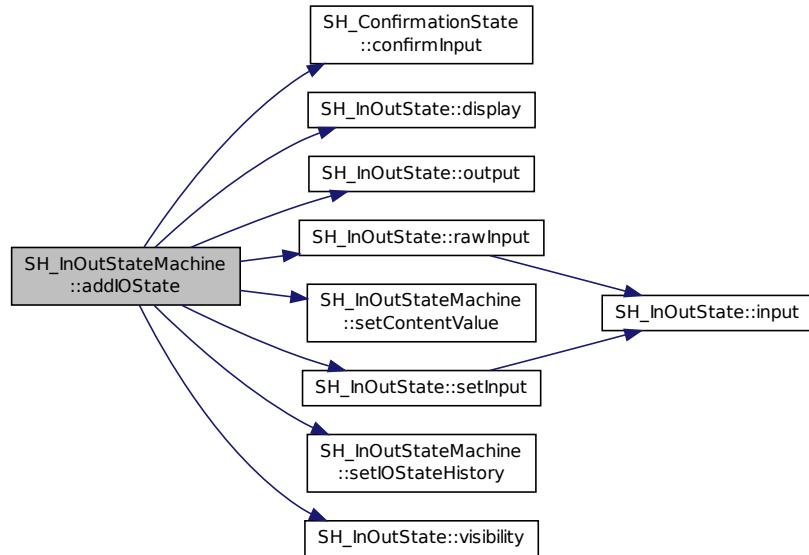
```

```

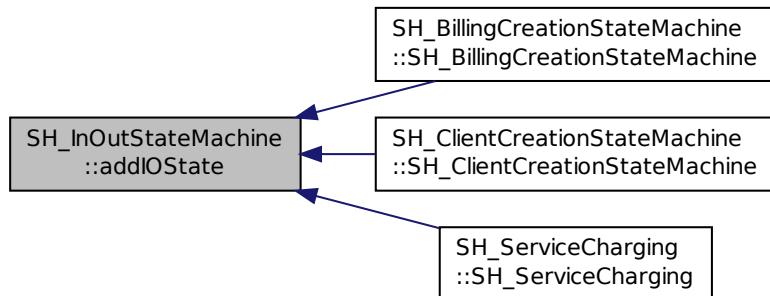
0013     connect(state, &QState::entered, [=]() {
0014         qDebug() << "entered !";
0015         state->display(true);
0016         connect(this, &SH_InOutStateMachine::receiveInput, state, &
0017             SH_InOutState::setInput); /* la réception d'une valeur entraîne son enregistrement
0018             comme entrée de l'utilisateur auprès de l'état*/
0019     ){ qDebug() << "hello world !"; state->setInput(in);} /* la réception d'une valeur entraîne son
0020             enregistrement comme entrée de l'utilisateur auprès de l'état*/
0021     connect(state, &SH_InOutState::sendOutput, [=](QVariant out) {qDebug() <<
0022         "connected !"; emit this->sendText(out.toString(), false);});
0023     connect(state, &SH_InOutState::resendInput, [=](QVariant in) {emit this->
0024         resendText(in.toString(), true);});
0025         if(state->visibility()) {
0026             state->sendOutput(QVariant(state->output()));
0027         } else {
0028             qDebug() << "invisible";
0029         }
0030     });
0031     SH_ValidationState *validationState = qobject_cast<
0032         SH_ValidationState*>(state);
0033     if(validationState) {
0034         /*à faire au moment de l'entrée dans l'état state*/
0035         connect(validationState, &QState::entered, [=]() {
0036             connect(this, &SH_InOutStateMachine::validateInput,
0037                 validationState, &SH_ValidationState::confirmInput);
0038         });
0039     }
0040     SH_ConfirmationState *confirmationState = qobject_cast<
0041         SH_ConfirmationState*>(state);
0042     if(confirmationState) {
0043         /*à faire au moment de l'entrée dans l'état state*/
0044         connect(confirmationState, &QState::entered, [=]() {
0045             connect(this, &SH_InOutStateMachine::validateInput,
0046                 confirmationState, &SH_ConfirmationState::confirmInput);
0047         });
0048     }
0049     SH_DateQuestionState *dateState = qobject_cast<
0050         SH_DateQuestionState*>(state);
0051     if(dateState) {
0052         /*à faire au moment de l'entrée dans l'état state*/
0053         connect(dateState, &QState::entered, this, &
0054             SH_InOutStateMachine::displayCalendar);
0055     }
0056     SH_FileSelectionState *fileState = qobject_cast<
0057         SH_FileSelectionState*>(state);
0058     if(fileState) {
0059         /*à faire au moment de l'entrée dans l'état state*/
0060         connect(fileState, &QState::entered, this, &
0061             SH_InOutStateMachine::displayFileDialog);
0062     }
0063     /*à faire au moment de la sortie de l'état state*/
0064     connect(state, &QState::exited, [=]() {
0065         qDebug() << "exited !";
0066         if(!field.isEmpty()) {
0067             setContentValue(state->rawInput(), field);
0068             /*gestion de l'historique des états pour pouvoir revenir à l'état state après l'avoir quitté*/
0069             QHistoryState* hState = new QHistoryState(state);
0070             setIOStateHistory(hState, field);
0071         }
0072         state->disconnect(this); /*plus aucune action sur l'état ne pourra être provoquée par la machine*/
0073     });
0074
0075     QAbstractState* astate = qobject_cast<QAbstractState *>(state);
0076     if(astate) {
0077         addState(astate);
0078     }
0079 }

```

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :



4.17.3.3 void SH_InOutStateMachine ::addIOStateMachine (SH_InOutStateMachine * fsm) [slot], [inherited]

Définition à la ligne 175 du fichier [SH_IOStateMachine.cpp](#).

Références `SH_InOutStateMachine ::cancelReplacement()`, `SH_InOutStateMachine ::confirmInput()`, `SH_InOutStateMachine ::displayCalendar()`, `SH_InOutStateMachine ::receiveInput()`, `SH_InOutStateMachine ::replaceInput()`, `SH_InOutStateMachine ::resendText()`, `SH_InOutStateMachine ::sendText()`, et `SH_InOutStateMachine ::validateInput()`.

Référencé par [SH_BillingCreationStateMachine\(\)](#).

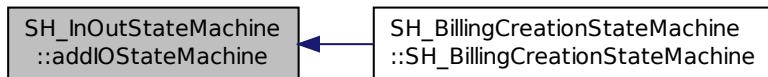
```

00176 {
00177     /* à faire au moment de l'entrée dans la machine d'état fsm*/
  
```

```

00178     connect(fsm, &QState::entered, [=]() {
00179         connect(this, &SH_InOutStateMachine::receiveInput, fsm, &
00180             SH_InOutStateMachine::receiveInput);
00181         connect(this, &SH_InOutStateMachine::sendText, fsm, &
00182             SH_InOutStateMachine::sendText);
00183         connect(this, &SH_InOutStateMachine::resendText, fsm, &
00184             SH_InOutStateMachine::resendText);
00185         connect(this, &SH_InOutStateMachine::confirmInput, fsm, &
00186             SH_InOutStateMachine::confirmInput);
00187         connect(this, &SH_InOutStateMachine::validateInput, fsm, &
00188             SH_InOutStateMachine::validateInput);
00189         connect(this, &SH_InOutStateMachine::replaceInput, fsm, &
00190             SH_InOutStateMachine::replaceInput);
00191         connect(this, &SH_InOutStateMachine::cancelReplacement, fsm,
00192             &SH_InOutStateMachine::cancelReplacement);
00193         connect(this, &SH_InOutStateMachine::displayCalendar, fsm, &
00194             SH_InOutStateMachine::displayCalendar);
00195     });
00196     /*à faire au moment de la sortie de la machine d'état fsm*/
00197     connect(fsm, &QState::exited, [=]() {
00198         fsm->disconnect(this); /*plus aucune action sur la machine d'état fille ne pourra être provoquée
00199         par la machine mère*/
00200     });
00201 }
00202
00203 }
```

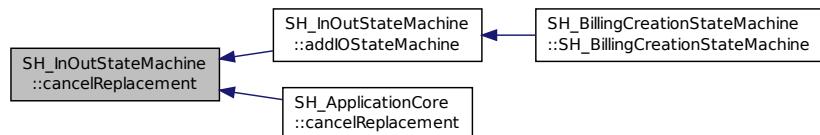
Voici le graphe des appels de cette fonction :



4.17.3.4 void SH_InOutStateMachine ::cancelReplacement() [signal], [inherited]

Référencé par [SH_InOutStateMachine ::addIOStateMachine\(\)](#), et [SH_ApplicationCore ::cancelReplacement\(\)](#).

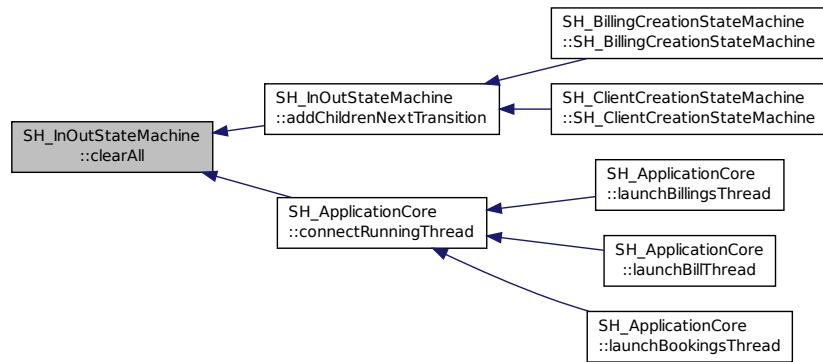
Voici le graphe des appels de cette fonction :



4.17.3.5 void SH_InOutStateMachine ::clearAll() [signal], [inherited]

Référencé par [SH_InOutStateMachine ::addChildrenNextTransition\(\)](#), et [SH_ApplicationCore ::connectRunningThread\(\)](#).

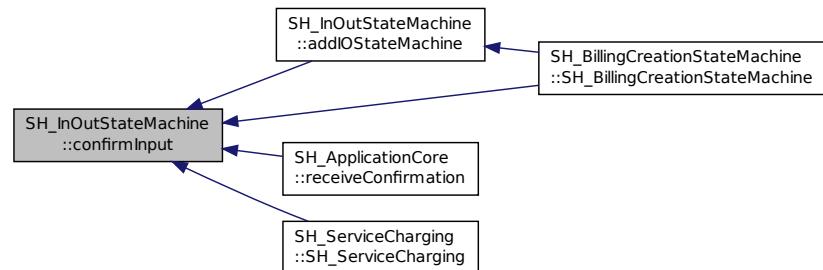
Voici le graphe des appelants de cette fonction :



4.17.3.6 void SH_InOutStateMachine ::confirmInput() [signal], [inherited]

Référencé par [SH_InOutStateMachine ::addIOStateMachine\(\)](#), [SH_ApplicationCore ::receiveConfirmation\(\)](#), [SH_BillingCreationStateMachine\(\)](#), et [SH_ServiceCharging ::SH_ServiceCharging\(\)](#).

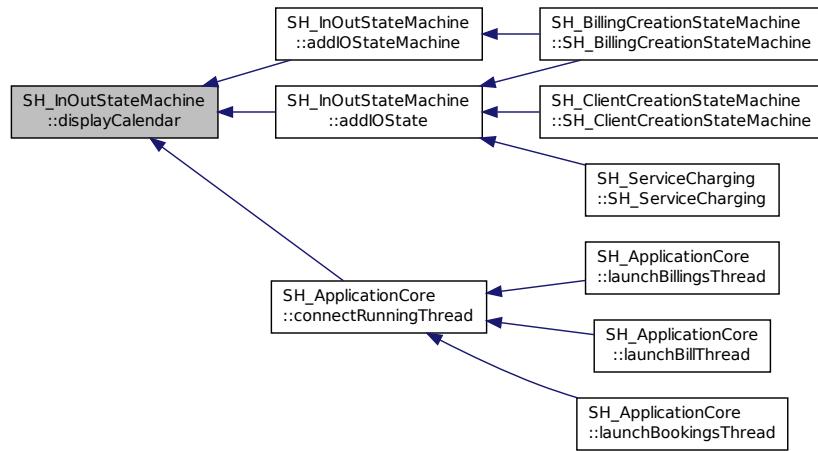
Voici le graphe des appelants de cette fonction :



4.17.3.7 void SH_InOutStateMachine ::displayCalendar() [signal], [inherited]

Référencé par [SH_InOutStateMachine ::addIOState\(\)](#), [SH_InOutStateMachine ::addIOStateMachine\(\)](#), et [SH_ApplicationCore ::connectRunningThread\(\)](#).

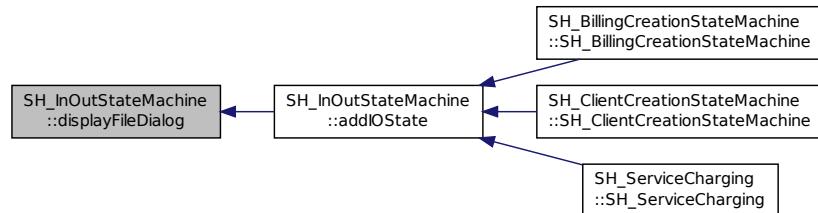
Voici le graphe des appelants de cette fonction :



4.17.3.8 void SH_InOutStateMachine ::displayFileDialog() [signal], [inherited]

Référencé par [SH_InOutStateMachine ::addIOState\(\)](#).

Voici le graphe des appelants de cette fonction :



4.17.3.9 QVariant SH_InOutStateMachine ::getContentValue(QString field) [inherited]

Définition à la ligne 65 du fichier [SH_IOStateMachine.cpp](#).

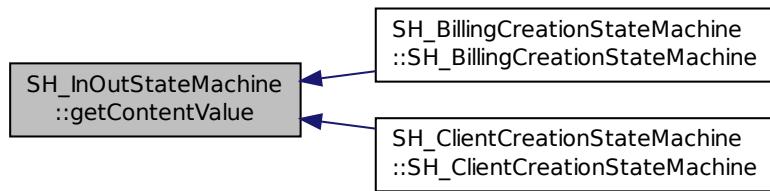
Références [SH_InOutStateMachine ::m_ioContent](#).

Référencé par [SH_BillingCreationStateMachine\(\)](#), et [SH_ClientCreationStateMachine ::SH_ClientCreationStateMachine\(\)](#).

```

00066 {
00067     return m_ioContent.value(field);
00068 }
  
```

Voici le graphe des appelants de cette fonction :



4.17.3.10 QHistoryState * SH_InOutStateMachine ::historyValue (QString field) [inherited]

Définition à la ligne 238 du fichier [SH_IOStateMachine.cpp](#).

Références [SH_InOutStateMachine ::m_ioStatesHistory](#).

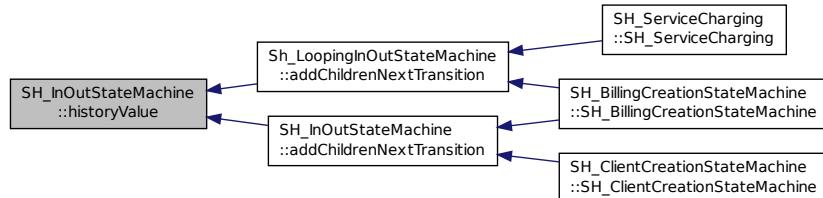
Référencé par [Sh_LoopingInOutStateMachine ::addChildrenNextTransition\(\)](#), et [SH_InOutStateMachine ::addChildsNextTransition\(\)](#).

```

00239 {
00240     return m_ioStatesHistory.value(field);
00241 }

```

Voici le graphe des appelants de cette fonction :



4.17.3.11 QVariantMap SH_InOutStateMachine ::ioContent () const [inherited]

Définition à la ligne 43 du fichier [SH_IOStateMachine.cpp](#).

Références [SH_InOutStateMachine ::m_ioContent](#).

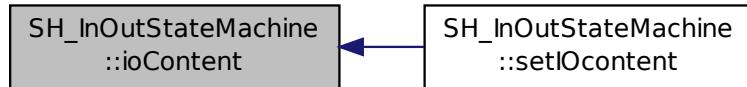
Référencé par [SH_InOutStateMachine ::setIOcontent\(\)](#).

```

00044 {
00045     return m_ioContent;
00046 }

```

Voici le graphe des appelants de cette fonction :



4.17.3.12 QMap< QString, QHistoryState * > SH_InOutStateMachine ::ioStatesHistory() const [protected], [inherited]

Définition à la ligne 202 du fichier [SH_IOStateMachine.cpp](#).

Références [SH_InOutStateMachine ::m_ioStatesHistory](#).

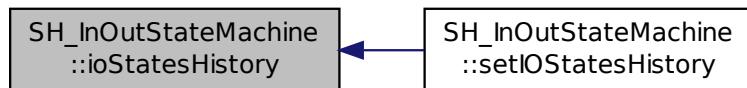
Référencé par [SH_InOutStateMachine ::setIOStatesHistory\(\)](#).

```

00203 {
00204     return m_ioStatesHistory;
00205 }

```

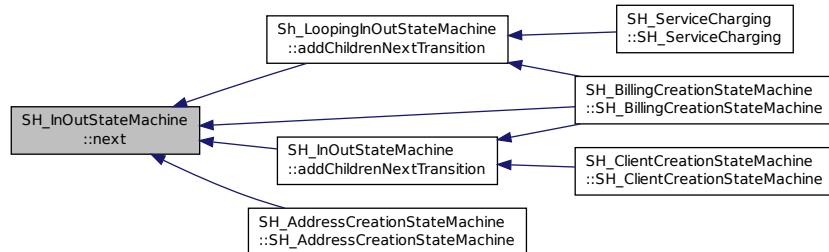
Voici le graphe des appelants de cette fonction :



4.17.3.13 void SH_InOutStateMachine ::next() [signal], [inherited]

Référencé par [Sh_LoopingInOutStateMachine ::addChildrenNextTransition\(\)](#), [SH_InOutStateMachine ::addChildrenNextTransition\(\)](#), [SH_AddressCreationStateMachine ::SH_AddressCreationStateMachine\(\)](#), et [SH_BillingCreationStateMachine\(\)](#).

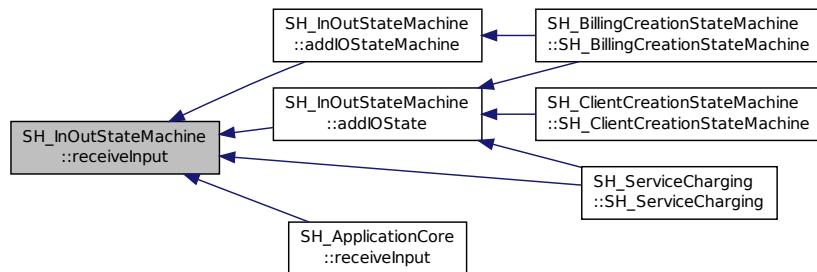
Voici le graphe des appelants de cette fonction :



4.17.3.14 void SH_InOutStateMachine ::receiveInput (QString *input*) [signal], [inherited]

Référencé par [SH_InOutStateMachine ::addIOState\(\)](#), [SH_InOutStateMachine ::addIOStateMachine\(\)](#), [SH_ApplicationCore ::receiveInput\(\)](#), et [SH_ServiceCharging ::SH_ServiceCharging\(\)](#).

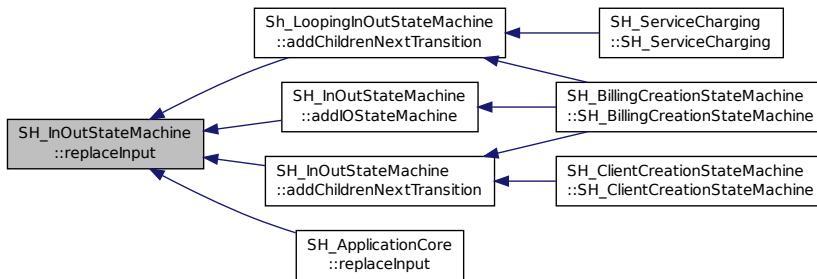
Voici le graphe des appelleurs de cette fonction :



4.17.3.15 void SH_InOutStateMachine ::replaceInput (QString *field*) [signal], [inherited]

Référencé par [Sh_LoopingInOutStateMachine ::addChildsNextTransition\(\)](#), [SH_InOutStateMachine ::addChildsNextTransition\(\)](#), [SH_InOutStateMachine ::addIOStateMachine\(\)](#), et [SH_ApplicationCore ::replaceInput\(\)](#).

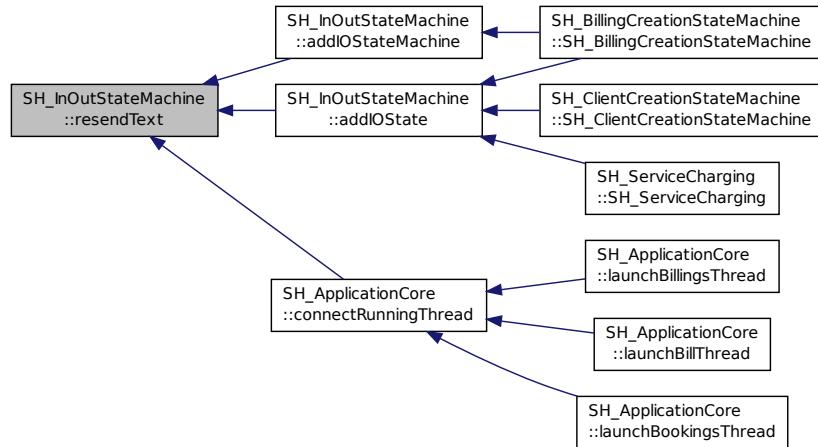
Voici le graphe des appelleurs de cette fonction :



4.17.3.16 void SH_InOutStateMachine ::resendText (QString *text*, bool *editable* = false) [signal], [inherited]

Référencé par [SH_InOutStateMachine ::addIOState\(\)](#), [SH_InOutStateMachine ::addIOStateMachine\(\)](#), et [SH_ApplicationCore ::connectRunningThread\(\)](#).

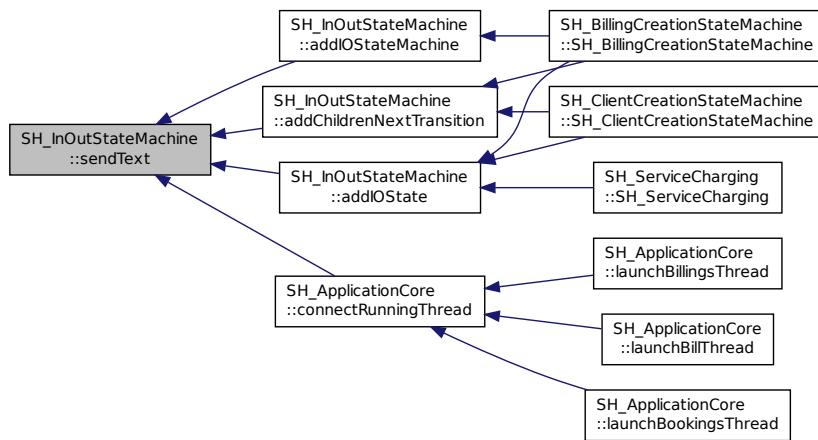
Voici le graphe des appelants de cette fonction :



4.17.3.17 void SH_InOutStateMachine ::sendText (QString text, bool editable = false) [signal], [inherited]

Référencé par [SH_InOutStateMachine ::addChildrenNextTransition\(\)](#), [SH_InOutStateMachine ::addIOState\(\)](#), [SH_InOutStateMachine ::addIOStateMachine\(\)](#), et [SH_ApplicationCore ::connectRunningThread\(\)](#).

Voici le graphe des appelants de cette fonction :



4.17.3.18 void SH_InOutStateMachine ::setContentValue (QVariant content, QString field) [slot], [inherited]

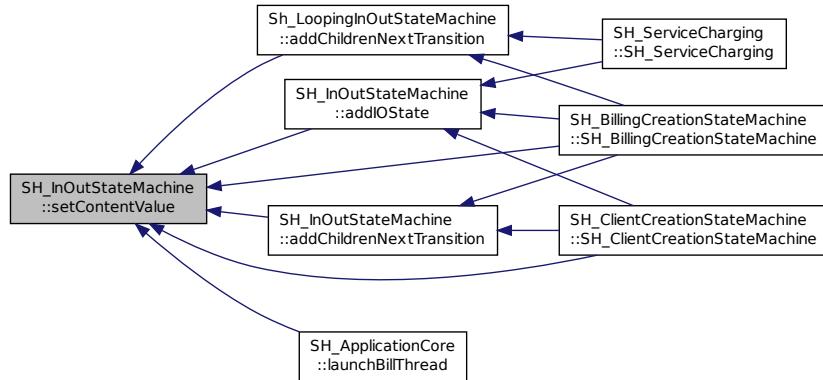
Définition à la ligne 99 du fichier [SH_IOStateMachine.cpp](#).

Références [SH_InOutStateMachine ::m_ioContent](#).

Référencé par [Sh_LoopingInOutStateMachine ::addChildrenNextTransition\(\)](#), [SH_InOutStateMachine ::addChildrenNextTransition\(\)](#), [SH_InOutStateMachine ::addIOState\(\)](#), [SH_ApplicationCore ::launchBillThread\(\)](#), [SH_BillingCreationStateMachine\(\)](#), et [SH_ClientCreationStateMachine ::SH_ClientCreationStateMachine\(\)](#).

```
00100 {
00101     m_ioContent.insert(field, content);
00102 }
```

Voici le graphe des appels de cette fonction :



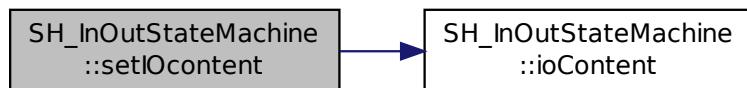
4.17.3.19 void SH_InOutStateMachine ::setIOcontent (const QVariantMap & ioContent) [inherited]

Définition à la ligne 54 du fichier [SH_IOSMachine.cpp](#).

Références [SH_InOutStateMachine ::ioContent\(\)](#), et [SH_InOutStateMachine ::m_ioContent](#).

```
00055 {
00056     m_ioContent = ioContent;
00057 }
```

Voici le graphe d'appel pour cette fonction :



4.17.3.20 void SH_InOutStateMachine ::setIOStateHistory (QHistoryState * state, QString field) [inherited]

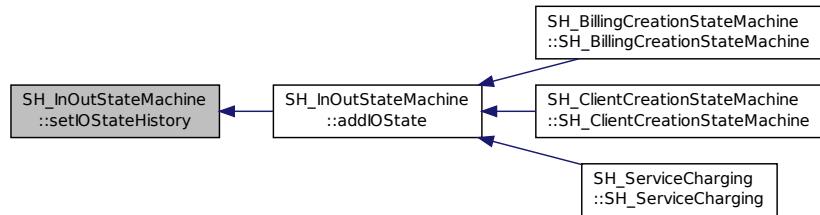
Définition à la ligne 226 du fichier [SH_IOSMachine.cpp](#).

Références [SH_InOutStateMachine ::m_ioStatesHistory](#).

Référencé par [SH_InOutStateMachine ::addIOState\(\)](#).

```
00227 {
00228     m_ioStatesHistory.insert(field, state); /*remplacement si plusieurs fois*/
00229 }
```

Voici le graphe des appelants de cette fonction :



4.17.3.21 void SH_InOutStateMachine : :setTableName (const QString & tableName) [inherited]

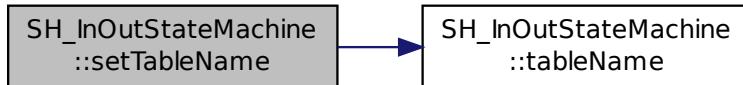
Définition à la ligne 87 du fichier [SH_IOStateMachine.cpp](#).

Références [SH_InOutStateMachine : :m_tableName](#), et [SH_InOutStateMachine : :tableName\(\)](#).

```

00088 {
00089     m_tableName = tableName;
00090 }
```

Voici le graphe d'appel pour cette fonction :



4.17.3.22 QString SH_InOutStateMachine : :tableName () const [inherited]

Définition à la ligne 76 du fichier [SH_IOStateMachine.cpp](#).

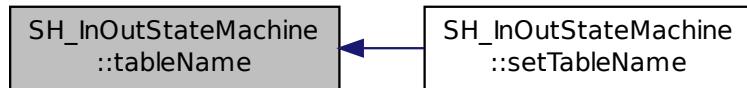
Références [SH_InOutStateMachine : :m_tableName](#).

Référencé par [SH_InOutStateMachine : :setTableName\(\)](#).

```

00077 {
00078     return m_tableName;
00079 }
```

Voici le graphe des appelants de cette fonction :



4.17.3.23 QString SH_InOutStateMachine ::toString() [virtual], [inherited]

Réimplémentée à partir de [SH_NamedObject](#).

Définition à la ligne 26 du fichier [SH_IOSStateMachine.cpp](#).

Références [SH_NamedObject ::toString\(\)](#), et [SH_GenericState ::toString\(\)](#).

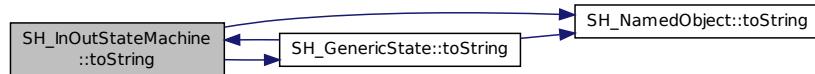
Référencé par [Sh_LoopingInOutStateMachine ::addChildsNextTransition\(\)](#), [SH_InOutStateMachine ::addChildsNextTransition\(\)](#), [SH_ApplicationCore ::launchBillingsThread\(\)](#), [SH_BillingCreationStateMachine\(\)](#), et [SH_GenericState ::toString\(\)](#).

```

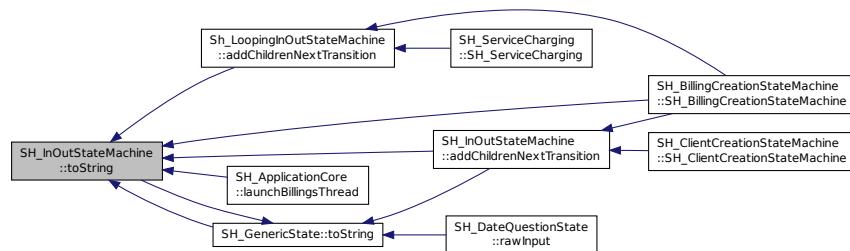
00027 {
00028     QObject* parent = this->parent();
00029     SH_GenericState* par = qobject_cast<SH_GenericState *>(parent);
00030     if(par) {
00031         return SH_NamedObject::toString() + " [descending from "+par-
00032             toString()+" ] ";
00033     } else {
00034         return SH_NamedObject::toString();
00035     }

```

Voici le graphe d'appel pour cette fonction :



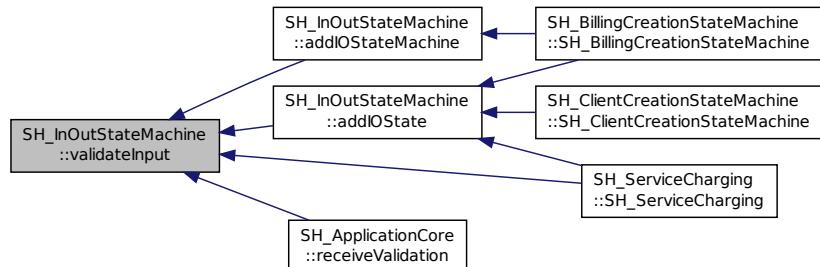
Voici le graphe des appelants de cette fonction :



4.17.3.24 void SH_InOutStateMachine ::validateInput() [signal], [inherited]

Référencé par [SH_InOutStateMachine ::addIOState\(\)](#), [SH_InOutStateMachine ::addIOSMachine\(\)](#), [SH_ApplicationCore ::receiveValidation\(\)](#), et [SH_ServiceCharging ::SH_ServiceCharging\(\)](#).

Voici le graphe des appelleurs de cette fonction :



4.17.4 Documentation des données membres

4.17.4.1 QVariantMap SH_InOutStateMachine ::m_ioContent [protected], [inherited]

`m_ioContent`

Définition à la ligne 209 du fichier [SH_IOStateMachine.h](#).

Référencé par [Sh_LoopingInOutStateMachine ::addChildsNextTransition\(\)](#), [SH_InOutStateMachine ::addChildsNextTransition\(\)](#), [SH_InOutStateMachine ::getContentValue\(\)](#), [SH_InOutStateMachine :::ioContent\(\)](#), [SH_InOutStateMachine ::setContentValue\(\)](#), [SH_InOutStateMachine ::setIOcontent\(\)](#), et [SH_BillingCreationStateMachine\(\)](#).

4.17.4.2 QMap<QString, QHistoryState*> SH_InOutStateMachine ::m_ioStatesHistory [protected], [inherited]

`m_ioStatesHistory`

Définition à la ligne 217 du fichier [SH_IOStateMachine.h](#).

Référencé par [SH_InOutStateMachine ::historyValue\(\)](#), [SH_InOutStateMachine :::ioStatesHistory\(\)](#), [SH_InOutStateMachine ::setIOStateHistory\(\)](#), et [SH_InOutStateMachine ::setIOStatesHistory\(\)](#).

4.17.4.3 QString SH_InOutStateMachine ::m_tableName [protected], [inherited]

`m_tableName`

Définition à la ligne 213 du fichier [SH_IOStateMachine.h](#).

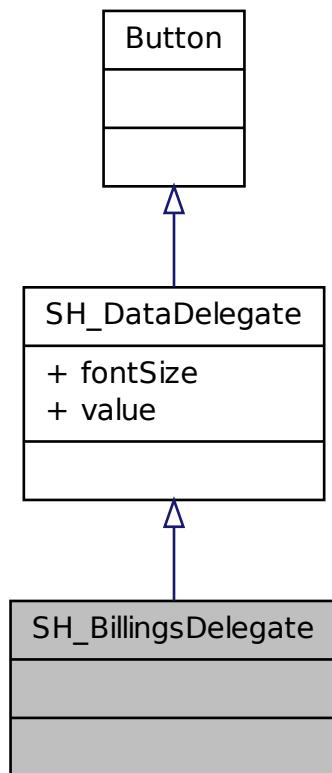
Référencé par [Sh_LoopingInOutStateMachine ::addChildsNextTransition\(\)](#), [SH_InOutStateMachine ::addChildsNextTransition\(\)](#), [SH_InOutStateMachine :::setTableName\(\)](#), [SH_BillingCreationStateMachine\(\)](#), et [SH_InOutStateMachine ::tableName\(\)](#).

La documentation de cette classe a été générée à partir des fichiers suivants :

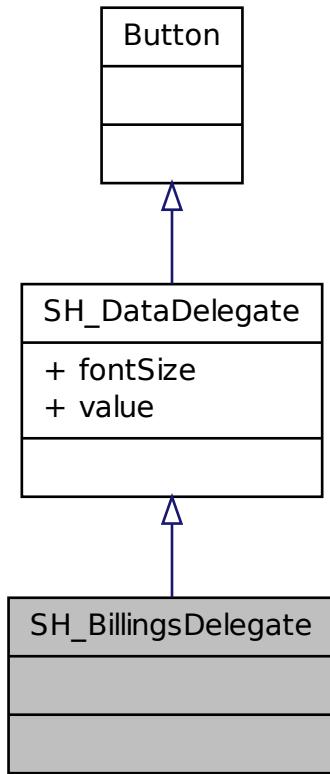
- logic/[SH_BillingCreation.h](#)
- logic/[SH_BillingCreation.cpp](#)

4.18 Référence de la classe SH_BillingsDelegate

Graphe d'héritage de SH_BillingsDelegate :



Graphe de collaboration de SH_BillingsDelegate :



Propriétés

- int `fontSize`
- string `value`

4.18.1 Description détaillée

Définition à la ligne 4 du fichier [SH_BillingsDelegate.qml](#).

4.18.2 Documentation des propriétés

4.18.2.1 int SH_DataDelegate ::fontSize [inherited]

Définition à la ligne 9 du fichier [SH_DataDelegate.qml](#).

4.18.2.2 string SH_DataDelegate ::value [inherited]

Définition à la ligne 7 du fichier [SH_DataDelegate.qml](#).

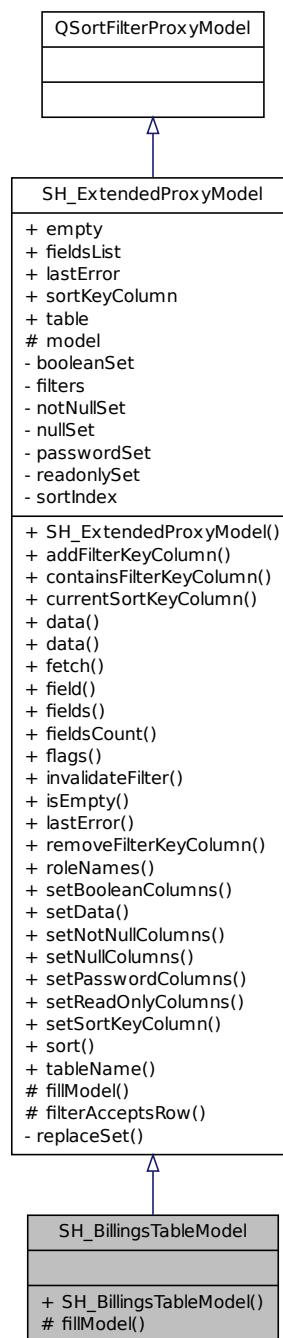
La documentation de cette classe a été générée à partir du fichier suivant :

- [views/qml/SH_BillingsDelegate.qml](#)

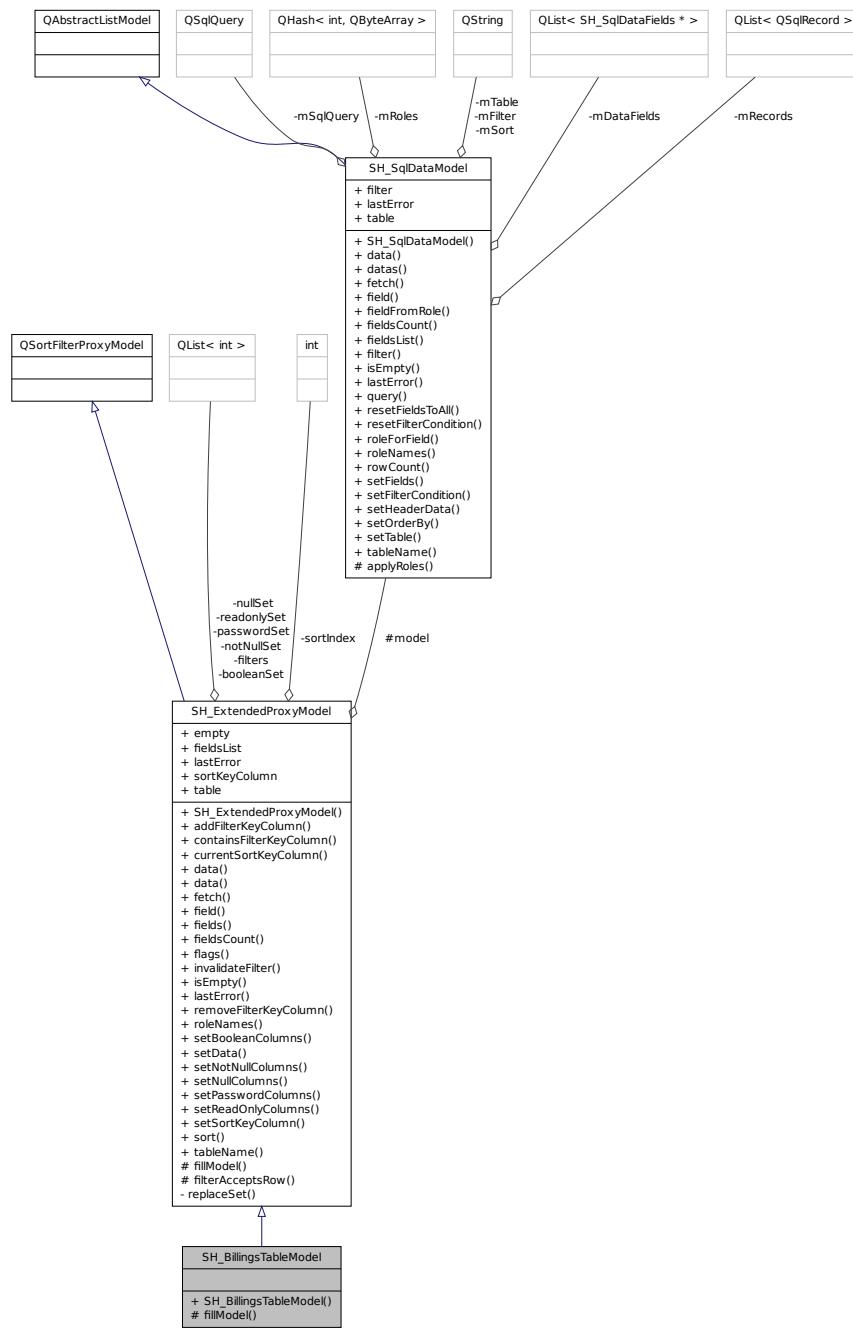
4.19 Référence de la classe SH_BillingsTableModel

```
#include <SH_BillingsTableModel.h>
```

Graphe d'héritage de SH_BillingsTableModel :



Graphe de collaboration de SH_BillingsTableModel :



Signaux

- void `sortChanged ()`

Fonctions membres publiques

- `SH_BillingsTableModel (QObject *parent=0)`
- `Q_INVOKABLE void addFilterKeyColumn (int column)`
- `Q_INVOKABLE bool containsFilterKeyColumn (int column)`
- `const int currentSortKeyColumn () const`

- Q_INVOKABLE QVariant **data** (int row, int column) const
- Q_INVOKABLE QVariant **data** (const QModelIndex &index, int role=Qt::DisplayRole) const
- Q_INVOKABLE bool **fetch** (QString **tableName**="", QString **filter**="", QString **sort**="", QStringList **fields**=QStringList())
- Q_INVOKABLE SH_SqlDataFields * **field** (int i) const
- const QString **fields** () const
- Q_INVOKABLE int **fieldsCount** () const
- Q_INVOKABLE Qt::ItemFlags **flags** (const QModelIndex &index) const
- void **invalidateFilter** ()
- const bool **isEmpty** () const
- const QString **lastError** () const
- Q_INVOKABLE void **removeFilterKeyColumn** (int column)
- virtual Q_INVOKABLE QHash<int, QByteArray > **roleNames** () const
- void **setBooleanColumns** (QList<int> **boolCols**)
- Q_INVOKABLE bool **setData** (const QModelIndex &index, const QVariant &value, int role=Qt::EditRole)
- void **setNotNullColumns** (QList<int> **notNullCols**)
- void **setNullColumns** (QList<int> **nullCols**)
- void **setPasswordColumns** (QList<int> **passwordCols**)
- void **setReadOnlyColumns** (QList<int> **readonlyCols**)
- void **setSortKeyColumn** (int column)
- Q_INVOKABLE void **sort** (int column, Qt::SortOrder newOrder=Qt::AscendingOrder)
- const QString **tableName** () const

Fonctions membres protégées

- void **fillModel** ()
- bool **filterAcceptsRow** (int source_row, const QModelIndex &source_parent) const

Attributs protégés

- SH_SqlDataModel * **model**

Propriétés

- bool **empty**
- QString **fieldsList**
- QString **lastError**
- int **sortKeyColumn**
- QString **table**

4.19.1 Description détaillée

Définition à la ligne 13 du fichier [SH_BillingsTableModel.h](#).

4.19.2 Documentation des constructeurs et destructeur

4.19.2.1 SH_BillingsTableModel : :SH_BillingsTableModel (QObject * parent = 0)

Paramètres

<i>parent</i>	
---------------	--

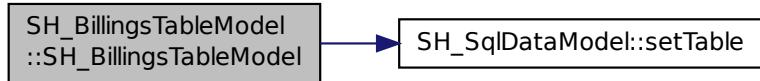
Définition à la ligne 10 du fichier [SH_BillingsTableModel.cpp](#).

Références [SH_ExtendedProxyModel](#) : :model, et [SH_SqlDataModel](#) : :setTable().

```

00010
00011     SH_ExtendedProxyModel (parent)
00012 {
00013     SH_ExtendedProxyModel::model->setTable ("BILLINGSINFOS");
00014 }
```

Voici le graphe d'appel pour cette fonction :



4.19.3 Documentation des fonctions membres

4.19.3.1 void SH_ExtendedProxyModel : :addFilterKeyColumn (int column) [inherited]

Définition à la ligne 259 du fichier [SH_ExtendedProxyModel.cpp](#).

Références [SH_ExtendedProxyModel : :filters](#).

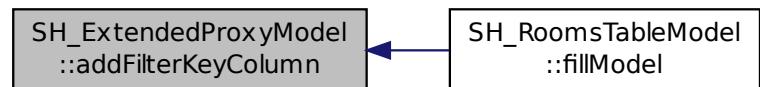
Référencé par [SH_RoomsTableModel : :fillModel\(\)](#).

```

00260 {
00261     this->filters.append\(column\);
00262 }

```

Voici le graphe des appels de cette fonction :



4.19.3.2 bool SH_ExtendedProxyModel : :containsFilterKeyColumn (int column) [inherited]

Définition à la ligne 225 du fichier [SH_ExtendedProxyModel.cpp](#).

Références [SH_ExtendedProxyModel : :filters](#).

```

00226 {
00227     return this->filters.contains\(column\);
00228 }

```

4.19.3.3 const int SH_ExtendedProxyModel : :currentSortKeyColumn () const [inline], [inherited]

Définition à la ligne 38 du fichier [SH_ExtendedSqlProxyModel.h](#).

Références [SH_ExtendedProxyModel : :sortIndex](#).

```
00038 {return this->sortIndex;}
```

4.19.3.4 QVariant SH_ExtendedProxyModel ::data (int row, int column) const [inherited]

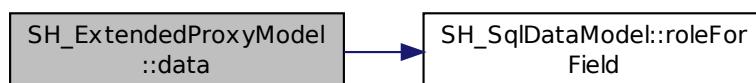
Définition à la ligne 269 du fichier [SH_ExtendedProxyModel.cpp](#).

Références [SH_ExtendedProxyModel ::model](#), et [SH_SqlDataModel ::roleForField\(\)](#).

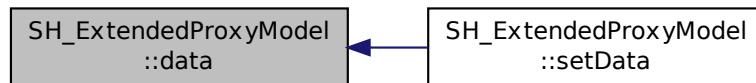
Référencé par [SH_ExtendedProxyModel ::setData\(\)](#).

```
00270 {
00271     QModelIndex modelIndex = this->index(row, 0);
00272     return this->data(modelIndex, this->model->roleForField(column));
00273 }
```

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appels de cette fonction :



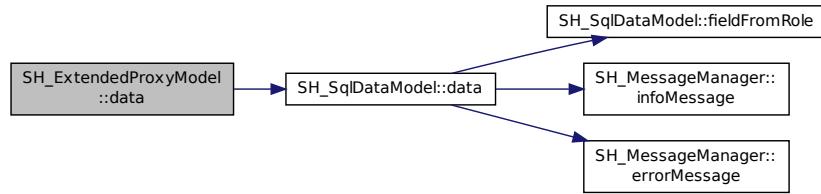
4.19.3.5 QVariant SH_ExtendedProxyModel ::data (const QModelIndex & index, int role = Qt ::DisplayRole) const [inherited]

Définition à la ligne 127 du fichier [SH_ExtendedProxyModel.cpp](#).

Références [SH_ExtendedProxyModel ::booleanSet](#), [SH_SqlDataModel ::data\(\)](#), [SH_ExtendedProxyModel ::filters](#), [SH_ExtendedProxyModel ::model](#), et [SH_ExtendedProxyModel ::passwordSet](#).

```
00128 {
00129     if (index.isValid())
00130     {
00131         if (this->booleanSet.contains(role))
00132         {
00133             return index.data(Qt::EditRole).toBool() ? QVariant(Qt::Checked) : QVariant(Qt::Unchecked);
00134         }
00135         else if (this->passwordSet.contains(role))
00136         {
00137             return QVariant("****");
00138         }
00139         else if (!this->filters.contains(role))
00140         {
00141             QModelIndex source_index = QSortFilterProxyModel::mapToSource(index);
00142             if (source_index.isValid())
00143                 return this->model->data(source_index, role);
00144             }
00145         }
00146     }
00147     return QVariant();
00148 }
```

Voici le graphe d'appel pour cette fonction :



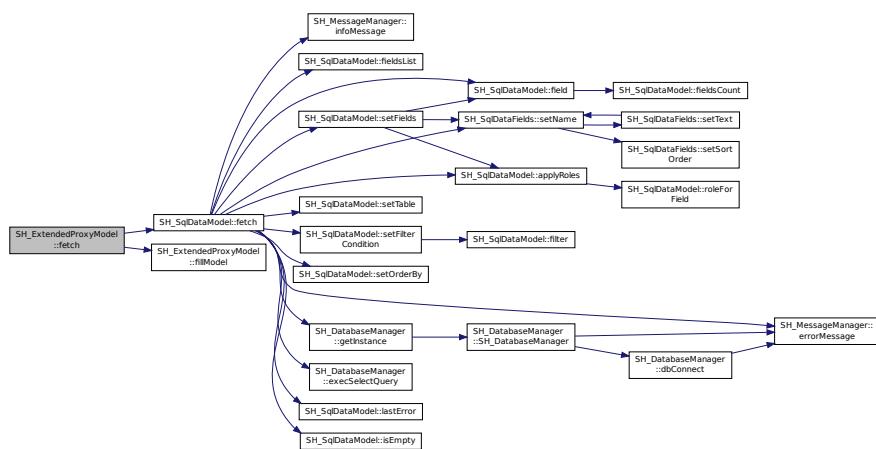
4.19.3.6 bool SH_ExtendedProxyModel::fetch (QString tableName = "", QString filter = "", QString sort = "", QStringList fields = QStringList()) [inherited]

Définition à la ligne 280 du fichier [SH_ExtendedProxyModel.cpp](#).

Références `SH_SqlDataModel` : `:fetch()`, `SH_ExtendedProxyModel` : `:fillModel()`, et `SH_ExtendedProxyModel` : `:model`.

```
00281 {  
00282     bool fetched = this->model->fetch(tableName, filter, sort,  
        fields);  
00283     if (fetched)  
00284     {  
00285         this->fillModel();  
00286     }  
00287     this->setSourceModel(this->model);  
00288     return fetched;  
00289 }
```

Voici le graphe d'appel pour cette fonction :



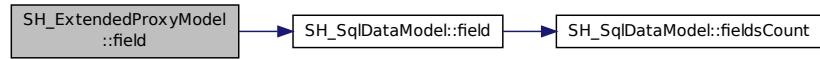
4.19.3.7 **Q_INVOKABLE SH_SqlDataFields* SH_ExtendedProxyModel ::field (int i) const [inline], [inherited]**

Définition à la ligne 82 du fichier `SH_ExtendedSqlProxyModel.h`.

Références SH `SqlDataModel::field()`, et SH `ExtendedProxyModel::model`.

```
00082 { return this->model->field(i); }
```

Voici le graphe d'appel pour cette fonction :



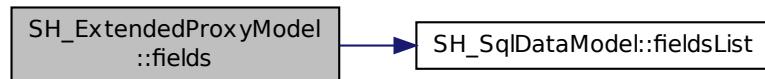
4.19.3.8 const QString SH_ExtendedProxyModel ::fields() const [inline], [inherited]

Définition à la ligne 52 du fichier [SH_ExtendedSqlProxyModel.h](#).

Références [SH_SqlDataModel ::fieldsList\(\)](#), et [SH_ExtendedProxyModel ::model](#).

```
00052 { if(this->model->fieldsList().isEmpty()){ return "*"; } else { return this->model->fieldsList().join(", "); } }
```

Voici le graphe d'appel pour cette fonction :



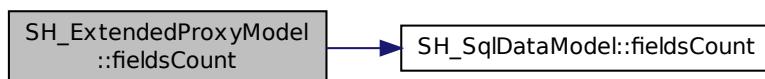
4.19.3.9 Q_INVOKABLE int SH_ExtendedProxyModel ::fieldsCount() const [inline], [inherited]

Définition à la ligne 89 du fichier [SH_ExtendedSqlProxyModel.h](#).

Références [SH_SqlDataModel ::fieldsCount\(\)](#), et [SH_ExtendedProxyModel ::model](#).

```
00089 { return this->model->fieldsCount(); }
```

Voici le graphe d'appel pour cette fonction :



4.19.3.10 void SH_BillingsTableModel : :fillModel () [protected], [virtual]

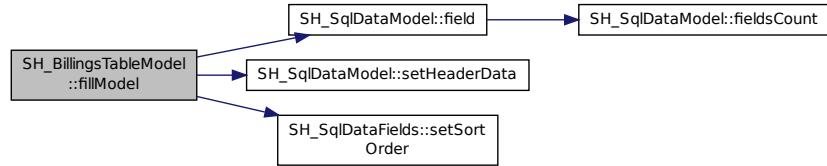
Implémente SH_ExtendedProxyModel.

Définition à la ligne 22 du fichier SH_BillingsTableModel.cpp.

Références SH_SqlDataModel : :field(), SH_ExtendedProxyModel : :model, SH_SqlDataModel : :setHeaderData(), et SH_SqlDataFields : :setSortOrder().

```
00023 {
00024     SH_ExtendedProxyModel::model->setHeaderData(1, Qt::Horizontal,
00025     QObject::tr("Nom client"));
00026     SH_ExtendedProxyModel::model->setHeaderData(3, Qt::Horizontal,
00027     QObject::tr("Chambre"));
00028     SH_ExtendedProxyModel::model->setHeaderData(4, Qt::Horizontal,
00029     QObject::tr("Date arrivée"));
00030     SH_ExtendedProxyModel::model->setHeaderData(5, Qt::Horizontal,
00031     QObject::tr("Date départ prévue"));
00032     SH_ExtendedProxyModel::model->field(4)->
00033         setSortOrder(Qt::AscendingOrder);
00034 }
```

Voici le graphe d'appel pour cette fonction :



4.19.3.11 bool SH_ExtendedProxyModel : :filterAcceptsRow (int source_row, const QModelIndex & source_parent) const [protected], [inherited]

Définition à la ligne 92 du fichier SH_ExtendedProxyModel.cpp.

Références SH_ExtendedProxyModel : :notNullSet, et SH_ExtendedProxyModel : :nullSet.

```
00093 {
00094     Q_UNUSED(source_parent);
00095
00096     if (!this->notNullSet.isEmpty())
00097     {
00098         QSqlQueryModel *m = static_cast<QSqlQueryModel *>(sourceModel());
00099         foreach(int column, this->notNullSet)
00100         {
00101             if (m->record(source_row).isNull(column))
00102             {
00103                 return false;
00104             }
00105         }
00106     }
00107
00108     if (!this->nullSet.isEmpty())
00109     {
00110         QSqlQueryModel *m = static_cast<QSqlQueryModel *>(sourceModel());
00111         foreach(int column, this->nullSet)
00112         {
00113             if (!m->record(source_row).isNull(column))
00114             {
00115                 return false;
00116             }
00117         }
00118     }
00119     return true;
00120 }
```

4.19.3.12 Qt ::itemFlags SH_ExtendedProxyModel ::flags (const QModelIndex & index) const [inherited]

Définition à la ligne 179 du fichier [SH_ExtendedProxyModel.cpp](#).

Références [SH_ExtendedProxyModel ::booleanSet](#), et [SH_ExtendedProxyModel ::readonlySet](#).

```
00180 {
00181     if (!index.isValid())
00182     {
00183         return Qt::ItemIsEnabled;
00184     }
00185     if (!this->booleanSet.isEmpty())
00186     {
00187         return Qt::ItemIsUserCheckable | Qt::ItemIsSelectable | Qt::ItemIsEnabled;
00188     }
00189     else if (!this->readonlySet.isEmpty())
00190     {
00191         return Qt::ItemIsSelectable;
00192     }
00193     else
00194     {
00195         return QSortFilterProxyModel::flags(index);
00196     }
00197 }
00198 }
```

4.19.3.13 void SH_ExtendedProxyModel ::invalidateFilter () [inherited]

Définition à la ligne 205 du fichier [SH_ExtendedProxyModel.cpp](#).

Références [SH_ExtendedProxyModel ::filters](#).

```
00206 {
00207     this->filters.clear();
00208 }
```

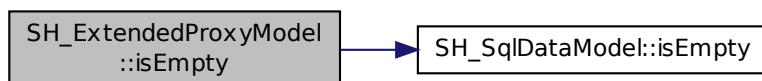
4.19.3.14 const bool SH_ExtendedProxyModel ::isEmpty () const [inline], [inherited]

Définition à la ligne 66 du fichier [SH_ExtendedSqlProxyModel.h](#).

Références [SH_SqlDataManager ::isEmpty\(\)](#), et [SH_ExtendedProxyModel ::model](#).

```
00066 { return this->model->isEmpty(); }
```

Voici le graphe d'appel pour cette fonction :



4.19.3.15 const QString SH_ExtendedProxyModel ::lastError () const [inline], [inherited]

Définition à la ligne 59 du fichier [SH_ExtendedSqlProxyModel.h](#).

Références [SH_SqlDataManager ::lastError](#), et [SH_ExtendedProxyModel ::model](#).

```
00059 { return this->model->lastError(); }
```

4.19.3.16 void SH_ExtendedProxyModel : :removeFilterKeyColumn (int *column*) [inherited]

Définition à la ligne 215 du fichier [SH_ExtendedProxyModel.cpp](#).

Références [SH_ExtendedProxyModel : :filters](#).

```
00216 {
00217     this->filters.removeAt(this->filters.indexOf(column));
00218 }
```

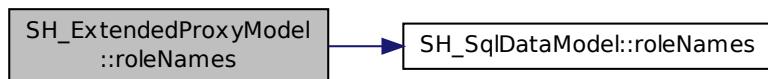
4.19.3.17 virtual Q_INVOKABLE QHash<int, QByteArray> SH_ExtendedProxyModel : :roleNames () const [inline], [virtual], [inherited]

Définition à la ligne 165 du fichier [SH_ExtendedSqlProxyModel.h](#).

Références [SH_ExtendedProxyModel : :model](#), et [SH_SqldataModel : :roleNames\(\)](#).

```
00165 { return this->model->roleNames(); }
```

Voici le graphe d'appel pour cette fonction :



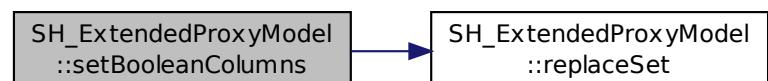
4.19.3.18 void SH_ExtendedProxyModel : :setBooleanColumns (QList< int > *boolCols*) [inherited]

Définition à la ligne 41 du fichier [SH_ExtendedProxyModel.cpp](#).

Références [SH_ExtendedProxyModel : :booleanSet](#), et [SH_ExtendedProxyModel : :replaceSet\(\)](#).

```
00041 {
00042     replaceSet(this->booleanSet, boolCols);
00043 }
```

Voici le graphe d'appel pour cette fonction :



4.19.3.19 bool SH_ExtendedProxyModel : :setData (const QModelIndex & *index*, const QVariant & *value*, int *role* = Qt : :EditRole) [inherited]

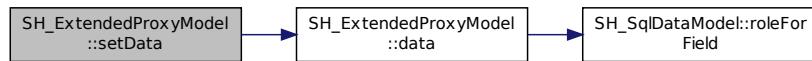
Définition à la ligne 156 du fichier [SH_ExtendedProxyModel.cpp](#).

Références [SH_ExtendedProxyModel : :booleanSet](#), et [SH_ExtendedProxyModel : :data\(\)](#).

```

00157 {
00158     if (!index.isValid())
00159         return false;
00160
00161     if (this->booleanSet.contains(role))
00162     {
00163         QVariant data = (value.toInt() == Qt::Checked) ? QVariant(1) : QVariant(0);
00164         return QSortFilterProxyModel::setData(index, data, role);
00165     }
00166     else
00167     {
00168         return QSortFilterProxyModel::setData(index, value, role);
00169     }
00170 }
00171 }
```

Voici le graphe d'appel pour cette fonction :



4.19.3.20 void SH_ExtendedProxyModel ::setNotNullColumns (QList< int > notNullCols) [inherited]

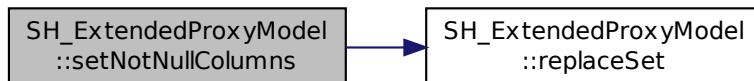
Définition à la ligne 80 du fichier [SH_ExtendedProxyModel.cpp](#).

Références [SH_ExtendedProxyModel ::notNullSet](#), et [SH_ExtendedProxyModel ::replaceSet\(\)](#).

```

00080
00081     if (sourceModel()->inherits("QSqlQueryModel")) {
00082         replaceSet(this->notNullSet, notNullCols);
00083     }
00084 }
```

Voici le graphe d'appel pour cette fonction :



4.19.3.21 void SH_ExtendedProxyModel ::setNullColumns (QList< int > nullCols) [inherited]

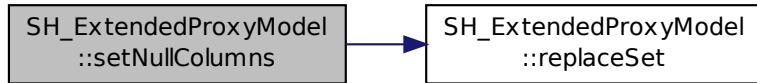
Définition à la ligne 68 du fichier [SH_ExtendedProxyModel.cpp](#).

Références [SH_ExtendedProxyModel ::nullSet](#), et [SH_ExtendedProxyModel ::replaceSet\(\)](#).

```

00068
00069     if (sourceModel()->inherits("QSqlQueryModel")) {
00070         replaceSet(this->nullSet, nullCols);
00071     }
00072 }
```

Voici le graphe d'appel pour cette fonction :



4.19.3.22 void SH_ExtendedProxyModel : :setPasswordColumns (QList< int > passwordCols) [inherited]

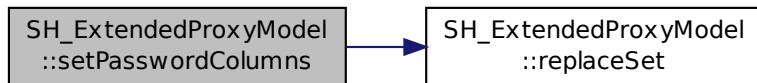
Définition à la ligne 59 du fichier [SH_ExtendedProxyModel.cpp](#).

Références [SH_ExtendedProxyModel : :passwordSet](#), et [SH_ExtendedProxyModel : :replaceSet\(\)](#).

```

00059
00060     replaceSet(this->passwordSet, passwordCols);
00061 }
  
```

Voici le graphe d'appel pour cette fonction :



4.19.3.23 void SH_ExtendedProxyModel : :setReadOnlyColumns (QList< int > readonlyCols) [inherited]

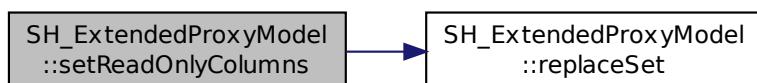
Définition à la ligne 50 du fichier [SH_ExtendedProxyModel.cpp](#).

Références [SH_ExtendedProxyModel : :readonlySet](#), et [SH_ExtendedProxyModel : :replaceSet\(\)](#).

```

00050
00051     replaceSet(this->readonlySet, readonlyCols);
00052 }
  
```

Voici le graphe d'appel pour cette fonction :



4.19.3.24 void SH_ExtendedProxyModel : :setSortKeyColumn (int column) [inherited]

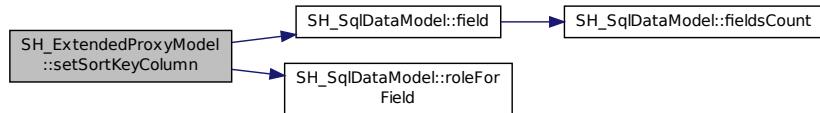
Définition à la ligne 246 du fichier [SH_ExtendedProxyModel.cpp](#).

Références [SH_SqlDataModel : :field\(\)](#), [SH_ExtendedProxyModel : :model](#), [SH_SqlDataModel : :roleForField\(\)](#), [SH_ExtendedProxyModel : :sortChanged\(\)](#), [SH_ExtendedProxyModel : :sortIndex](#), et [SH_SqlDataFields : :sortOrder](#).

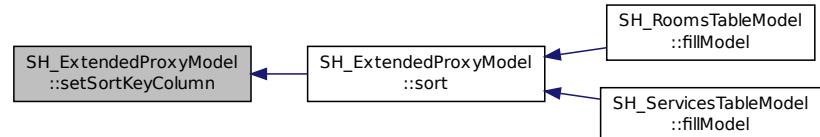
Référencé par [SH_ExtendedProxyModel : :sort\(\)](#).

```
00247 {
00248     this->sortIndex = column;
00249     QSortFilterProxyModel::setSortRole(this->model->roleForField(column));
00250     QSortFilterProxyModel::sort(0, this->model->field(column)->
00251         sortOrder());
00251     emit sortChanged();
00252 }
```

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :



4.19.3.25 void SH_ExtendedProxyModel : :sort (int column, Qt : :SortOrder newOrder = Qt : :AscendingOrder) [inherited]

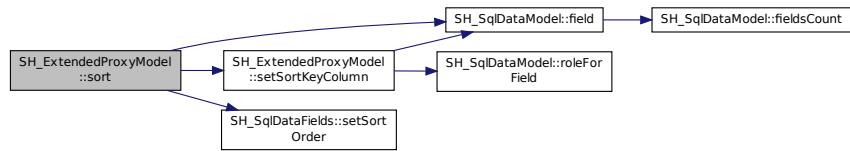
Définition à la ligne 235 du fichier [SH_ExtendedProxyModel.cpp](#).

Références [SH_SqlDataModel : :field\(\)](#), [SH_ExtendedProxyModel : :model](#), [SH_ExtendedProxyModel : :setSortKeyColumn\(\)](#), et [SH_SqlDataFields : :sortOrder\(\)](#).

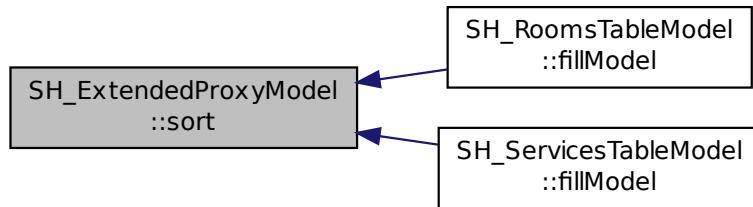
Référencé par [SH_RoomsTableModel : :fillModel\(\)](#), et [SH_ServicesTableModel : :fillModel\(\)](#).

```
00236 {
00237     this->model->field(column)->setSortOrder(newOrder);
00238     SH_ExtendedProxyModel::setSortKeyColumn(column);
00239 }
```

Voici le graphe d'appel pour cette fonction :



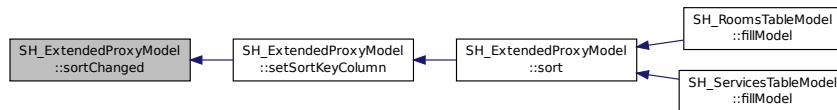
Voici le graphe des appels de cette fonction :



4.19.3.26 void SH_ExtendedProxyModel ::sortChanged() [signal], [inherited]

Référencé par [SH_ExtendedProxyModel ::setSortKeyColumn\(\)](#).

Voici le graphe des appels de cette fonction :



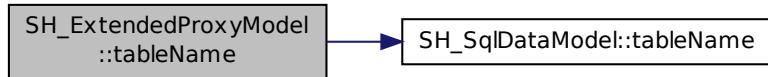
4.19.3.27 const QString SH_ExtendedProxyModel ::tableName() const [inline], [inherited]

Définition à la ligne 45 du fichier [SH_ExtendedSqlProxyModel.h](#).

Références [SH_ExtendedProxyModel ::model](#), et [SH_SqlDataModel ::tableName\(\)](#).

```
00045 { return this->model->tableName(); }
```

Voici le graphe d'appel pour cette fonction :



4.19.4 Documentation des données membres

4.19.4.1 **SH_SqlDataModel* SH_ExtendedProxyModel::model** [protected], [inherited]

Définition à la ligne 241 du fichier [SH_ExtendedSqlProxyModel.h](#).

Référencé par [SH_ExtendedProxyModel::data\(\)](#), [SH_ExtendedProxyModel::fetch\(\)](#), [SH_ExtendedProxyModel::field\(\)](#), [SH_ExtendedProxyModel::fields\(\)](#), [SH_ExtendedProxyModel::fieldsCount\(\)](#), [fillModel\(\)](#), [SH_RoomsTableModel::fillModel\(\)](#), [SH_B bookingsTableModel::fillModel\(\)](#), [SH_ExtendedProxyModel::isEmpty\(\)](#), [SH_ExtendedProxyModel::lastError\(\)](#), [SH_ExtendedProxyModel::roleNames\(\)](#), [SH_ExtendedProxyModel::setSortKeyColumn\(\)](#), [SH_BillingsTableModel\(\)](#), [SH_BillsTableModel::SH_BillsTableModel\(\)](#), [SH_BookingsTableModel::SH_BookingsTableModel\(\)](#), [SH_B bookingsTableModel::SH_B bookingsTableModel\(\)](#), [SH_ClientsTableModel::SH_ClientsTableModel\(\)](#), [SH_ExtendedProxyModel::SH_ExtendedProxyModel\(\)](#), [SH_GroupsTableModel::SH_GroupsTableModel\(\)](#), [SH_RoomsTableModel::SH_RoomsTableModel\(\)](#), [SH_ServicesTableModel::SH_ServicesTableModel\(\)](#), [SH_ExtendedProxyModel::sort\(\)](#), et [SH_ExtendedProxyModel::tableName\(\)](#).

4.19.5 Documentation des propriétés

4.19.5.1 **bool SH_ExtendedProxyModel::empty** [read], [inherited]

Définition à la ligne 21 du fichier [SH_ExtendedSqlProxyModel.h](#).

4.19.5.2 **QString SH_ExtendedProxyModel::fieldsList** [read], [inherited]

Définition à la ligne 18 du fichier [SH_ExtendedSqlProxyModel.h](#).

4.19.5.3 **QString SH_ExtendedProxyModel::lastError** [read], [inherited]

Définition à la ligne 19 du fichier [SH_ExtendedSqlProxyModel.h](#).

4.19.5.4 **int SH_ExtendedProxyModel::sortKeyColumn** [read], [write], [inherited]

Définition à la ligne 20 du fichier [SH_ExtendedSqlProxyModel.h](#).

4.19.5.5 **QString SH_ExtendedProxyModel::table** [read], [inherited]

Définition à la ligne 17 du fichier [SH_ExtendedSqlProxyModel.h](#).

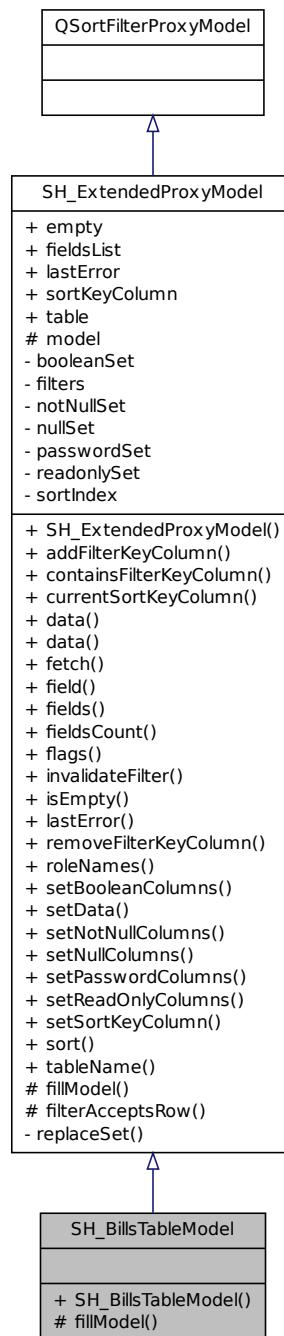
La documentation de cette classe a été générée à partir des fichiers suivants :

- models/[SH_BillingsTableModel.h](#)
- models/[SH_BillingsTableModel.cpp](#)

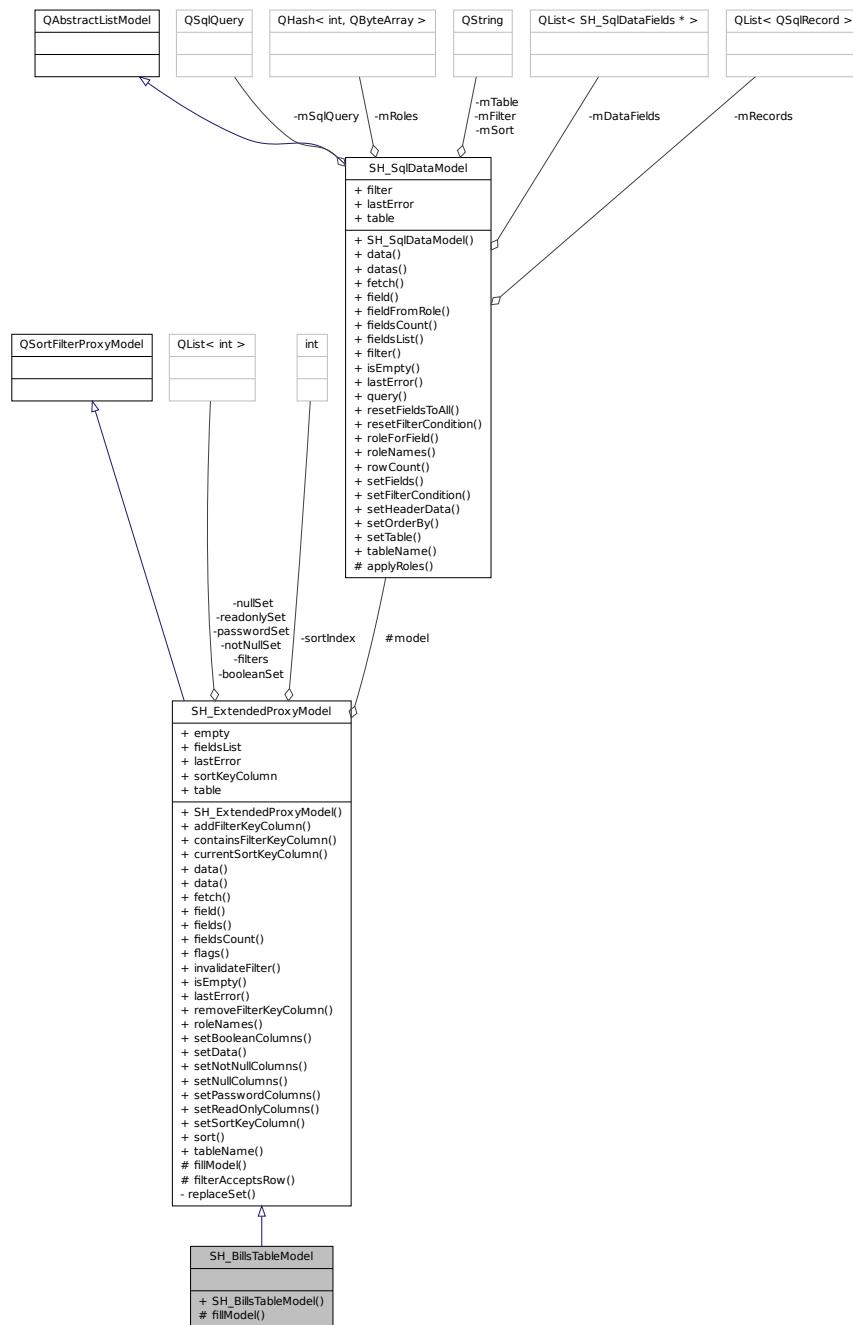
4.20 Référence de la classe SH_BillsTableModel

```
#include <SH_BillsTableModel.h>
```

Graphe d'héritage de SH_BillsTableModel :



Graphe de collaboration de SH_BillsTableModel :



Signaux

- void `sortChanged ()`

Fonctions membres publiques

- `SH_BillsTableModel (QObject *parent=0)`
- `Q_INVOKABLE void addFilterKeyColumn (int column)`
- `Q_INVOKABLE bool containsFilterKeyColumn (int column)`
- `const int currentSortKeyColumn () const`

- Q_INVOKABLE QVariant **data** (int row, int column) const
- Q_INVOKABLE QVariant **data** (const QModelIndex &index, int role=Qt::DisplayRole) const
- Q_INVOKABLE bool **fetch** (QString **tableName**="", QString **filter**="", QString **sort**="", QStringList **fields**=QStringList())
- Q_INVOKABLE SH_SqlDataFields * **field** (int i) const
- const QString **fields** () const
- Q_INVOKABLE int **fieldsCount** () const
- Q_INVOKABLE Qt::ItemFlags **flags** (const QModelIndex &index) const
- void **invalidateFilter** ()
- const bool **isEmpty** () const
- const QString **lastError** () const
- Q_INVOKABLE void **removeFilterKeyColumn** (int column)
- virtual Q_INVOKABLE QHash<int, QByteArray > **roleNames** () const
- void **setBooleanColumns** (QList<int> **boolCols**)
- Q_INVOKABLE bool **setData** (const QModelIndex &index, const QVariant &value, int role=Qt::EditRole)
- void **setNotNullColumns** (QList<int> **notNullCols**)
- void **setNullColumns** (QList<int> **nullCols**)
- void **setPasswordColumns** (QList<int> **passwordCols**)
- void **setReadOnlyColumns** (QList<int> **readonlyCols**)
- void **setSortKeyColumn** (int column)
- Q_INVOKABLE void **sort** (int column, Qt::SortOrder newOrder=Qt::AscendingOrder)
- const QString **tableName** () const

Fonctions membres protégées

- void **fillModel** ()
- bool **filterAcceptsRow** (int source_row, const QModelIndex &source_parent) const

Attributs protégés

- SH_SqlDataModel * **model**

Propriétés

- bool **empty**
- QString **fieldsList**
- QString **lastError**
- int **sortKeyColumn**
- QString **table**

4.20.1 Description détaillée

Définition à la ligne 13 du fichier [SH_BillsTableModel.h](#).

4.20.2 Documentation des constructeurs et destructeur

4.20.2.1 SH_BillsTableModel : :SH_BillsTableModel (QObject * parent = 0)

Paramètres

<i>parent</i>	
---------------	--

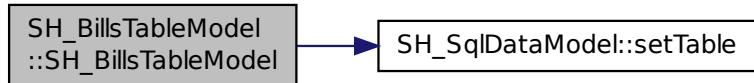
Définition à la ligne 9 du fichier [SH_BillsTableModel.cpp](#).

Références [SH_ExtendedProxyModel](#) : :model, et [SH_SqlDataModel](#) : :setTable().

```

00009
00010     SH_ExtendedProxyModel (parent)
00011 {
00012     SH_ExtendedProxyModel::model->setTable ("BILLS");
00013 }
```

Voici le graphe d'appel pour cette fonction :



4.20.3 Documentation des fonctions membres

4.20.3.1 void SH_ExtendedProxyModel : :addFilterKeyColumn (int column) [inherited]

Définition à la ligne 259 du fichier [SH_ExtendedProxyModel.cpp](#).

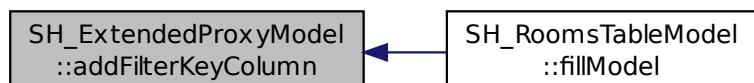
Références [SH_ExtendedProxyModel : :filters](#).

Référencé par [SH_RoomsTableModel : :fillModel\(\)](#).

```

00260 {
00261     this->filters.append\(column\);
00262 }
```

Voici le graphe des appels de cette fonction :



4.20.3.2 bool SH_ExtendedProxyModel : :containsFilterKeyColumn (int column) [inherited]

Définition à la ligne 225 du fichier [SH_ExtendedProxyModel.cpp](#).

Références [SH_ExtendedProxyModel : :filters](#).

```

00226 {
00227     return this->filters.contains\(column\);
00228 }
```

4.20.3.3 const int SH_ExtendedProxyModel : :currentSortKeyColumn () const [inline], [inherited]

Définition à la ligne 38 du fichier [SH_ExtendedSqlProxyModel.h](#).

Références [SH_ExtendedProxyModel : :sortIndex](#).

```
00038 {return this->sortIndex;}
```

4.20.3.4 QVariant SH_ExtendedProxyModel::data(int row, int column) const [inherited]

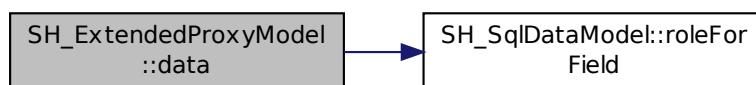
Définition à la ligne 269 du fichier [SH_ExtendedProxyModel.cpp](#).

Références [SH_ExtendedProxyModel::model](#), et [SH_SqlDataModel::roleForField\(\)](#).

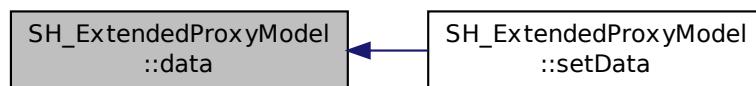
Référencé par [SH_ExtendedProxyModel::setData\(\)](#).

```
00270 {
00271     QModelIndex modelIndex = this->index(row, 0);
00272     return this->data(modelIndex, this->model->roleForField(column));
00273 }
```

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appels de cette fonction :



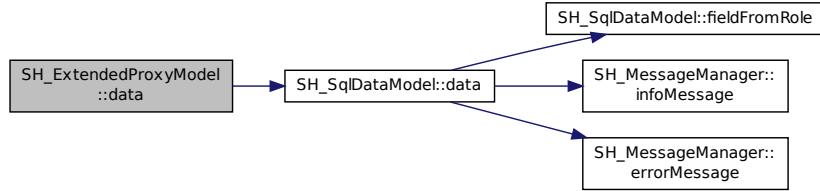
4.20.3.5 QVariant SH_ExtendedProxyModel::data(const QModelIndex & index, int role = Qt::DisplayRole) const [inherited]

Définition à la ligne 127 du fichier [SH_ExtendedProxyModel.cpp](#).

Références [SH_ExtendedProxyModel::booleanSet](#), [SH_SqlDataModel::data\(\)](#), [SH_ExtendedProxyModel::filters](#), [SH_ExtendedProxyModel::model](#), et [SH_ExtendedProxyModel::passwordSet](#).

```
00128 {
00129     if (index.isValid())
00130     {
00131         if (this->booleanSet.contains(role))
00132         {
00133             return index.data(Qt::EditRole).toBool() ? QVariant(Qt::Checked) : QVariant(Qt::Unchecked);
00134         }
00135         else if (this->passwordSet.contains(role))
00136         {
00137             return QVariant("****");
00138         }
00139         else if (!this->filters.contains(role))
00140         {
00141             QModelIndex source_index = QSortFilterProxyModel::mapToSource(index);
00142             if (source_index.isValid())
00143                 return this->model->data(source_index, role);
00144             }
00145         }
00146     }
00147     return QVariant();
00148 }
```

Voici le graphe d'appel pour cette fonction :



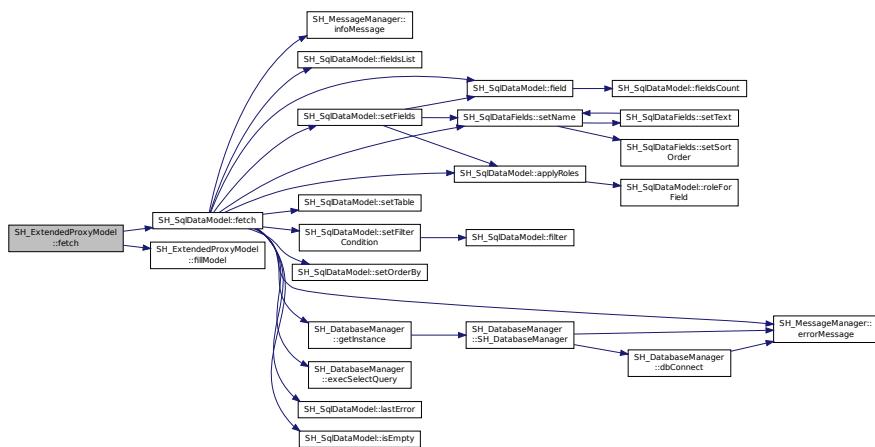
4.20.3.6 bool SH_ExtendedProxyModel ::fetch (QString tableName = "", QString filter = "", QString sort = "", QStringList fields = QStringList ()) [inherited]

Définition à la ligne 280 du fichier [SH_ExtendedProxyModel.cpp](#).

Références `SH_SqlDataModel` : `:fetch()`, `SH_ExtendedProxyModel` : `:fillModel()`, et `SH_ExtendedProxyModel` : `:model`.

```
00281 {  
00282     bool fetched = this->model->fetch(tableName, filter, sort,  
        fields);  
00283     if (fetched)  
00284     {  
00285         this->fillModel();  
00286     }  
00287     this->setSourceModel(this->model);  
00288     return fetched;  
00289 }
```

Voici le graphe d'appel pour cette fonction :



4.20.3.7 **Q_INVOKABLE SH_SqlDataFields* SH_ExtendedProxyModel ::field (int i) const [inline], [inherited]**

Définition à la ligne 82 du fichier [SH_ExtendedSqlProxyModel.h](#).

Références `SH_SqlDataModel` : `:field()`, et `SH_ExtendedProxyModel` : `:model`.

```
00082 { return this->model->field(i); }
```

Voici le graphe d'appel pour cette fonction :



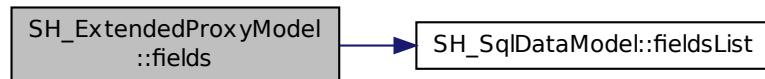
4.20.3.8 const QString SH_ExtendedProxyModel ::fields() const [inline], [inherited]

Définition à la ligne 52 du fichier [SH_ExtendedSqlProxyModel.h](#).

Références [SH_SqlDataModel ::fieldsList\(\)](#), et [SH_ExtendedProxyModel ::model](#).

```
00052 { if(this->model->fieldsList().isEmpty()){ return "*"; } else { return this->model->fieldsList().join(", "); } }
```

Voici le graphe d'appel pour cette fonction :



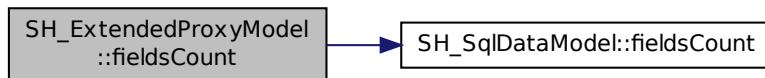
4.20.3.9 Q_INVOKABLE int SH_ExtendedProxyModel ::fieldsCount() const [inline], [inherited]

Définition à la ligne 89 du fichier [SH_ExtendedSqlProxyModel.h](#).

Références [SH_SqlDataModel ::fieldsCount\(\)](#), et [SH_ExtendedProxyModel ::model](#).

```
00089 { return this->model->fieldsCount(); }
```

Voici le graphe d'appel pour cette fonction :



4.20.3.10 void SH_BillsTableModel : :fillModel () [protected], [virtual]

Implémente [SH_ExtendedProxyModel](#).

Définition à la ligne 21 du fichier [SH_BillsTableModel.cpp](#).

```
00022 {
00023 }
```

4.20.3.11 bool SH_ExtendedProxyModel : :filterAcceptsRow (int source_row, const QModelIndex & source_parent) const [protected], [inherited]

Définition à la ligne 92 du fichier [SH_ExtendedProxyModel.cpp](#).

Références [SH_ExtendedProxyModel : :notNullSet](#), et [SH_ExtendedProxyModel : :nullSet](#).

```
00093 {
00094     Q_UNUSED(source_parent);
00095
00096     if (!this->notNullSet.isEmpty())
00097     {
00098         QSqlQueryModel *m = static_cast<QSqlQueryModel *>(sourceModel());
00099         foreach(int column, this->notNullSet)
00100         {
00101             if (m->record(source_row).isNull(column))
00102             {
00103                 return false;
00104             }
00105         }
00106     }
00107
00108     if (!this->nullSet.isEmpty())
00109     {
00110         QSqlQueryModel *m = static_cast<QSqlQueryModel *>(sourceModel());
00111         foreach(int column, this->nullSet)
00112         {
00113             if (!m->record(source_row).isNull(column))
00114             {
00115                 return false;
00116             }
00117         }
00118     }
00119     return true;
00120 }
```

4.20.3.12 Qt : :itemFlags SH_ExtendedProxyModel : :flags (const QModelIndex & index) const [inherited]

Définition à la ligne 179 du fichier [SH_ExtendedProxyModel.cpp](#).

Références [SH_ExtendedProxyModel : :booleanSet](#), et [SH_ExtendedProxyModel : :readonlySet](#).

```
00180 {
00181     if (!index.isValid())
00182     {
00183         return Qt::ItemIsEnabled;
00184     }
00185     if (!this->booleanSet.isEmpty())
00186     {
00187         return Qt::ItemIsUserCheckable | Qt::ItemIsSelectable | Qt::ItemIsEnabled;
00188     }
00189     else if (!this->readonlySet.isEmpty())
00190     {
00191         return Qt::ItemIsSelectable;
00192     }
00193     else
00194     {
00195         return QSortFilterProxyModel::flags(index);
00196     }
00197 }
00198 }
```

4.20.3.13 void SH_ExtendedProxyModel : :invalidateFilter () [inherited]

Définition à la ligne 205 du fichier [SH_ExtendedProxyModel.cpp](#).

Références [SH_ExtendedProxyModel : :filters](#).

```
00206 {
00207     this->filters.clear();
00208 }
```

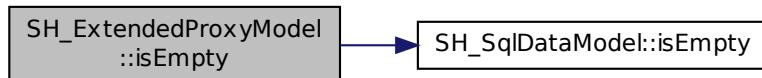
4.20.3.14 const bool SH_ExtendedProxyModel : :isEmpty () const [inline], [inherited]

Définition à la ligne 66 du fichier [SH_ExtendedSqlProxyModel.h](#).

Références [SH_SqlDataModel : :isEmpty\(\)](#), et [SH_ExtendedProxyModel : :model](#).

```
00066 { return this->model->isEmpty(); }
```

Voici le graphe d'appel pour cette fonction :



4.20.3.15 const QString SH_ExtendedProxyModel : :lastError () const [inline], [inherited]

Définition à la ligne 59 du fichier [SH_ExtendedSqlProxyModel.h](#).

Références [SH_SqlDataModel : :lastError](#), et [SH_ExtendedProxyModel : :model](#).

```
00059 { return this->model->lastError(); }
```

4.20.3.16 void SH_ExtendedProxyModel : :removeFilterKeyColumn (int column) [inherited]

Définition à la ligne 215 du fichier [SH_ExtendedProxyModel.cpp](#).

Références [SH_ExtendedProxyModel : :filters](#).

```
00216 {
00217     this->filters.removeAt(this->filters.indexOf(column));
00218 }
```

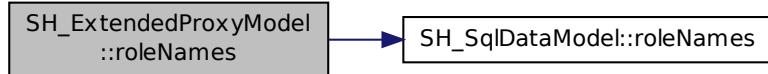
4.20.3.17 virtual Q_INVOKABLE QHash<int, QByteArray> SH_ExtendedProxyModel : :roleNames () const [inline], [virtual], [inherited]

Définition à la ligne 165 du fichier [SH_ExtendedSqlProxyModel.h](#).

Références [SH_ExtendedProxyModel : :model](#), et [SH_SqlDataModel : :roleNames\(\)](#).

```
00165 { return this->model->roleNames(); }
```

Voici le graphe d'appel pour cette fonction :



4.20.3.18 void SH_ExtendedProxyModel ::setBooleanColumns (QList< int > boolCols) [inherited]

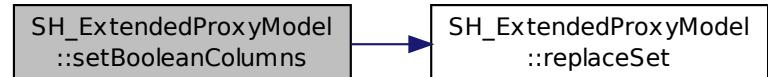
Définition à la ligne 41 du fichier [SH_ExtendedProxyModel.cpp](#).

Références [SH_ExtendedProxyModel ::booleanSet](#), et [SH_ExtendedProxyModel ::replaceSet\(\)](#).

```

00041
00042     replaceSet(this->booleanSet, boolCols);
00043 }
```

Voici le graphe d'appel pour cette fonction :



4.20.3.19 bool SH_ExtendedProxyModel ::setData (const QModelIndex & index, const QVariant & value, int role = Qt :: EditRole) [inherited]

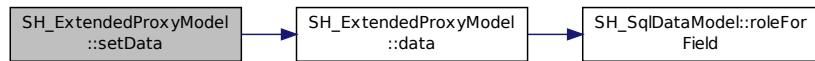
Définition à la ligne 156 du fichier [SH_ExtendedProxyModel.cpp](#).

Références [SH_ExtendedProxyModel ::booleanSet](#), et [SH_ExtendedProxyModel ::data\(\)](#).

```

00157 {
00158     if (!index.isValid())
00159         return false;
00160
00161     if (this->booleanSet.contains(role))
00162     {
00163         QVariant data = (value.toInt() == Qt :: Checked) ? QVariant(1) : QVariant(0);
00164         return QSortFilterProxyModel :: setData(index, data, role);
00165     }
00166     else
00167     {
00168         return QSortFilterProxyModel :: setData(index, value, role);
00169     }
00170
00171 }
```

Voici le graphe d'appel pour cette fonction :



4.20.3.20 void SH_ExtendedProxyModel ::setNotNullColumns (QList< int > notNullCols) [inherited]

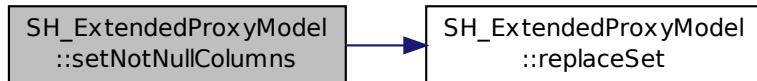
Définition à la ligne 80 du fichier [SH_ExtendedProxyModel.cpp](#).

Références [SH_ExtendedProxyModel ::notNullSet](#), et [SH_ExtendedProxyModel ::replaceSet\(\)](#).

```

00080
00081     if (sourceModel()->inherits("QSqlQueryModel")) {
00082         replaceSet(this->notNullSet, notNullCols);
00083     }
00084 }
```

Voici le graphe d'appel pour cette fonction :



4.20.3.21 void SH_ExtendedProxyModel ::setNullColumns (QList< int > nullCols) [inherited]

Définition à la ligne 68 du fichier [SH_ExtendedProxyModel.cpp](#).

Références [SH_ExtendedProxyModel ::nullSet](#), et [SH_ExtendedProxyModel ::replaceSet\(\)](#).

```

00068
00069     if (sourceModel()->inherits("QSqlQueryModel")) {
00070         replaceSet(this->nullSet, nullCols);
00071     }
00072 }
```

Voici le graphe d'appel pour cette fonction :



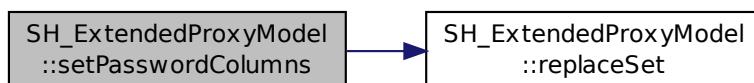
4.20.3.22 void SH_ExtendedProxyModel ::setPasswordColumns (QList< int > *passwordCols*) [inherited]

Définition à la ligne 59 du fichier [SH_ExtendedProxyModel.cpp](#).

Références [SH_ExtendedProxyModel ::passwordSet](#), et [SH_ExtendedProxyModel ::replaceSet\(\)](#).

```
00059
00060     replaceSet(this->passwordSet, passwordCols);
00061 }
```

Voici le graphe d'appel pour cette fonction :



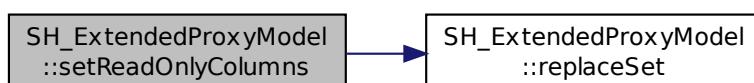
4.20.3.23 void SH_ExtendedProxyModel ::setReadOnlyColumns (QList< int > *readonlyCols*) [inherited]

Définition à la ligne 50 du fichier [SH_ExtendedProxyModel.cpp](#).

Références [SH_ExtendedProxyModel ::readonlySet](#), et [SH_ExtendedProxyModel ::replaceSet\(\)](#).

```
00050
00051     replaceSet(this->readonlySet, readonlyCols);
00052 }
```

Voici le graphe d'appel pour cette fonction :



4.20.3.24 void SH_ExtendedProxyModel ::setSortKeyColumn (int *column*) [inherited]

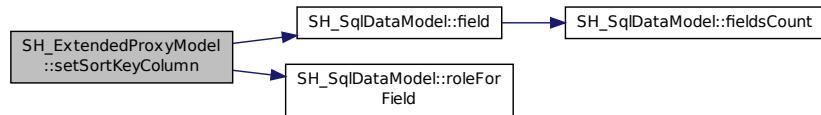
Définition à la ligne 246 du fichier [SH_ExtendedProxyModel.cpp](#).

Références [SH_SqlDataModel ::field\(\)](#), [SH_ExtendedProxyModel ::model](#), [SH_SqlDataModel ::roleForField\(\)](#), [SH_ExtendedProxyModel ::sortChanged\(\)](#), [SH_ExtendedProxyModel ::sortIndex](#), et [SH_SqlDataFields ::sortOrder](#).

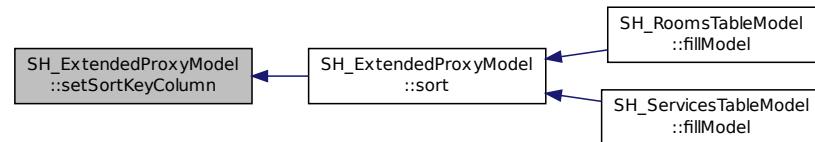
Référencé par [SH_ExtendedProxyModel ::sort\(\)](#).

```
00247 {
00248     this->sortIndex = column;
00249     QSortFilterProxyModel::setSortRole(this->model->roleForField(column));
00250     QSortFilterProxyModel::sort(0, this->model->field(column)->
00251         sortOrder());
00251     emit sortChanged();
00252 }
```

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appels de cette fonction :



4.20.3.25 void SH_ExtendedProxyModel ::sort (int column, Qt ::SortOrder newOrder = Qt ::AscendingOrder) [inherited]

Définition à la ligne 235 du fichier [SH_ExtendedProxyModel.cpp](#).

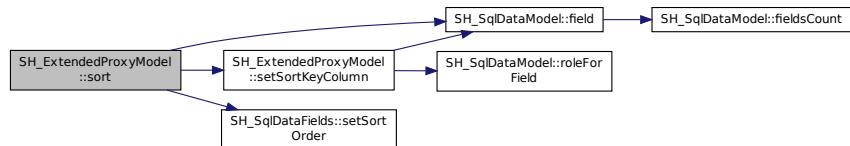
Références [SH_SqlDataModel ::field\(\)](#), [SH_ExtendedProxyModel ::model](#), [SH_ExtendedProxyModel ::setSortKeyColumn\(\)](#), et [SH_SqlDataFields ::setSortOrder\(\)](#).

Référencé par [SH_RoomsTableModel ::fillModel\(\)](#), et [SH_ServicesTableModel ::fillModel\(\)](#).

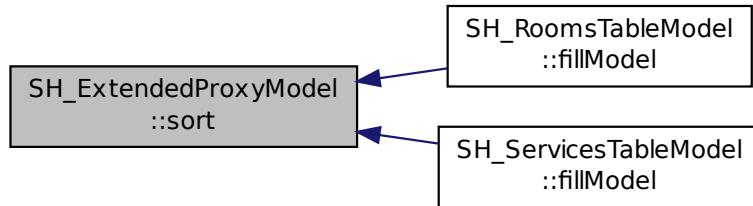
```

00236 {
00237     this->model->field(column)->setSortOrder(newOrder);
00238     SH_ExtendedProxyModel::setSortKeyColumn(column);
00239 }
  
```

Voici le graphe d'appel pour cette fonction :



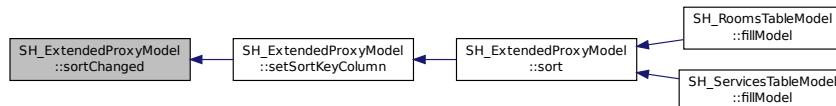
Voici le graphe des appelants de cette fonction :



4.20.3.26 void SH_ExtendedProxyModel ::sortChanged() [signal], [inherited]

Référencé par [SH_ExtendedProxyModel ::setSortKeyColumn\(\)](#).

Voici le graphe des appelants de cette fonction :



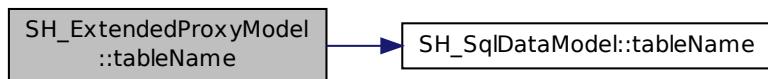
4.20.3.27 const QString SH_ExtendedProxyModel ::tableName() const [inline], [inherited]

Définition à la ligne 45 du fichier [SH_ExtendedSqlProxyModel.h](#).

Références [SH_ExtendedProxyModel ::model](#), et [SH_SqlDataModel ::tableName\(\)](#).

```
00045 { return this->model->tableName(); }
```

Voici le graphe d'appel pour cette fonction :



4.20.4 Documentation des données membres

4.20.4.1 SH_SqlDataModel* SH_ExtendedProxyModel ::model [protected], [inherited]

Définition à la ligne 241 du fichier [SH_ExtendedSqlProxyModel.h](#).

Référencé par [SH_ExtendedProxyModel](#) : `:data()`, [SH_ExtendedProxyModel](#) : `:fetch()`, [SH_ExtendedProxyModel](#) : `:field()`, [SH_ExtendedProxyModel](#) : `:fields()`, [SH_ExtendedProxyModel](#) : `:fieldsCount()`, [SH_BillingsTableModel](#) : `:fillModel()`, [SH_RoomsTableModel](#) : `:fillModel()`, [SH_BookingsTableModel](#) : `:fillModel()`, [SH_ExtendedProxyModel](#) : `:isEmpty()`, [SH_ExtendedProxyModel](#) : `:lastError()`, [SH_ExtendedProxyModel](#) : `:roleNames()`, [SH_ExtendedProxyModel](#) : `:setSortKeyColumn()`, [SH_BillingsTableModel](#) : `:SH_BillingsTableModel()`, [SH_BillsTableModel](#) : `:SH_BillingsTableModel()`, [SH_BookingsTableModel](#) : `:SH_BookingsTableModel()`, [SH_ClientsTableModel](#) : `:SH_ClientsTableModel()`, [SH_ExtendedProxyModel](#) : `:SH_ExtendedProxyModel()`, [SH_GroupsTableModel](#) : `:SH_GroupsTableModel()`, [SH_RoomsTableModel](#) : `:SH_RoomsTableModel()`, [SH_ServicesTableModel](#) : `:SH_ServicesTableModel()`, [SH_ExtendedProxyModel](#) : `:sort()`, et [SH_ExtendedProxyModel](#) : `:tableName()`.

4.20.5 Documentation des propriétés

4.20.5.1 `bool SH_ExtendedProxyModel::empty [read], [inherited]`

Définition à la ligne 21 du fichier [SH_ExtendedSqlProxyModel.h](#).

4.20.5.2 `QString SH_ExtendedProxyModel::fieldsList [read], [inherited]`

Définition à la ligne 18 du fichier [SH_ExtendedSqlProxyModel.h](#).

4.20.5.3 `QString SH_ExtendedProxyModel::lastError [read], [inherited]`

Définition à la ligne 19 du fichier [SH_ExtendedSqlProxyModel.h](#).

4.20.5.4 `int SH_ExtendedProxyModel::sortKeyColumn [read], [write], [inherited]`

Définition à la ligne 20 du fichier [SH_ExtendedSqlProxyModel.h](#).

4.20.5.5 `QString SH_ExtendedProxyModel::table [read], [inherited]`

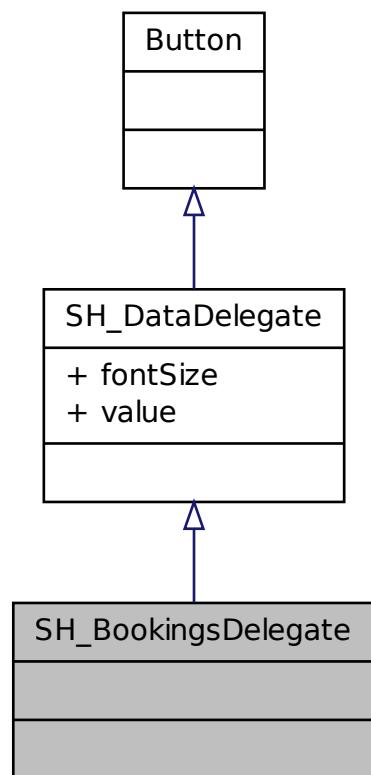
Définition à la ligne 17 du fichier [SH_ExtendedSqlProxyModel.h](#).

La documentation de cette classe a été générée à partir des fichiers suivants :

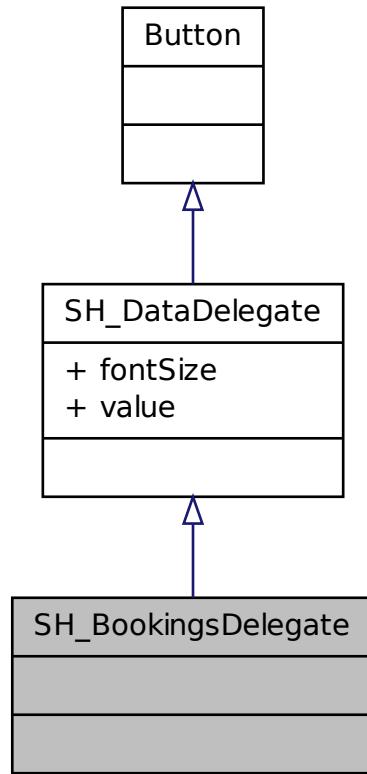
- [models/SH_BillsTableModel.h](#)
- [models/SH_BillsTableModel.cpp](#)

4.21 Référence de la classe SH_BookingsDelegate

Graphe d'héritage de SH_BookingsDelegate :



Graphe de collaboration de SH_BookingsDelegate :



Propriétés

- int `fontSize`
- string `value`

4.21.1 Description détaillée

Définition à la ligne 4 du fichier [SH_BookingsDelegate.qml](#).

4.21.2 Documentation des propriétés

4.21.2.1 int SH_DataDelegate ::fontSize [inherited]

Définition à la ligne 9 du fichier [SH_DataDelegate.qml](#).

4.21.2.2 string SH_DataDelegate ::value [inherited]

Définition à la ligne 7 du fichier [SH_DataDelegate.qml](#).

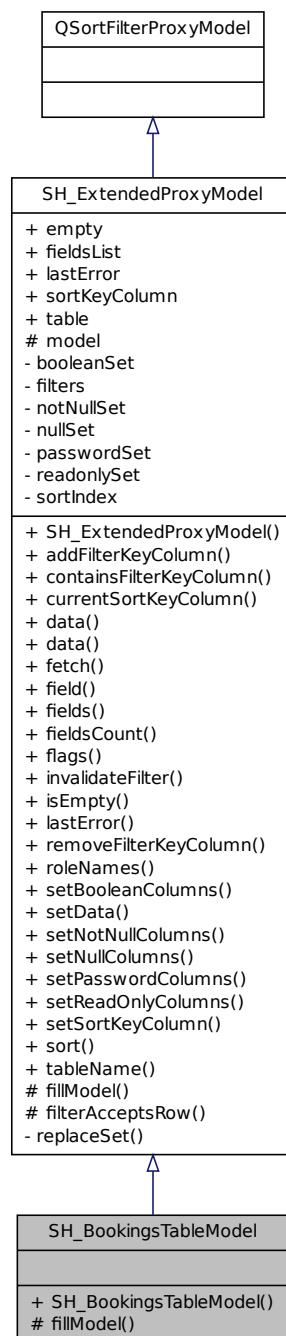
La documentation de cette classe a été générée à partir du fichier suivant :

- [views/qml/SH_BookingsDelegate.qml](#)

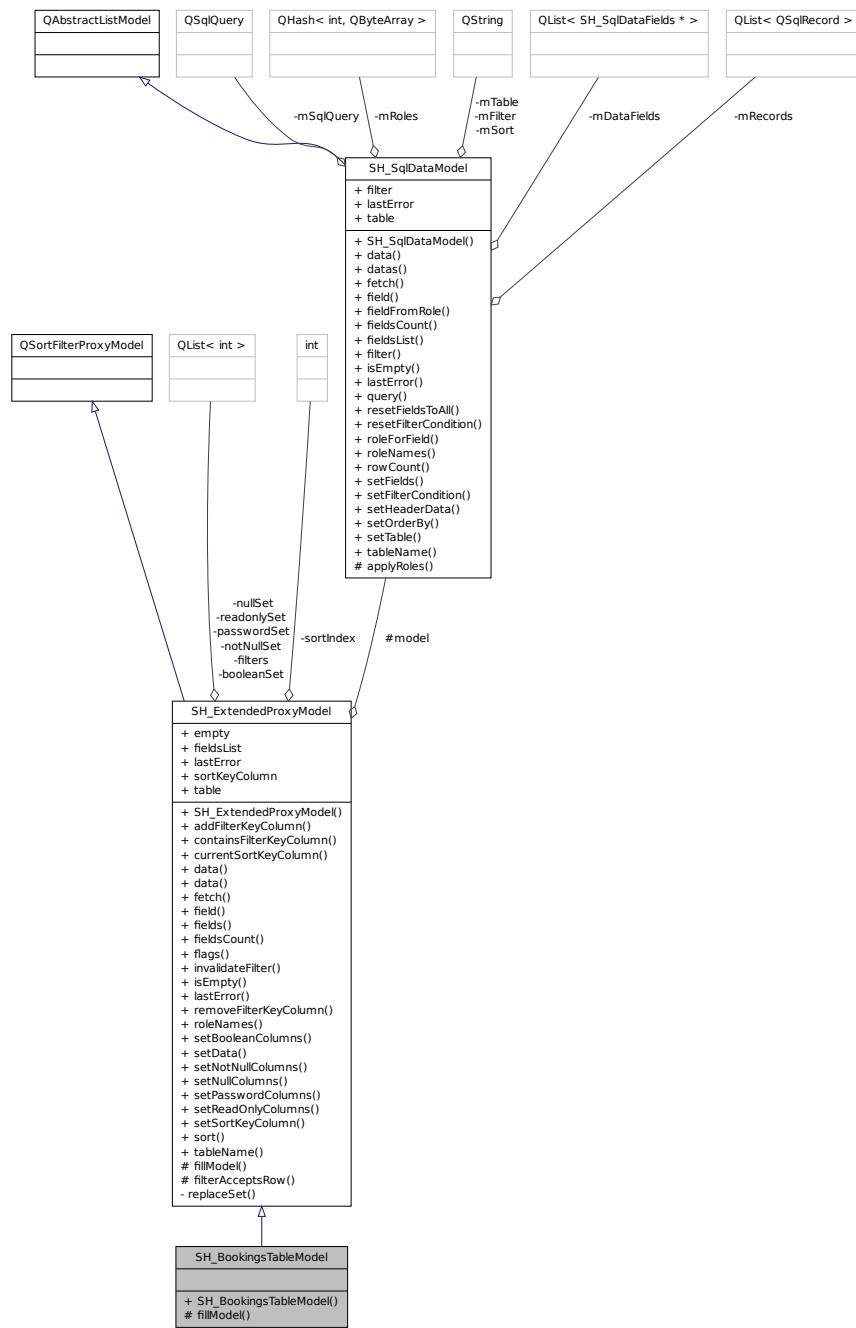
4.22 Référence de la classe SH_BookingsTableModel

```
#include <SH_BookingsTableModel.h>
```

Graphe d'héritage de SH_BookingsTableModel :



Graphe de collaboration de SH_BookingsTableModel :



Signaux

- void `sortChanged ()`

Fonctions membres publiques

- `SH_BookingsTableModel (QObject *parent=0)`
- `Q_INVOKABLE void addFilterKeyColumn (int column)`
- `Q_INVOKABLE bool containsFilterKeyColumn (int column)`
- `const int currentSortKeyColumn () const`

- Q_INVOKABLE QVariant **data** (int row, int column) const
- Q_INVOKABLE QVariant **data** (const QModelIndex &index, int role=Qt::DisplayRole) const
- Q_INVOKABLE bool **fetch** (QString **tableName**="", QString **filter**="", QString **sort**="", QStringList **fields**=QStringList())
- Q_INVOKABLE SH_SQLDataFields * **field** (int i) const
- const QString **fields** () const
- Q_INVOKABLE int **fieldsCount** () const
- Q_INVOKABLE Qt::ItemFlags **flags** (const QModelIndex &index) const
- void **invalidateFilter** ()
- const bool **isEmpty** () const
- const QString **lastError** () const
- Q_INVOKABLE void **removeFilterKeyColumn** (int column)
- virtual Q_INVOKABLE QHash<int, QByteArray> **roleNames** () const
- void **setBooleanColumns** (QList<int> **boolCols**)
- Q_INVOKABLE bool **setData** (const QModelIndex &index, const QVariant &value, int role=Qt::EditRole)
- void **setNotNullColumns** (QList<int> **notNullCols**)
- void **setNullColumns** (QList<int> **nullCols**)
- void **setPasswordColumns** (QList<int> **passwordCols**)
- void **setReadOnlyColumns** (QList<int> **readonlyCols**)
- void **setSortKeyColumn** (int column)
- Q_INVOKABLE void **sort** (int column, Qt::SortOrder newOrder=Qt::AscendingOrder)
- const QString **tableName** () const

Fonctions membres protégées

- void **fillModel** ()
- bool **filterAcceptsRow** (int source_row, const QModelIndex &source_parent) const

Attributs protégés

- SH_SQLDataModel * **model**

Propriétés

- bool **empty**
- QString **fieldsList**
- QString **lastError**
- int **sortKeyColumn**
- QString **table**

4.22.1 Description détaillée

Définition à la ligne 14 du fichier [SH_BookingsTableModel.h](#).

4.22.2 Documentation des constructeurs et destructeur

4.22.2.1 SH_BookingsTableModel : :SH_BookingsTableModel (QObject * *parent* = 0)

Paramètres

<i>parent</i>	
---------------	--

Définition à la ligne 9 du fichier [SH_BookingsTableModel.cpp](#).

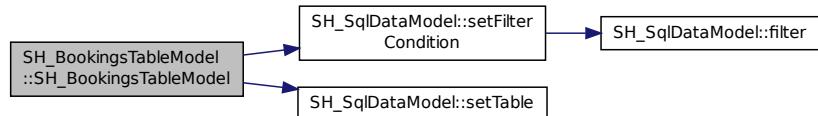
Références [SH_ExtendedProxyModel](#) : :model, [SH_SQLDataModel](#) : :setFilterCondition(), et [SH_SQLDataModel](#) : :setTable().

```

00009
00010     SH_ExtendedProxyModel (parent)
00011 {
00012     SH_ExtendedProxyModel::model->setTable ("BOOKINGS");
00013     SH_ExtendedProxyModel::model->setFilterCondition (
00014         QObject::tr ("ISCONFIRMED") + "='1'" );

```

Voici le graphe d'appel pour cette fonction :



4.22.3 Documentation des fonctions membres

4.22.3.1 void SH_ExtendedProxyModel ::addFilterKeyColumn (int column) [inherited]

Définition à la ligne 259 du fichier [SH_ExtendedProxyModel.cpp](#).

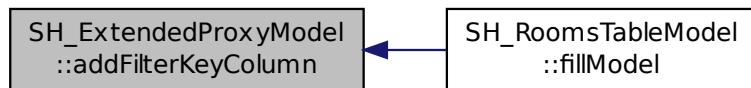
Références [SH_ExtendedProxyModel ::filters](#).

Référencé par [SH_RoomsTableModel ::fillModel\(\)](#).

```

00260 {
00261     this->filters.append(column);
00262 }
```

Voici le graphe des appels de cette fonction :



4.22.3.2 bool SH_ExtendedProxyModel ::containsFilterKeyColumn (int column) [inherited]

Définition à la ligne 225 du fichier [SH_ExtendedProxyModel.cpp](#).

Références [SH_ExtendedProxyModel ::filters](#).

```

00226 {
00227     return this->filters.contains(column);
00228 }
```

4.22.3.3 const int SH_ExtendedProxyModel ::currentSortKeyColumn () const [inline], [inherited]

Définition à la ligne 38 du fichier [SH_ExtendedSqlProxyModel.h](#).

Références [SH_ExtendedProxyModel ::sortIndex](#).

```
00038 {return this->sortIndex;}
```

4.22.3.4 QVariant SH_ExtendedProxyModel ::data (int row, int column) const [inherited]

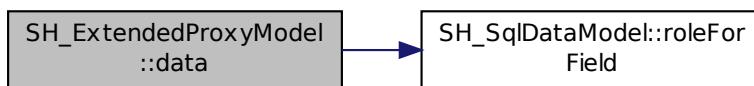
Définition à la ligne 269 du fichier [SH_ExtendedProxyModel.cpp](#).

Références [SH_ExtendedProxyModel ::model](#), et [SH_SqlDataModel ::roleForField\(\)](#).

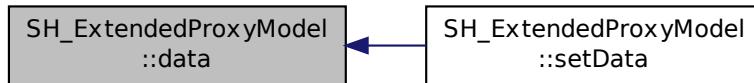
Référencé par [SH_ExtendedProxyModel ::setData\(\)](#).

```
00270 {
00271     QModelIndex modelIndex = this->index(row, 0);
00272     return this->data(modelIndex, this->model->roleForField(column));
00273 }
```

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appels de cette fonction :



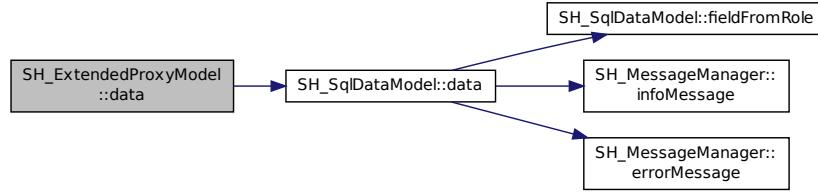
4.22.3.5 QVariant SH_ExtendedProxyModel ::data (const QModelIndex & index, int role = Qt ::DisplayRole) const [inherited]

Définition à la ligne 127 du fichier [SH_ExtendedProxyModel.cpp](#).

Références [SH_ExtendedProxyModel ::booleanSet](#), [SH_SqlDataModel ::data\(\)](#), [SH_ExtendedProxyModel ::filters](#), [SH_ExtendedProxyModel ::model](#), et [SH_ExtendedProxyModel ::passwordSet](#).

```
00128 {
00129     if (index.isValid())
00130     {
00131         if (this->booleanSet.contains(role))
00132         {
00133             return index.data(Qt::EditRole).toBool() ? QVariant(Qt::Checked) : QVariant(Qt::Unchecked);
00134         }
00135         else if (this->passwordSet.contains(role))
00136         {
00137             return QVariant("****");
00138         }
00139         else if (!this->filters.contains(role))
00140         {
00141             QModelIndex source_index = QSortFilterProxyModel::mapToSource(index);
00142             if (source_index.isValid())
00143                 return this->model->data(source_index, role);
00144             }
00145         }
00146     }
00147     return QVariant();
00148 }
```

Voici le graphe d'appel pour cette fonction :



4.22.3.6 bool SH_ExtendedProxyModel::fetch (QString tableName = "", QString filter = "", QString sort = "", QStringList fields = QStringList ()) [inherited]

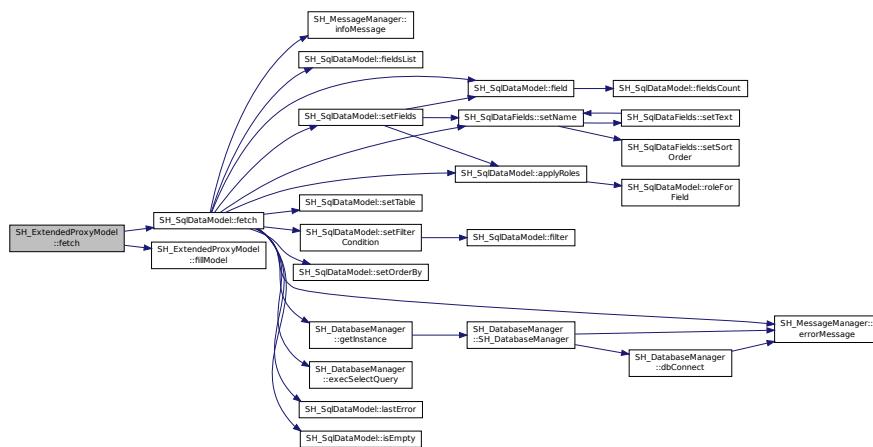
Définition à la ligne 280 du fichier [SH_ExtendedProxyModel.cpp](#).

Références [SH_SqlDataModel::fetch\(\)](#), [SH_ExtendedProxyModel::fillModel\(\)](#), et [SH_ExtendedProxyModel::model](#).

```

00281 {
00282     bool fetched = this->model->fetch(tableName, filter, sort,
00283         fields);
00284     if (fetched)
00285         this->fillModel();
00286     this->setSourceModel(this->model);
00287     return fetched;
00288 }
00289 }
```

Voici le graphe d'appel pour cette fonction :



4.22.3.7 Q_INVOKABLE SH_SqlDataFields* SH_ExtendedProxyModel::field (int i) const [inline], [inherited]

Définition à la ligne 82 du fichier [SH_ExtendedSqlProxyModel.h](#).

Références [SH_SqlDataModel::field\(\)](#), et [SH_ExtendedProxyModel::model](#).

```
00082 { return this->model->field(i); }
```

Voici le graphe d'appel pour cette fonction :



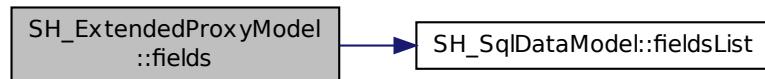
4.22.3.8 const QString SH_ExtendedProxyModel ::fields() const [inline], [inherited]

Définition à la ligne 52 du fichier [SH_ExtendedSqlProxyModel.h](#).

Références [SH_SqlDataModel ::fieldsList\(\)](#), et [SH_ExtendedProxyModel ::model](#).

```
00052 { if(this->model->fieldsList().isEmpty()){ return "*"; } else { return this->model->fieldsList().join(", "); } }
```

Voici le graphe d'appel pour cette fonction :



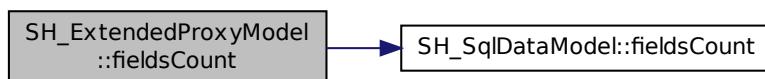
4.22.3.9 Q_INVOKABLE int SH_ExtendedProxyModel ::fieldsCount() const [inline], [inherited]

Définition à la ligne 89 du fichier [SH_ExtendedSqlProxyModel.h](#).

Références [SH_SqlDataModel ::fieldsCount\(\)](#), et [SH_ExtendedProxyModel ::model](#).

```
00089 { return this->model->fieldsCount(); }
```

Voici le graphe d'appel pour cette fonction :



4.22.3.10 void SH_BookingsTableModel : :fillModel() [protected], [virtual]

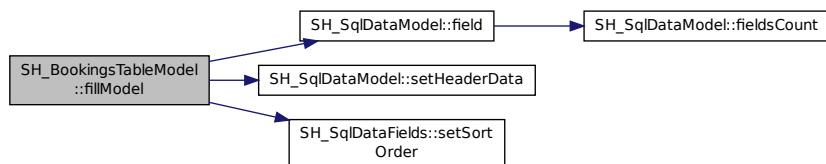
Implémente [SH_ExtendedProxyModel](#).

Définition à la ligne 22 du fichier [SH_BookingsTableModel.cpp](#).

Références [SH_SqlDataModel : :field\(\)](#), [SH_ExtendedProxyModel : :model](#), [SH_SqlDataModel : :setHeaderData\(\)](#), et [SH_SqlDataFields : :setSortOrder\(\)](#).

```
00023 {
00024     SH_ExtendedProxyModel::model->setHeaderData(0, Qt::Horizontal,
00025     QObject::tr("Date réservation"));
00026     SH_ExtendedProxyModel::model->setHeaderData(1, Qt::Horizontal,
00027     QObject::tr("Nom client"));
00028     SH_ExtendedProxyModel::model->setHeaderData(2, Qt::Horizontal,
00029     QObject::tr("Date arrivée prévue"));
00030     SH_ExtendedProxyModel::model->setHeaderData(3, Qt::Horizontal,
00031     QObject::tr("Nb Personnes"));
00032     SH_ExtendedProxyModel::model->field(0)->
00033         setSortOrder(Qt::AscendingOrder);
00034 }
```

Voici le graphique d'appel pour cette fonction :



4.22.3.11 bool SH_ExtendedProxyModel : :filterAcceptsRow (int source_row, const QModelIndex & source_parent) const [protected], [inherited]

Définition à la ligne 92 du fichier [SH_ExtendedProxyModel.cpp](#).

Références [SH_ExtendedProxyModel : :notNullSet](#), et [SH_ExtendedProxyModel : :nullSet](#).

```
00093 {
00094     Q_UNUSED(source_parent);
00095
00096     if (!this->notNullSet.isEmpty())
00097     {
00098         QSqlQueryModel *m = static_cast<QSqlQueryModel *>(sourceModel());
00099         foreach(int column, this->notNullSet)
00100         {
00101             if (m->record(source_row).isNull(column))
00102             {
00103                 return false;
00104             }
00105         }
00106     }
00107
00108     if (!this->nullSet.isEmpty())
00109     {
00110         QSqlQueryModel *m = static_cast<QSqlQueryModel *>(sourceModel());
00111         foreach(int column, this->nullSet)
00112         {
00113             if (!m->record(source_row).isNull(column))
00114             {
00115                 return false;
00116             }
00117         }
00118     }
00119     return true;
00120 }
```

4.22.3.12 Qt ::itemFlags SH_ExtendedProxyModel ::flags (const QModelIndex & index) const [inherited]

Définition à la ligne 179 du fichier [SH_ExtendedProxyModel.cpp](#).

Références [SH_ExtendedProxyModel ::booleanSet](#), et [SH_ExtendedProxyModel ::readonlySet](#).

```
00180 {
00181     if (!index.isValid())
00182     {
00183         return Qt::ItemIsEnabled;
00184     }
00185     if (!this->booleanSet.isEmpty())
00186     {
00187         return Qt::ItemIsUserCheckable | Qt::ItemIsSelectable | Qt::ItemIsEnabled;
00188     }
00189     else if (!this->readonlySet.isEmpty())
00190     {
00191         return Qt::ItemIsSelectable;
00192     }
00193     else
00194     {
00195         return QSortFilterProxyModel::flags(index);
00196     }
00197 }
00198 }
```

4.22.3.13 void SH_ExtendedProxyModel ::invalidateFilter () [inherited]

Définition à la ligne 205 du fichier [SH_ExtendedProxyModel.cpp](#).

Références [SH_ExtendedProxyModel ::filters](#).

```
00206 {
00207     this->filters.clear();
00208 }
```

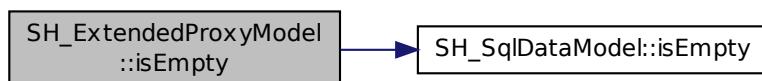
4.22.3.14 const bool SH_ExtendedProxyModel ::isEmpty () const [inline], [inherited]

Définition à la ligne 66 du fichier [SH_ExtendedSqlProxyModel.h](#).

Références [SH_SqlDataManager ::isEmpty\(\)](#), et [SH_ExtendedProxyModel ::model](#).

```
00066 { return this->model->isEmpty(); }
```

Voici le graphe d'appel pour cette fonction :



4.22.3.15 const QString SH_ExtendedProxyModel ::lastError () const [inline], [inherited]

Définition à la ligne 59 du fichier [SH_ExtendedSqlProxyModel.h](#).

Références [SH_SqlDataManager ::lastError](#), et [SH_ExtendedProxyModel ::model](#).

```
00059 { return this->model->lastError(); }
```

4.22.3.16 void SH_ExtendedProxyModel : :removeFilterKeyColumn (int *column*) [inherited]

Définition à la ligne 215 du fichier [SH_ExtendedProxyModel.cpp](#).

Références [SH_ExtendedProxyModel : :filters](#).

```
00216 {
00217     this->filters.removeAt(this->filters.indexOf(column));
00218 }
```

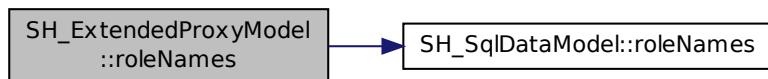
4.22.3.17 virtual Q_INVOKABLE QHash<int, QByteArray> SH_ExtendedProxyModel : :roleNames () const [inline], [virtual], [inherited]

Définition à la ligne 165 du fichier [SH_ExtendedSqlProxyModel.h](#).

Références [SH_ExtendedProxyModel : :model](#), et [SH_SqDataModel : :roleNames\(\)](#).

```
00165 { return this->model->roleNames(); }
```

Voici le graphe d'appel pour cette fonction :



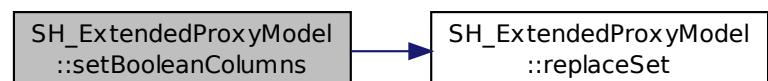
4.22.3.18 void SH_ExtendedProxyModel : :setBooleanColumns (QList< int > *boolCols*) [inherited]

Définition à la ligne 41 du fichier [SH_ExtendedProxyModel.cpp](#).

Références [SH_ExtendedProxyModel : :booleanSet](#), et [SH_ExtendedProxyModel : :replaceSet\(\)](#).

```
00041 {
00042     replaceSet(this->booleanSet, boolCols);
00043 }
```

Voici le graphe d'appel pour cette fonction :



4.22.3.19 bool SH_ExtendedProxyModel : :setData (const QModelIndex & *index*, const QVariant & *value*, int *role* = Qt : :EditRole) [inherited]

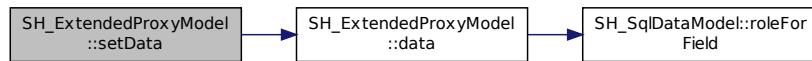
Définition à la ligne 156 du fichier [SH_ExtendedProxyModel.cpp](#).

Références [SH_ExtendedProxyModel : :booleanSet](#), et [SH_ExtendedProxyModel : :data\(\)](#).

```

00157 {
00158     if (!index.isValid())
00159         return false;
00160
00161     if (this->booleanSet.contains(role))
00162     {
00163         QVariant data = (value.toInt() == Qt::Checked) ? QVariant(1) : QVariant(0);
00164         return QSortFilterProxyModel::setData(index, data, role);
00165     }
00166     else
00167     {
00168         return QSortFilterProxyModel::setData(index, value, role);
00169     }
00170 }
00171 }
```

Voici le graphe d'appel pour cette fonction :



4.22.3.20 void SH_ExtendedProxyModel ::setNotNullColumns (QList< int > notNullCols) [inherited]

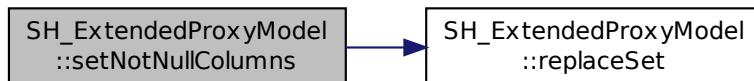
Définition à la ligne 80 du fichier [SH_ExtendedProxyModel.cpp](#).

Références [SH_ExtendedProxyModel ::notNullSet](#), et [SH_ExtendedProxyModel ::replaceSet\(\)](#).

```

00080
00081     if (sourceModel()->inherits("QSqlQueryModel")) {
00082         replaceSet(this->notNullSet, notNullCols);
00083     }
00084 }
```

Voici le graphe d'appel pour cette fonction :



4.22.3.21 void SH_ExtendedProxyModel ::setNullColumns (QList< int > nullCols) [inherited]

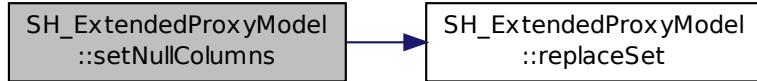
Définition à la ligne 68 du fichier [SH_ExtendedProxyModel.cpp](#).

Références [SH_ExtendedProxyModel ::nullSet](#), et [SH_ExtendedProxyModel ::replaceSet\(\)](#).

```

00068
00069     if (sourceModel()->inherits("QSqlQueryModel")) {
00070         replaceSet(this->nullSet, nullCols);
00071     }
00072 }
```

Voici le graphe d'appel pour cette fonction :



4.22.3.22 void SH_ExtendedProxyModel : :setPasswordColumns (QList< int > passwordCols) [inherited]

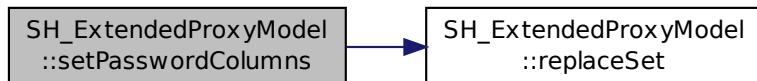
Définition à la ligne 59 du fichier [SH_ExtendedProxyModel.cpp](#).

Références [SH_ExtendedProxyModel : :passwordSet](#), et [SH_ExtendedProxyModel : :replaceSet\(\)](#).

```

00059
00060     replaceSet(this->passwordSet, passwordCols);
00061 }
  
```

Voici le graphe d'appel pour cette fonction :



4.22.3.23 void SH_ExtendedProxyModel : :setReadOnlyColumns (QList< int > readonlyCols) [inherited]

Définition à la ligne 50 du fichier [SH_ExtendedProxyModel.cpp](#).

Références [SH_ExtendedProxyModel : :readonlySet](#), et [SH_ExtendedProxyModel : :replaceSet\(\)](#).

```

00050
00051     replaceSet(this->readonlySet, readonlyCols);
00052 }
  
```

Voici le graphe d'appel pour cette fonction :



4.22.3.24 void SH_ExtendedProxyModel : :setSortKeyColumn (int column) [inherited]

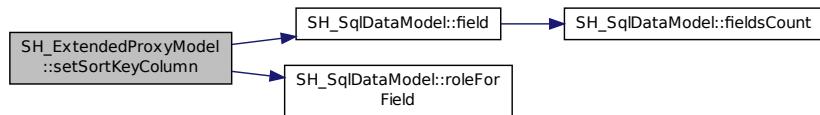
Définition à la ligne 246 du fichier [SH_ExtendedProxyModel.cpp](#).

Références [SH_SqlDataModel : :field\(\)](#), [SH_ExtendedProxyModel : :model](#), [SH_SqlDataModel : :roleForField\(\)](#), [SH_ExtendedProxyModel : :sortChanged\(\)](#), [SH_ExtendedProxyModel : :sortIndex](#), et [SH_SqlDataFields : :sortOrder](#).

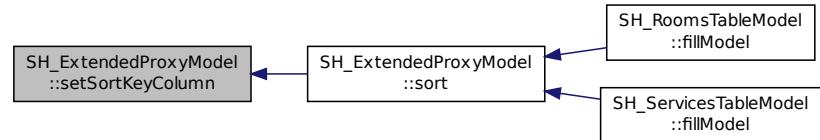
Référencé par [SH_ExtendedProxyModel : :sort\(\)](#).

```
00247 {
00248     this->sortIndex = column;
00249     QSortFilterProxyModel::setSortRole(this->model->roleForField(column));
00250     QSortFilterProxyModel::sort(0, this->model->field(column)->
00251         sortOrder());
00251     emit sortChanged();
00252 }
```

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :



4.22.3.25 void SH_ExtendedProxyModel : :sort (int column, Qt : :SortOrder newOrder = Qt : :AscendingOrder) [inherited]

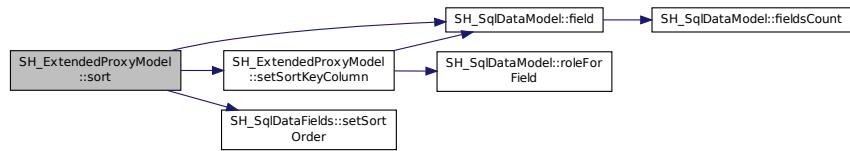
Définition à la ligne 235 du fichier [SH_ExtendedProxyModel.cpp](#).

Références [SH_SqlDataModel : :field\(\)](#), [SH_ExtendedProxyModel : :model](#), [SH_ExtendedProxyModel : :setSortKeyColumn\(\)](#), et [SH_SqlDataFields : :setSortOrder\(\)](#).

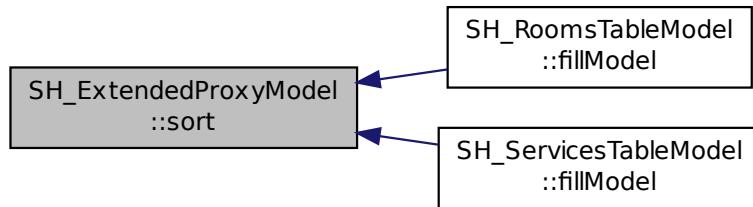
Référencé par [SH_RoomsTableModel : :fillModel\(\)](#), et [SH_ServicesTableModel : :fillModel\(\)](#).

```
00236 {
00237     this->model->field(column)->setSortOrder(newOrder);
00238     SH_ExtendedProxyModel::setSortKeyColumn(column);
00239 }
```

Voici le graphe d'appel pour cette fonction :



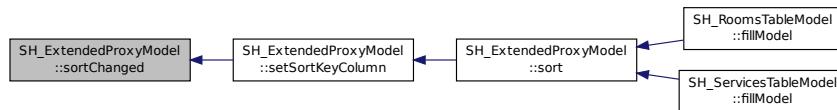
Voici le graphe des appelants de cette fonction :



4.22.3.26 void SH_ExtendedProxyModel ::sortChanged() [signal], [inherited]

Référencé par [SH_ExtendedProxyModel ::setSortKeyColumn\(\)](#).

Voici le graphe des appelants de cette fonction :



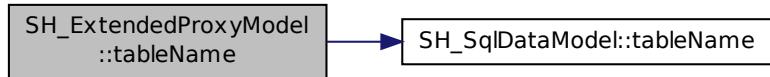
4.22.3.27 const QString SH_ExtendedProxyModel ::tableName() const [inline], [inherited]

Définition à la ligne 45 du fichier [SH_ExtendedSqlProxyModel.h](#).

Références [SH_ExtendedProxyModel ::model](#), et [SH_SqlDataModel ::tableName\(\)](#).

```
00045 { return this->model->tableName(); }
```

Voici le graphe d'appel pour cette fonction :



4.22.4 Documentation des données membres

4.22.4.1 `SH_SqlDataModel* SH_ExtendedProxyModel::model` [protected], [inherited]

Définition à la ligne 241 du fichier [SH_ExtendedSqlProxyModel.h](#).

Référencé par `SH_ExtendedProxyModel::data()`, `SH_ExtendedProxyModel::fetch()`, `SH_ExtendedProxyModel::field()`, `SH_ExtendedProxyModel::fields()`, `SH_ExtendedProxyModel::fieldsCount()`, `SH_BillingsTableModel::fillModel()`, `SH_RoomsTableModel::fillModel()`, `fillModel()`, `SH_ExtendedProxyModel::isEmpty()`, `SH_ExtendedProxyModel::lastError()`, `SH_ExtendedProxyModel::roleNames()`, `SH_ExtendedProxyModel::setSortKeyColumn()`, `SH_BillingsTableModel::SH_BillingsTableModel()`, `SH_BillsTableModel::SH_BillsTableModel()`, `SH_BookingsTableModel::SH_BookingsTableModel()`, `SH_ClientsTableModel::SH_ClientsTableModel()`, `SH_ExtendedProxyModel::SH_ExtendedProxyModel()`, `SH_GroupsTableModel::SH_GroupsTableModel()`, `SH_RoomsTableModel::SH_RoomsTableModel()`, `SH_ServicesTableModel::SH_ServicesTableModel()`, `SH_ExtendedProxyModel::sort()`, et `SH_ExtendedProxyModel::tableName()`.

4.22.5 Documentation des propriétés

4.22.5.1 `bool SH_ExtendedProxyModel::empty` [read], [inherited]

Définition à la ligne 21 du fichier [SH_ExtendedSqlProxyModel.h](#).

4.22.5.2 `QString SH_ExtendedProxyModel::fieldsList` [read], [inherited]

Définition à la ligne 18 du fichier [SH_ExtendedSqlProxyModel.h](#).

4.22.5.3 `QString SH_ExtendedProxyModel::lastError` [read], [inherited]

Définition à la ligne 19 du fichier [SH_ExtendedSqlProxyModel.h](#).

4.22.5.4 `int SH_ExtendedProxyModel::sortKeyColumn` [read], [write], [inherited]

Définition à la ligne 20 du fichier [SH_ExtendedSqlProxyModel.h](#).

4.22.5.5 `QString SH_ExtendedProxyModel::table` [read], [inherited]

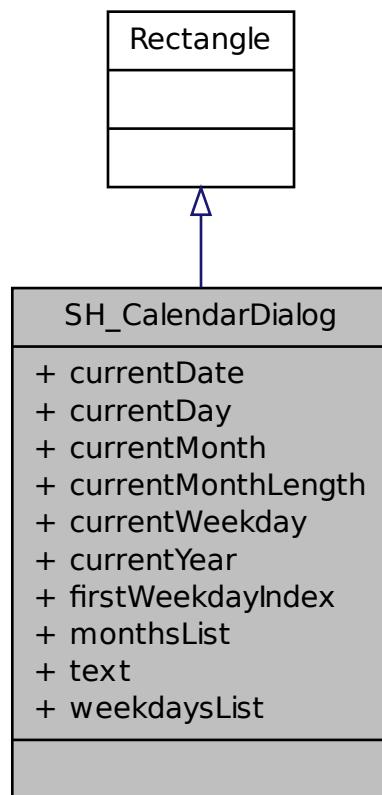
Définition à la ligne 17 du fichier [SH_ExtendedSqlProxyModel.h](#).

La documentation de cette classe a été générée à partir des fichiers suivants :

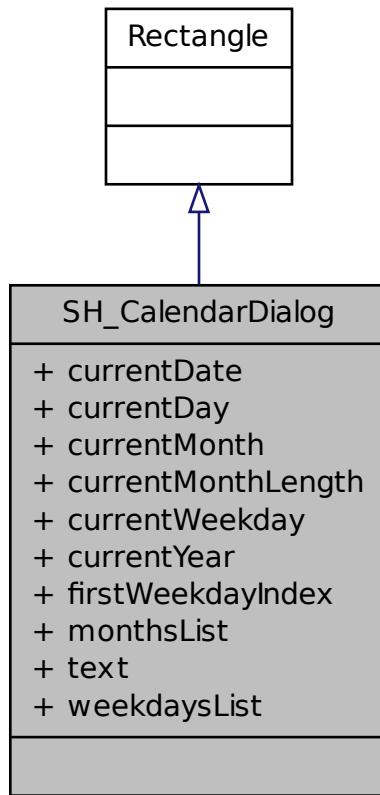
- models/[SH_BookingsTableModel.h](#)
- models/[SH_BookingsTableModel.cpp](#)

4.23 Référence de la classe SH_CalendarDialog

Graphe d'héritage de SH_CalendarDialog :



Graphe de collaboration de SH_CalendarDialog :



Signaux

- void `clicked` (date selectedDate)
- void `closed` ()
- void `opened` ()
- void `refresh` (date newDate)
- void `selected` (string selectedDate)

Propriétés

- alias `currentDate`
- alias `currentDay`
- alias `currentMonth`
- alias `currentMonthLength`
- alias `currentWeekday`
- alias `currentYear`
- int `firstWeekdayIndex`
- variant `monthsList`
- string `text`
- variant `weekdaysList`

4.23.1 Description détaillée

Définition à la ligne 4 du fichier [SH_CalendarDialog.qml](#).

4.23.2 Documentation des fonctions membres

- 4.23.2.1 void SH_CalendarDialog : :clicked (date *selectedDate*) [signal]
- 4.23.2.2 void SH_CalendarDialog : :closed () [signal]
- 4.23.2.3 void SH_CalendarDialog : :opened () [signal]
- 4.23.2.4 void SH_CalendarDialog : :refresh (date *newDate*) [signal]
- 4.23.2.5 void SH_CalendarDialog : :selected (string *selectedDate*) [signal]

4.23.3 Documentation des propriétés

- 4.23.3.1 alias SH_CalendarDialog : :currentDate

Définition à la ligne 11 du fichier [SH_CalendarDialog.qml](#).

- 4.23.3.2 alias SH_CalendarDialog : :currentDay

Définition à la ligne 15 du fichier [SH_CalendarDialog.qml](#).

- 4.23.3.3 alias SH_CalendarDialog : :currentMonth

Définition à la ligne 17 du fichier [SH_CalendarDialog.qml](#).

- 4.23.3.4 alias SH_CalendarDialog : :currentMonthLength

Définition à la ligne 21 du fichier [SH_CalendarDialog.qml](#).

- 4.23.3.5 alias SH_CalendarDialog : :currentWeekday

Définition à la ligne 13 du fichier [SH_CalendarDialog.qml](#).

- 4.23.3.6 alias SH_CalendarDialog : :currentYear

Définition à la ligne 19 du fichier [SH_CalendarDialog.qml](#).

- 4.23.3.7 int SH_CalendarDialog : :firstWeekdayIndex

Définition à la ligne 7 du fichier [SH_CalendarDialog.qml](#).

- 4.23.3.8 variant SH_CalendarDialog : :monthsList

Définition à la ligne 23 du fichier [SH_CalendarDialog.qml](#).

- 4.23.3.9 string SH_CalendarDialog : :text

Définition à la ligne 9 du fichier [SH_CalendarDialog.qml](#).

4.23.3.10 variant SH_CalendarDialog ::weekdaysList

Définition à la ligne 25 du fichier [SH_CalendarDialog.qml](#).

La documentation de cette classe a été générée à partir du fichier suivant :

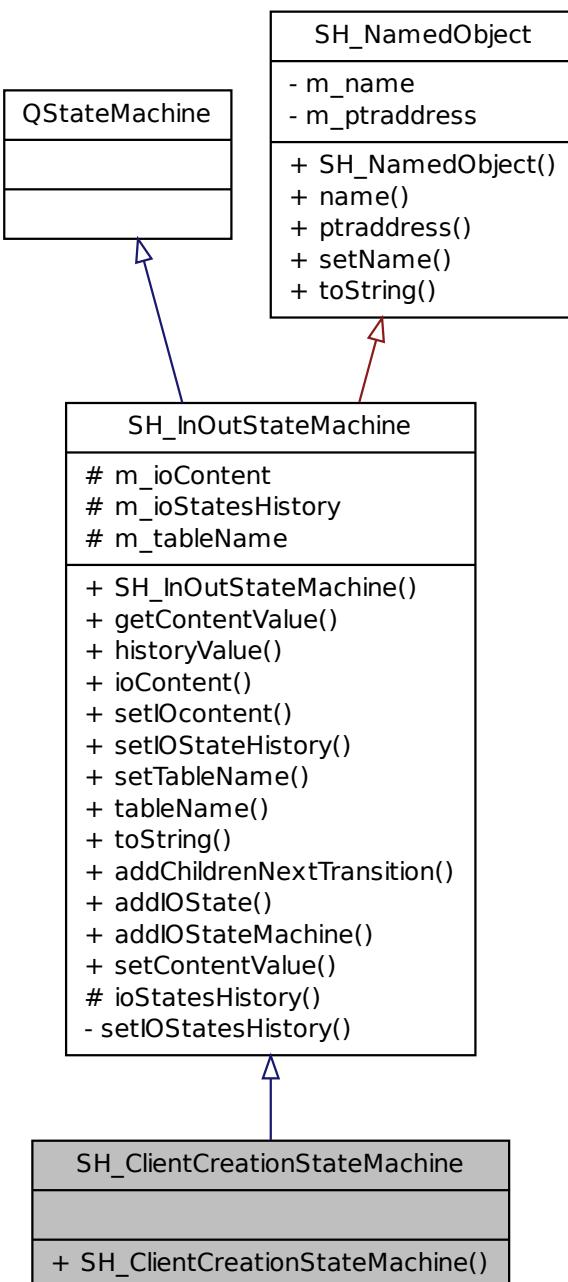
- [views/qml/SH_CalendarDialog.qml](#)

4.24 Référence de la classe SH_ClientCreationStateMachine

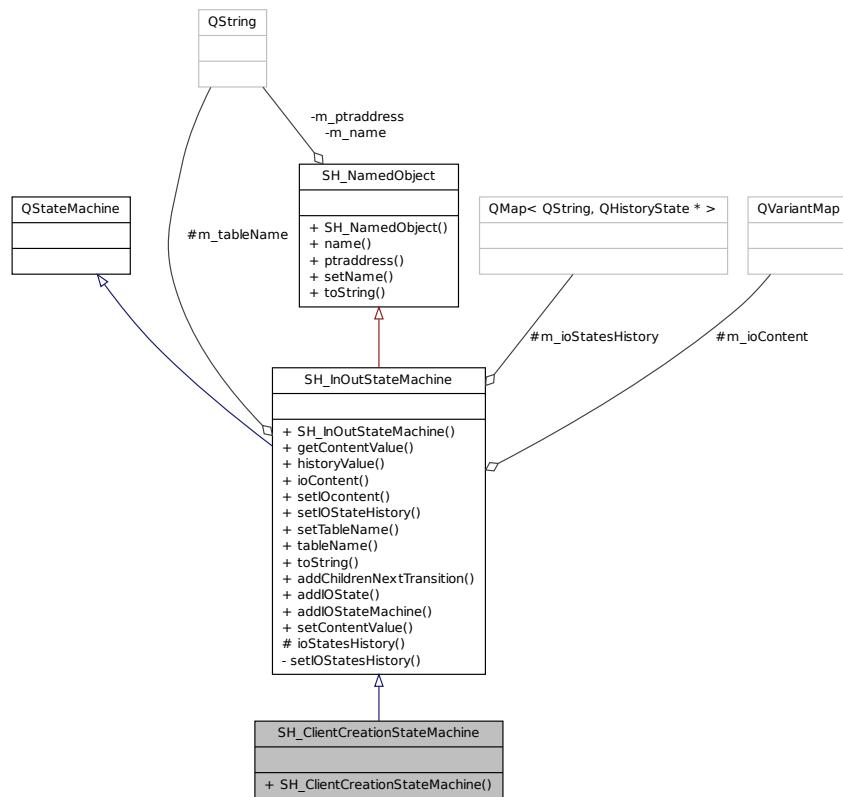
The [SH_ClientCreationStateMachine](#) class.

```
#include <SH_ClientCreation.h>
```

Graphe d'héritage de SH_ClientCreationStateMachine :



Graphe de collaboration de SH_ClientCreationStateMachine :



Connecteurs publics

- void `addChildsNextTransition` (QAbstractState *previousState, QAbstractState *nextState)
- void `addIOState` (SH_InOutState *state, QString field)
- void `addIOStateMachine` (SH_InOutStateMachine *fsm)
- void `setContentValue` (QVariant content, QString field)

Signaux

- void `cancelReplacement` ()
- void `clearAll` ()
- void `confirmInput` ()
- void `displayCalendar` ()
- void `displayFileDialog` ()
- void `next` ()
- void `receiveInput` (QString input)
- void `replaceInput` (QString field)
- void `resendText` (QString text, bool editable=false)
- void `sendText` (QString text, bool editable=false)
- void `validateInput` ()

Fonctions membres publiques

- `SH_ClientCreationStateMachine` (QString name, QObject *parent=0)
- QVariant `getContentValue` (QString field)
- QHistoryState * `historyValue` (QString field)
- QVariantMap `ioContent` () const
- void `setIOContent` (const QVariantMap &ioContent)
- void `setIOStateHistory` (QHistoryState *state, QString field)
- void `setTableName` (const QString &tableName)

- QString [tableName \(\) const](#)
- QString [toString \(\)](#)

Fonctions membres protégées

- QMap< QString, QHistoryState * > [ioStatesHistory \(\) const](#)

Attributs protégés

- QVariantMap [m_ioContent](#)
m_ioContent
- QMap< QString, QHistoryState * > [m_ioStatesHistory](#)
m_ioStatesHistory
- QString [m_tableName](#)
m_tableName

4.24.1 Description détaillée

The [SH_ClientCreationStateMachine](#) class.

Définition à la ligne 8 du fichier [SH_ClientCreation.h](#).

4.24.2 Documentation des constructeurs et destructeur

4.24.2.1 SH_ClientCreationStateMachine : :SH_ClientCreationStateMachine (QString name, QObject * parent = 0)

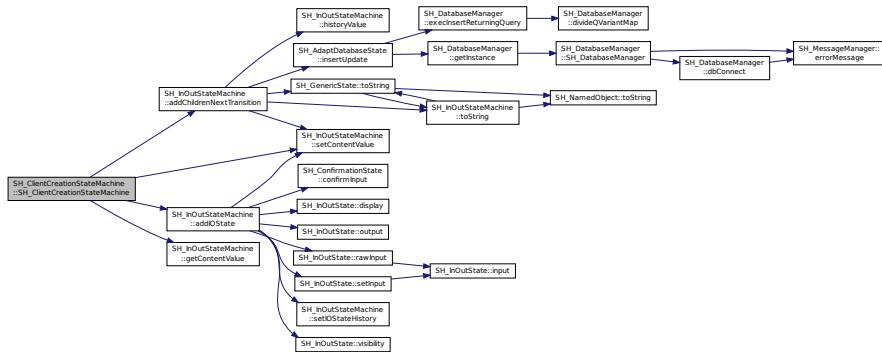
Définition à la ligne 13 du fichier [SH_ClientCreation.cpp](#).

Références [SH_InOutStateMachine : :addChildsNextTransition\(\)](#), [SH_InOutStateMachine : :addIOState\(\)](#), [SH_InOutStateMachine : :getContentValue\(\)](#), et [SH_InOutStateMachine : :setContentValue\(\)](#).

```

00013
00014     SH_InOutStateMachine("CLIENTS",name, parent)
00015 {
00016     SH_AddressCreationStateMachine* address = new
00017         SH_AddressCreationStateMachine("address in client creation");
00018     SH_DatabaseContentQuestionState* nationality = new
00019         SH_DatabaseContentQuestionState("Veuillez entrer le code de nationalité du
00020         client", "nationality in client creation", "NATIONALITIES", "CODE");
00021     SH_FileSelectionState* IDscan = new
00022         SH_FileSelectionState("Veuillez sélectionner l'image scannée des papiers d'identité du
00023         client", "ID image in client creation");
00024     SH_RegExpQuestionState* phone = new
00025         SH_RegExpQuestionState("Veuillez entrer le numéro de téléphone du client avec le
00026         préfixe international", "phone client creation", QRegularExpression("[[+|00][1-9]]{3}\\d{8}]?"));
00027     SH_RegExpQuestionState* email = new
00028         SH_RegExpQuestionState("Veuillez entrer l'adresse email du client", "email client
00029         creation", QRegularExpression("[[\\d\\w\\.\\%\\+\\-]+@[\\d\\w\\.-]+\\.\\[\\w+]?"));
00030     QFinalState* final = new QFinalState();
00031
00032     this->addState(final);
00033     this->addIOState(IDscan, "IDSCAN");
00034     this->addIOState(nationality, "NATIONALITY_CODE");
00035     this->addIOState(phone, "PHONE");
00036     this->addIOState(email, "EMAIL");
00037
00038     connect(address, &SH_InOutStateMachine::exited, [=] () {setContentValue(address->
00039         getContentValue("ID"), "HOMEADDRESS_ID");});
00040     this->addChildsNextTransition(IDscan, address);
00041     this->addChildsNextTransition(address, phone);
00042     this->addChildsNextTransition(phone, email);
00043     this->addChildsNextTransition(email, nationality);
00044     this->setInitialState(IDscan);
00045
00046     /*connect (IDscan,&QState::entered, [=] () {
00047         connect (this, &IOStateMachine::confirmInput, IDscan, &GenericState::next);
00048     });
00049 */
00050 }
```

Voici le graphe d'appel pour cette fonction :



4.24.3 Documentation des fonctions membres

4.24.3.1 void SH_InOutStateMachine : :addChildrenNextTransition (QAbstractState * *previousState*, QAbstractState * *nextState*) [slot], [inherited]

Définition à la ligne 250 du fichier SH_IStateMachine.cpp.

Références SH_InOutStateMachine ::clearAll(), SH_InOutStateMachine ::historyValue(), SH_AdaptDatabase-State ::insertUpdate(), SH_InOutStateMachine ::m_ioContent, SH_InOutStateMachine ::m_tableName, SH_In-OutStateMachine ::next(), SH_InOutStateMachine ::replaceInput(), SH_InOutStateMachine ::sendText(), SH_In-OutStateMachine ::setContentValue(), SH_GenericState ::toString(), et SH_InOutStateMachine ::toString().

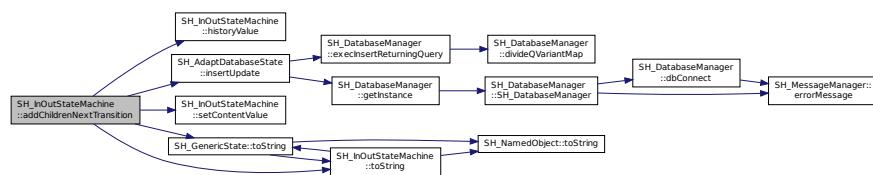
Référencé par [SH_BillingCreationStateMachine](#) : `.SH_BillingCreationStateMachine()`, et [SH_ClientCreationStateMachine](#) : `.SH_ClientCreationStateMachine()`.

```
00251 {  
00252     SH_InOutStateMachine* fsmPreviousState = qobject_cast<  
SH_InOutStateMachine*>(previousState);  
00253     SH_GenericState* genPreviousState = qobject_cast<  
SH_GenericState*>(previousState);  
00254     QFinalState* final = qobject_cast<QFinalState*>(nextState);  
00255     if(final) {  
00256         SH_AdaptDatabaseState* saveState = new  
SH_AdaptDatabaseState("enregistrement de la machine " +  
toString());  
00257         if(genPreviousState) {  
00258             genPreviousState->addTransition(genPreviousState, SIGNAL(next()), saveState);  
00259         }  
00260         if(fsmPreviousState) {  
00261             fsmPreviousState->addTransition(fsmPreviousState, SIGNAL(next()), saveState);  
00262         }  
00263         if(genPreviousState || fsmPreviousState) {  
00264             connect(previousState, &QAbstractState::exited, [=]() {  
00265                 connect(saveState, &QAbstractState::entered, [=]() {  
00266                     emit this->sendText("Merci !");  
00267                     setContentValue(saveState->insertUpdate(  
m_tableName, m_ioContent), "ID");  
00268                     emit this->clearAll();  
00269                 });  
00270             });  
00271             saveState->addTransition(saveState, SIGNAL(next()), final);  
00272         }  
00273     } else {  
00274         if(genPreviousState) {  
00275             qDebug() << "next transition between " << genPreviousState->toString() << " and " <<  
nextState;  
00276             genPreviousState->addTransition(genPreviousState, SIGNAL(next()), nextState);  
00277         }  
00278         if(fsmPreviousState) {  
00279             qDebug() << "next transition between " << fsmPreviousState->toString() << " and " <<  
nextState;  
00280             fsmPreviousState->addTransition(fsmPreviousState, SIGNAL(next()), nextState);  
00281         }  
00282     }  
00283     if(genPreviousState) {  
00284         connect(previousState, &QAbstractState::exited, [=]() {  
00285             if(fsmPreviousState) {  
00286                 fsmPreviousState->removeTransition(fsmPreviousState, SIGNAL(next()));  
00287             }  
00288             if(genPreviousState) {  
00289                 genPreviousState->removeTransition(genPreviousState, SIGNAL(next()));  
00290             }  
00291         });  
00292     }  
00293 }
```

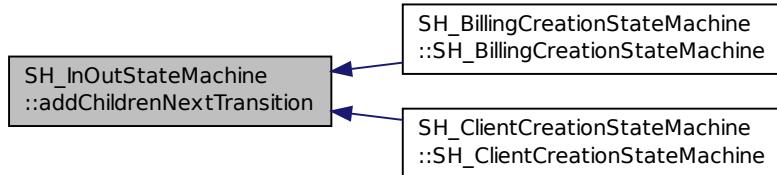
```

00284     /*à faire au moment de l'entrée dans l'état previousState*/
00285     connect(genPreviousState, &QAbstractState::entered, [=]() {
00286         connect(this, &SH_InOutStateMachine::replaceInput, [=]()
00287             QString field) {
00288             /*après avoir demandé à revenir sur un état précédent, on attend la fin de l'état actuel
00289             puis on retourne à l'historique de l'état désiré; celui-ci fini, on passe à l'état qui aurait du suivre celui
00290             pendant lequel on a demandé à revenir sur un état précédent*/
00291             QHistoryState* hState = historyValue(field);
00292             if(hState) { /*si l'historique existe (on a déjà quitté l'état voulu)*/
00293                 hState->parentState()->addTransition(hState->parentState(), SIGNAL(
00294                     next()), nextState);
00295                 genPreviousState->addTransition(genPreviousState, SIGNAL(
00296                     next()), hState);
00297             }
00298         });
00299     });
00300 }
```

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :



4.24.3.2 void SH_InOutStateMachine ::addIOState (SH_InOutState * state, QString field) [slot], [inherited]

Définition à la ligne 110 du fichier [SH_IOStateMachine.cpp](#).

Références [SH_ConfirmationState ::confirmInput\(\)](#), [SH_InOutState ::display\(\)](#), [SH_InOutStateMachine ::displayCalendar\(\)](#), [SH_InOutStateMachine ::displayFileDialog\(\)](#), [SH_InOutState ::output\(\)](#), [SH_InOutState ::rawInput\(\)](#), [SH_InOutStateMachine ::receiveInput\(\)](#), [SH_InOutState ::resendInput\(\)](#), [SH_InOutStateMachine ::resendText\(\)](#), [SH_InOutState ::sendOutput\(\)](#), [SH_InOutStateMachine ::sendText\(\)](#), [SH_InOutStateMachine ::setContentValue\(\)](#), [SH_InOutState ::setInput\(\)](#), [SH_InOutStateMachine ::setIOStateHistory\(\)](#), [SH_InOutStateMachine ::validateInput\(\)](#), et [SH_InOutState ::visibility\(\)](#).

Référencé par [SH_BillingCreationStateMachine ::SH_BillingCreationStateMachine\(\)](#), [SH_ClientCreationStateMachine\(\)](#), et [SH_ServiceCharging ::SH_ServiceCharging\(\)](#).

```

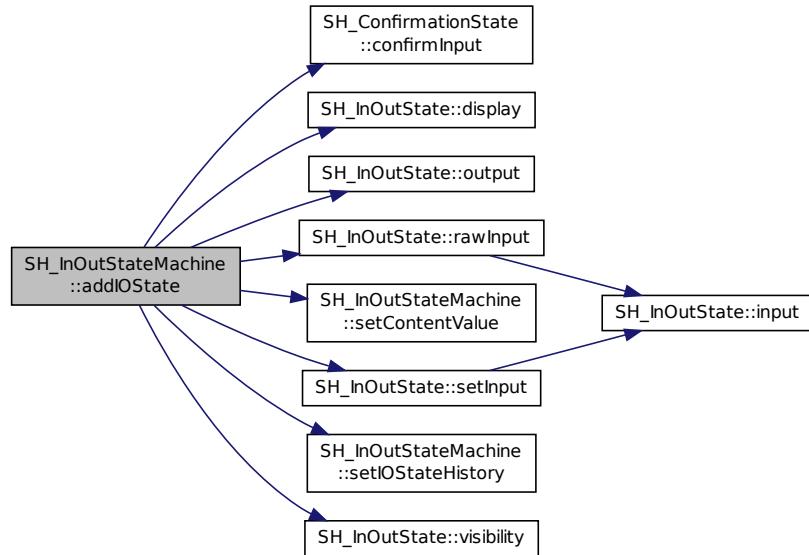
00111 {
00112     /*à faire au moment de l'entrée dans l'état state*/
00113     connect(state, &QState::entered, [=]() {
00114         qDebug() << "entered !";
00115         state->display(true);
00116         connect(this, &SH_InOutStateMachine::receiveInput, state, &
```

```

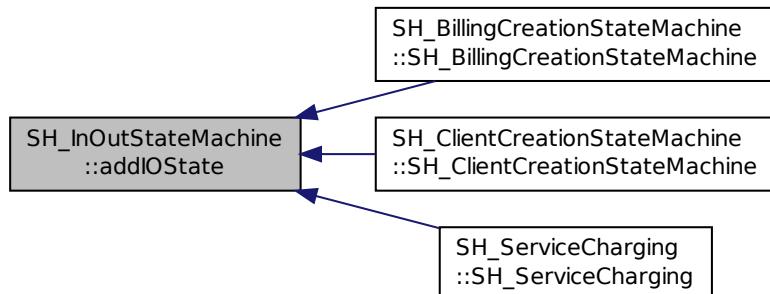
SH_InOutState::setInput); /* la réception d'une valeur entraîne son enregistrement
comme entrée de l'utilisateur auprès de l'état*/
0017   connect(this, &SH_InOutStateMachine::receiveInput, [=](QString in
){ qDebug() << "hello world !"; state->setInput(in);}); /* la réception d'une valeur entraîne son
enregistrement comme entrée de l'utilisateur auprès de l'état*/
0018   connect(state, &SH_InOutState::sendOutput, [=](QVariant out) {qDebug() <<
"connected !"; emit this->sendText(out.toString(), false);});
0019   connect(state, &SH_InOutState::resendInput, [=](QVariant in) {emit this->
resendText(in.toString(), true);});
0020   if(state->visibility()) {
0021     state->sendOutput(QVariant(state->output()));
0022   } else {
0023     qDebug() << "invisible";
0024   }
0025 });
0026 SH_ValidationState *validationState = qobject_cast<
SH_ValidationState*>(state);
0027 if(validationState) {
0028   /*à faire au moment de l'entrée dans l'état state*/
0029   connect(validationState, &QState::entered, [=]() {
0030     connect(this, &SH_InOutStateMachine::validateInput,
validationState, &SH_ValidationState::confirmInput);
0031   });
0032 }
0033 SH_ConfirmationState *confirmationState = qobject_cast<
SH_ConfirmationState*>(state);
0034 if(confirmationState) {
0035   /*à faire au moment de l'entrée dans l'état state*/
0036   connect(confirmationState, &QState::entered, [=]() {
0037     connect(this, &SH_InOutStateMachine::validateInput,
confirmationState, &SH_ConfirmationState::confirmInput);
0038   });
0039 }
0040 SH_DateQuestionState *dateState = qobject_cast<
SH_DateQuestionState*>(state);
0041 if(dateState) {
0042   /*à faire au moment de l'entrée dans l'état state*/
0043   connect(dateState, &QState::entered, this, &
SH_InOutStateMachine::displayCalendar);
0044 }
0045 SH_FileSelectionState *fileState = qobject_cast<
SH_FileSelectionState*>(state);
0046 if(fileState) {
0047   /*à faire au moment de l'entrée dans l'état state*/
0048   connect(fileState, &QState::entered, this, &
SH_InOutStateMachine::displayFileDialog);
0049 }
0050 /*à faire au moment de la sortie de l'état state*/
0051 connect(state, &QState::exited, [=]() {
0052   qDebug() << "exited !";
0053   if(!field.isEmpty()) {
0054     setContentValue(state->rawInput(), field);
0055     /*gestion de l'historique des états pour pouvoir revenir à l'état state après l'avoir quitté*/
0056     QHistoryState* hState = new QHistoryState(state);
0057     setIOStateHistory(hState, field);
0058   }
0059   state->disconnect(this); /*plus aucune action sur l'état ne pourra être provoquée par la machine*/
0060 });
0061
0062
0063 QAbstractState* astate = qobject_cast<QAbstractState *>(state);
0064 if(astate) {
0065   addState(astate);
0066 }
0067 }

```

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :



4.24.3.3 void SH_InOutStateMachine ::addIOStateMachine (SH_InOutStateMachine * fsm) [slot], [inherited]

Définition à la ligne 175 du fichier [SH_IOStateMachine.cpp](#).

Références `SH_InOutStateMachine ::cancelReplacement()`, `SH_InOutStateMachine ::confirmInput()`, `SH_InOutStateMachine ::displayCalendar()`, `SH_InOutStateMachine ::receiveInput()`, `SH_InOutStateMachine ::replaceInput()`, `SH_InOutStateMachine ::resendText()`, `SH_InOutStateMachine ::sendText()`, et `SH_InOutStateMachine ::validateInput()`.

Référencé par [SH_BillingCreationStateMachine ::SH_BillingCreationStateMachine\(\)](#).

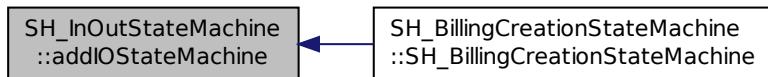
```

00176 {
00177     /* à faire au moment de l'entrée dans la machine d'état fsm*/
  
```

```

00178     connect(fsm, &QState::entered, [=]() {
00179         connect(this, &SH_InOutStateMachine::receiveInput, fsm, &
00180             SH_InOutStateMachine::receiveInput);
00180         connect(this, &SH_InOutStateMachine::sendText, fsm, &
00181             SH_InOutStateMachine::sendText);
00181         connect(this, &SH_InOutStateMachine::resendText, fsm, &
00182             SH_InOutStateMachine::resendText);
00182         connect(this, &SH_InOutStateMachine::confirmInput, fsm, &
00183             SH_InOutStateMachine::confirmInput);
00183         connect(this, &SH_InOutStateMachine::validateInput, fsm, &
00184             SH_InOutStateMachine::validateInput);
00184         connect(this, &SH_InOutStateMachine::replaceInput, fsm, &
00185             SH_InOutStateMachine::replaceInput);
00185         connect(this, &SH_InOutStateMachine::cancelReplacement, fsm,
00186             &SH_InOutStateMachine::cancelReplacement);
00186         connect(this, &SH_InOutStateMachine::displayCalendar, fsm, &
00187             SH_InOutStateMachine::displayCalendar);
00187     });
00188     /*à faire au moment de la sortie de la machine d'état fsm*/
00189     connect(fsm, &QState::exited, [=]() {
00190         fsm->disconnect(this); /*plus aucune action sur la machine d'état fille ne pourra être provoquée
00190                               par la machine mère*/
00191     });
00192 }
00193 }
```

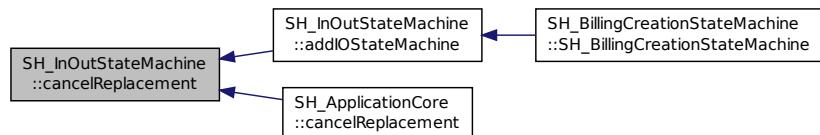
Voici le graphe des appels de cette fonction :



4.24.3.4 void SH_InOutStateMachine ::cancelReplacement() [signal], [inherited]

Référencé par [SH_InOutStateMachine ::addIOStateMachine\(\)](#), et [SH_ApplicationCore ::cancelReplacement\(\)](#).

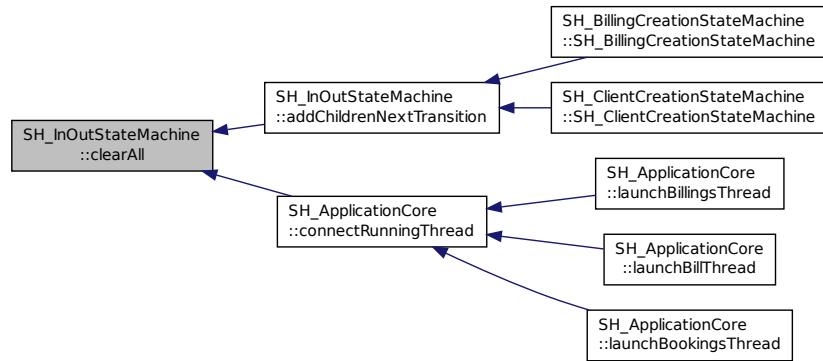
Voici le graphe des appels de cette fonction :



4.24.3.5 void SH_InOutStateMachine ::clearAll() [signal], [inherited]

Référencé par [SH_InOutStateMachine ::addChildrenNextTransition\(\)](#), et [SH_ApplicationCore ::connectRunningThread\(\)](#).

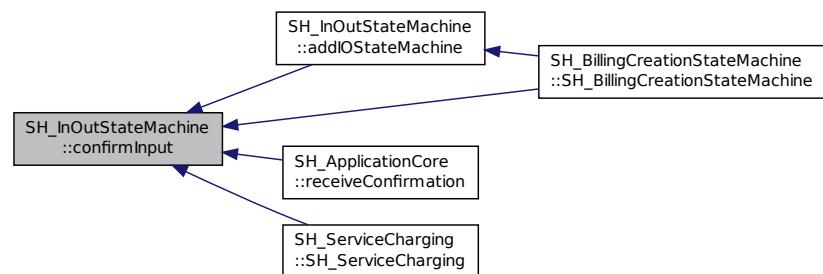
Voici le graphe des appelants de cette fonction :



4.24.3.6 void SH_InOutStateMachine ::confirmInput() [signal], [inherited]

Référencé par [SH_InOutStateMachine ::addIOStateMachine\(\)](#), [SH_ApplicationCore ::receiveConfirmation\(\)](#), [SH_BillingCreationStateMachine ::SH_BillingCreationStateMachine\(\)](#), et [SH_ServiceCharging ::SH_ServiceCharging\(\)](#).

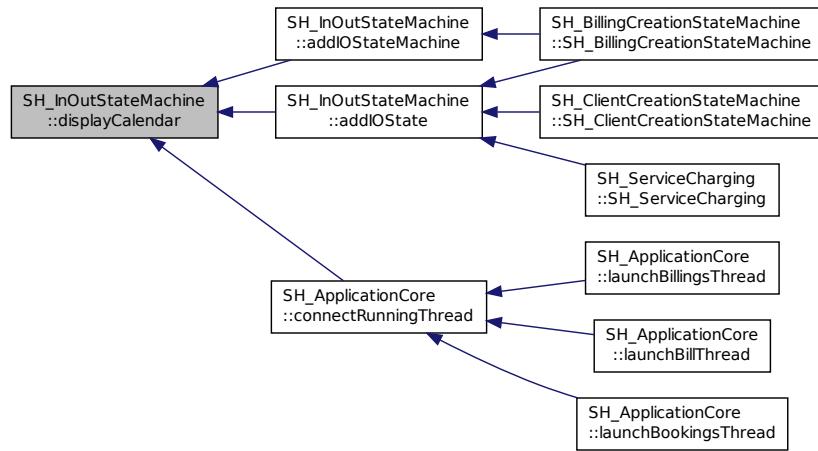
Voici le graphe des appelants de cette fonction :



4.24.3.7 void SH_InOutStateMachine ::displayCalendar() [signal], [inherited]

Référencé par [SH_InOutStateMachine ::addIOState\(\)](#), [SH_InOutStateMachine ::addIOStateMachine\(\)](#), et [SH_ApplicationCore ::connectRunningThread\(\)](#).

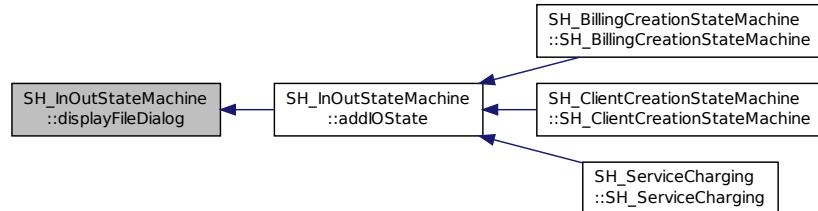
Voici le graphe des appelants de cette fonction :



4.24.3.8 void SH_InOutStateMachine ::displayFileDialog () [signal], [inherited]

Référencé par [SH_InOutStateMachine ::addIOState\(\)](#).

Voici le graphe des appelants de cette fonction :



4.24.3.9 QVariant SH_InOutStateMachine ::getContentValue (QString field) [inherited]

Définition à la ligne 65 du fichier [SH_IOStateMachine.cpp](#).

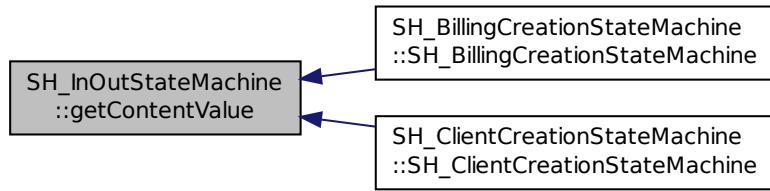
Références [SH_InOutStateMachine ::m_ioContent](#).

Référencé par [SH_BillingCreationStateMachine ::SH_BillingCreationStateMachine\(\)](#), et [SH_ClientCreationStateMachine\(\)](#).

```

00066 {
00067     return m_ioContent.value(field);
00068 }
```

Voici le graphe des appelants de cette fonction :



4.24.3.10 QHistoryState * SH_InOutStateMachine ::historyValue (QString field) [inherited]

Définition à la ligne 238 du fichier [SH_IOStateMachine.cpp](#).

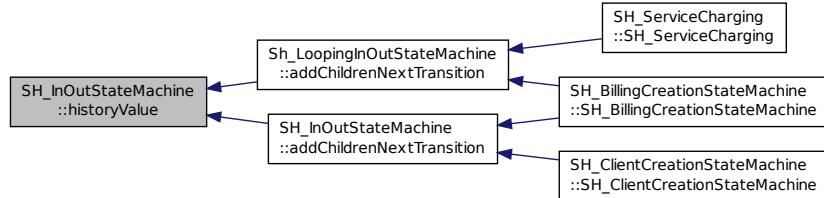
Références [SH_InOutStateMachine ::m_ioStatesHistory](#).

Référencé par [Sh_LoopingInOutStateMachine ::addChildrenNextTransition\(\)](#), et [SH_InOutStateMachine ::addChildrenNextTransition\(\)](#).

```

00239 {
00240     return m_ioStatesHistory.value(field);
00241 }
  
```

Voici le graphe des appelants de cette fonction :



4.24.3.11 QVariantMap SH_InOutStateMachine ::ioContent () const [inherited]

Définition à la ligne 43 du fichier [SH_IOStateMachine.cpp](#).

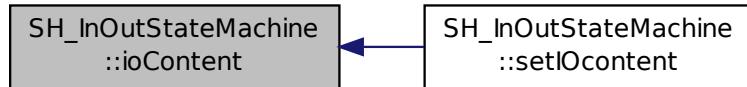
Références [SH_InOutStateMachine ::m_ioContent](#).

Référencé par [SH_InOutStateMachine ::setIOcontent\(\)](#).

```

00044 {
00045     return m_ioContent;
00046 }
  
```

Voici le graphe des appelants de cette fonction :



4.24.3.12 QMap< QString, QHistoryState * > SH_InOutStateMachine ::ioStatesHistory() const [protected], [inherited]

Définition à la ligne 202 du fichier [SH_IOStateMachine.cpp](#).

Références [SH_InOutStateMachine ::m_ioStatesHistory](#).

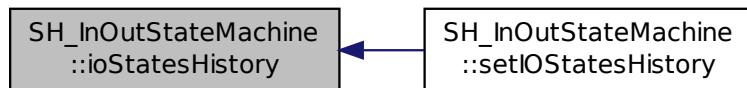
Référencé par [SH_InOutStateMachine ::setIOStatesHistory\(\)](#).

```

00203 {
00204     return m_ioStatesHistory;
00205 }

```

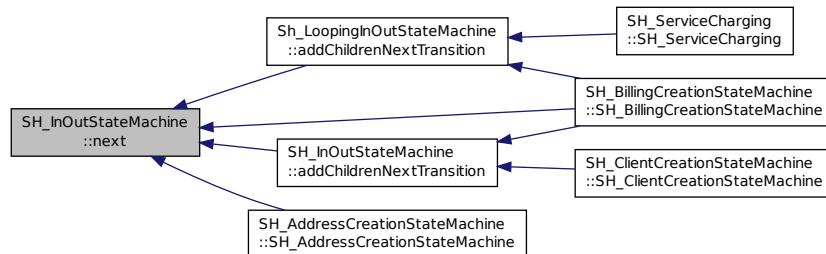
Voici le graphe des appelants de cette fonction :



4.24.3.13 void SH_InOutStateMachine ::next() [signal], [inherited]

Référencé par [Sh_LoopingInOutStateMachine ::addChildsNextTransition\(\)](#), [SH_InOutStateMachine ::addChildsNextTransition\(\)](#), [SH_AddressCreationStateMachine ::SH_AddressCreationStateMachine\(\)](#), et [SH_BillingCreationStateMachine ::SH_BillingCreationStateMachine\(\)](#).

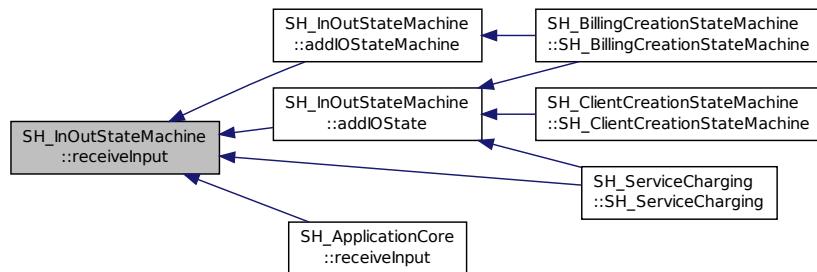
Voici le graphe des appelants de cette fonction :



4.24.3.14 void SH_InOutStateMachine ::receiveInput (QString *input*) [signal], [inherited]

Référencé par [SH_InOutStateMachine ::addIOState\(\)](#), [SH_InOutStateMachine ::addIOStateMachine\(\)](#), [SH_ApplicationCore ::receiveInput\(\)](#), et [SH_ServiceCharging ::SH_ServiceCharging\(\)](#).

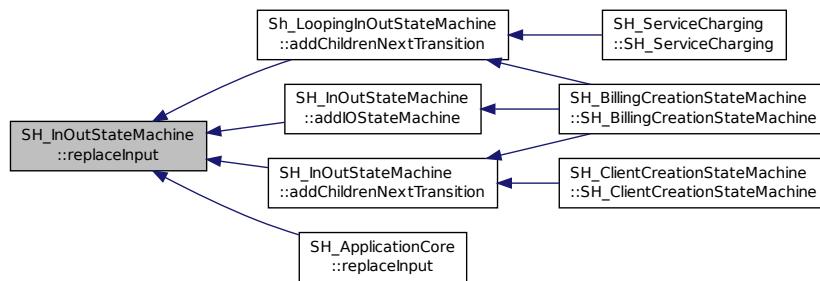
Voici le graphe des appelleurs de cette fonction :



4.24.3.15 void SH_InOutStateMachine ::replaceInput (QString *field*) [signal], [inherited]

Référencé par [Sh_LoopingInOutStateMachine ::addChildsNextTransition\(\)](#), [SH_InOutStateMachine ::addChildrenNextTransition\(\)](#), [SH_InOutStateMachine ::addIOStateMachine\(\)](#), et [SH_ApplicationCore ::replaceInput\(\)](#).

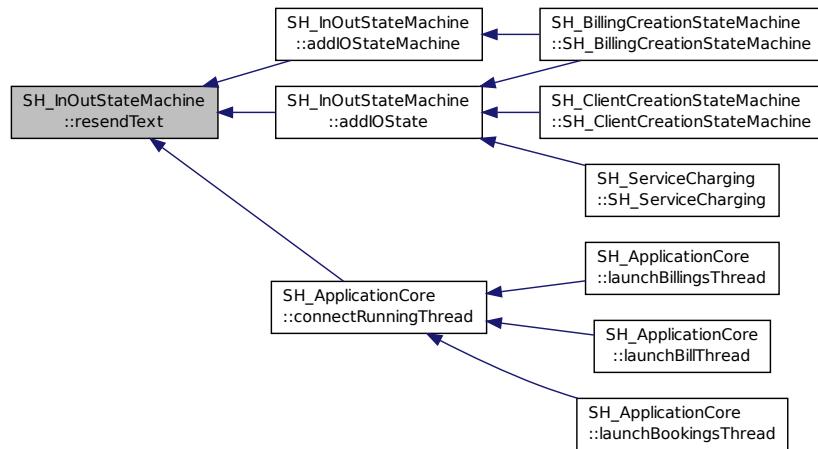
Voici le graphe des appelleurs de cette fonction :



4.24.3.16 void SH_InOutStateMachine ::resendText (QString *text*, bool *editable* = false) [signal], [inherited]

Référencé par [SH_InOutStateMachine ::addIOState\(\)](#), [SH_InOutStateMachine ::addIOStateMachine\(\)](#), et [SH_ApplicationCore ::connectRunningThread\(\)](#).

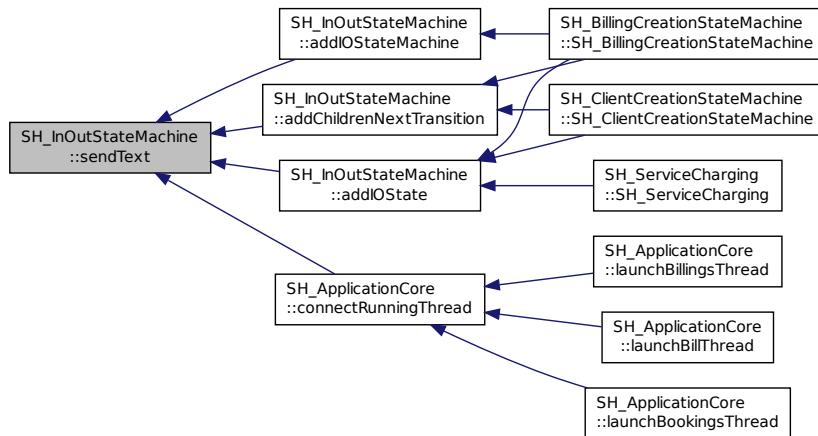
Voici le graphe des appels de cette fonction :



4.24.3.17 void **SH_InOutStateMachine** ::**sendText** (**QString text**, **bool editable = false**) [signal], [inherited]

Référencé par **SH_InOutStateMachine** ::**addChildrenNextTransition()**, **SH_InOutStateMachine** ::**addIOState()**, **SH_InOutStateMachine** ::**addIOStateMachine()**, et **SH_ApplicationCore** ::**connectRunningThread()**.

Voici le graphe des appels de cette fonction :



4.24.3.18 void **SH_InOutStateMachine** ::**setContentValue** (**QVariant content**, **QString field**) [slot], [inherited]

Définition à la ligne 99 du fichier **SH_IOStateMachine.cpp**.

Références **SH_InOutStateMachine** ::**m_ioContent**.

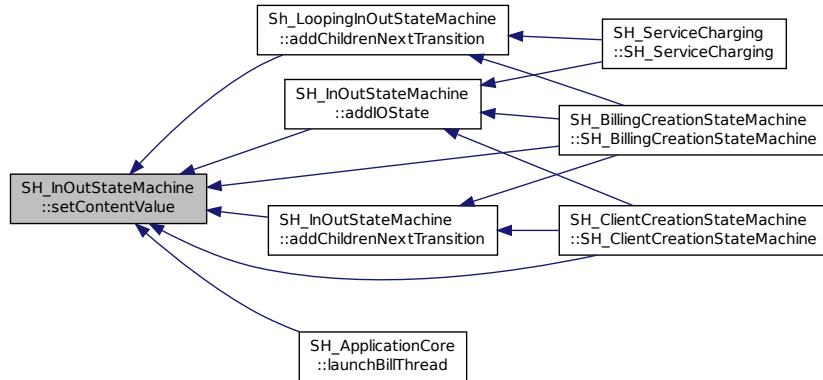
Référencé par **Sh_LoopingInOutStateMachine** ::**addChildrenNextTransition()**, **SH_InOutStateMachine** ::**addChildrenNextTransition()**, **SH_InOutStateMachine** ::**addIOState()**, **SH_ApplicationCore** ::**launchBillThread()**, **SH_BillingCreationStateMachine** ::**SH_BillingCreationStateMachine()**, et **SH_ClientCreationStateMachine()**.

```

00100 {
00101     m_ioContent.insert(field, content);
00102 }

```

Voici le graphe des appels de cette fonction :



4.24.3.19 void SH_InOutStateMachine ::setIOcontent (const QVariantMap & ioContent) [inherited]

Définition à la ligne 54 du fichier [SH_IOStateMachine.cpp](#).

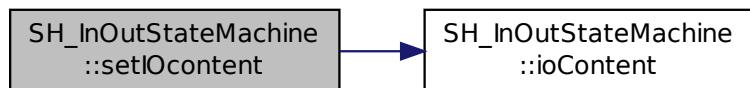
Références [SH_InOutStateMachine ::ioContent\(\)](#), et [SH_InOutStateMachine ::m_ioContent](#).

```

00055 {
00056     m_ioContent = ioContent;
00057 }

```

Voici le graphe d'appel pour cette fonction :



4.24.3.20 void SH_InOutStateMachine ::setIOStateHistory (QHistoryState * state, QString field) [inherited]

Définition à la ligne 226 du fichier [SH_IOStateMachine.cpp](#).

Références [SH_InOutStateMachine ::m_ioStatesHistory](#).

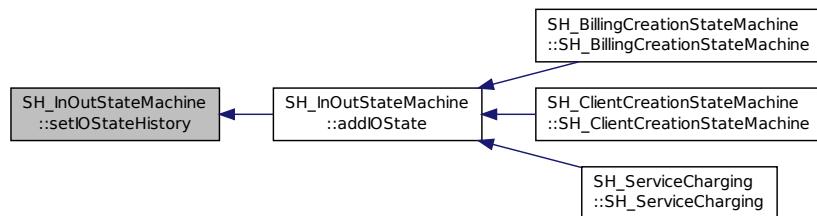
Référencé par [SH_InOutStateMachine ::addIOState\(\)](#).

```

00227 {
00228     m_ioStatesHistory.insert(field, state); /*remplacement si plusieurs fois*/
00229 }

```

Voici le graphe des appels de cette fonction :



4.24.3.21 void SH_InOutStateMachine ::setTableName (const QString & tableName) [inherited]

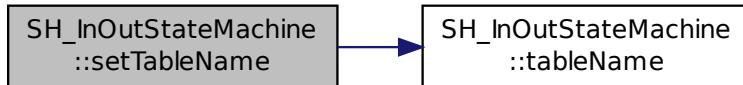
Définition à la ligne 87 du fichier [SH_IOStateMachine.cpp](#).

Références [SH_InOutStateMachine ::m_tableName](#), et [SH_InOutStateMachine ::tableName\(\)](#).

```

00088 {
00089     m_tableName = tableName;
00090 }
```

Voici le graphe d'appel pour cette fonction :



4.24.3.22 QString SH_InOutStateMachine ::tableName () const [inherited]

Définition à la ligne 76 du fichier [SH_IOStateMachine.cpp](#).

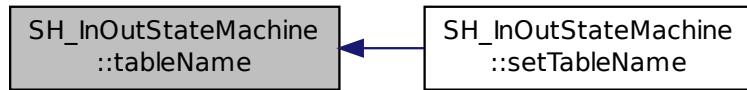
Références [SH_InOutStateMachine ::m_tableName](#).

Référencé par [SH_InOutStateMachine ::setTableName\(\)](#).

```

00077 {
00078     return m_tableName;
00079 }
```

Voici le graphe des appelants de cette fonction :



4.24.3.23 QString SH_InOutStateMachine ::toString() [virtual], [inherited]

Réimplémentée à partir de [SH_NamedObject](#).

Définition à la ligne 26 du fichier [SH_IOSMachine.cpp](#).

Références [SH_NamedObject ::toString\(\)](#), et [SH_GenericState ::toString\(\)](#).

Référencé par [Sh_LoopingInOutStateMachine ::addChildsNextTransition\(\)](#), [SH_InOutStateMachine ::addChildsNextTransition\(\)](#), [SH_ApplicationCore ::launchBillingsThread\(\)](#), [SH_BillingCreationStateMachine ::SH_BillingCreationStateMachine\(\)](#), et [SH_GenericState ::toString\(\)](#).

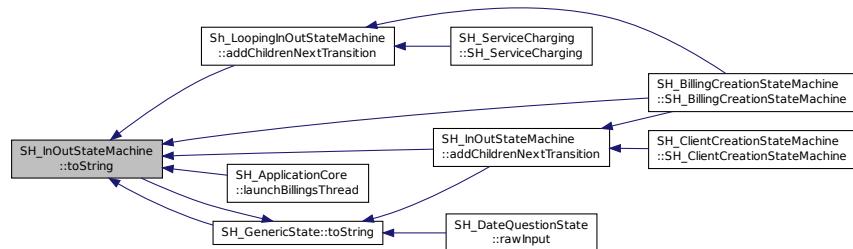
```

00027 {
00028     QObject* parent = this->parent();
00029     SH_GenericState* par = qobject_cast<SH_GenericState *>(parent);
00030     if(par) {
00031         return SH_NamedObject::toString() + " [descending from "+par->
00032             toString()+"] ";
00033     } else {
00034         return SH_NamedObject::toString();
00035     }
  
```

Voici le graphe d'appel pour cette fonction :



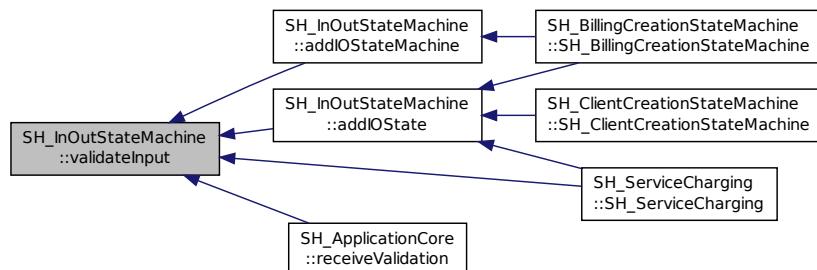
Voici le graphe des appelants de cette fonction :



4.24.3.24 void SH_InOutStateMachine ::validateInput() [signal], [inherited]

Référencé par [SH_InOutStateMachine ::addIOState\(\)](#), [SH_InOutStateMachine ::addIOStateMachine\(\)](#), [SH_ApplicationCore ::receiveValidation\(\)](#), et [SH_ServiceCharging ::SH_ServiceCharging\(\)](#).

Voici le graphe des appelleurs de cette fonction :



4.24.4 Documentation des données membres

4.24.4.1 QVariantMap SH_InOutStateMachine ::m_ioContent [protected], [inherited]

`m_ioContent`

Définition à la ligne [209](#) du fichier [SH_IOStateMachine.h](#).

Référencé par [Sh_LoopingInOutStateMachine ::addChildsNextTransition\(\)](#), [SH_InOutStateMachine ::addChildsNextTransition\(\)](#), [SH_InOutStateMachine ::getContentValue\(\)](#), [SH_InOutStateMachine :::ioContent\(\)](#), [SH_InOutStateMachine ::setContentValue\(\)](#), [SH_InOutStateMachine ::setIOcontent\(\)](#), et [SH_BillingCreationStateMachine ::SH_BillingCreationStateMachine\(\)](#).

4.24.4.2 QMap<QString, QHistoryState*> SH_InOutStateMachine ::m_ioStatesHistory [protected], [inherited]

`m_ioStatesHistory`

Définition à la ligne [217](#) du fichier [SH_IOStateMachine.h](#).

Référencé par [SH_InOutStateMachine ::historyValue\(\)](#), [SH_InOutStateMachine :::ioStatesHistory\(\)](#), [SH_InOutStateMachine ::setIOStateHistory\(\)](#), et [SH_InOutStateMachine ::setIOStatesHistory\(\)](#).

4.24.4.3 QString SH_InOutStateMachine ::m_tableName [protected], [inherited]

`m_tableName`

Définition à la ligne [213](#) du fichier [SH_IOStateMachine.h](#).

Référencé par [Sh_LoopingInOutStateMachine ::addChildsNextTransition\(\)](#), [SH_InOutStateMachine ::addChildsNextTransition\(\)](#), [SH_InOutStateMachine :::setTableName\(\)](#), [SH_BillingCreationStateMachine ::SH_BillingCreationStateMachine\(\)](#), et [SH_InOutStateMachine ::tableName\(\)](#).

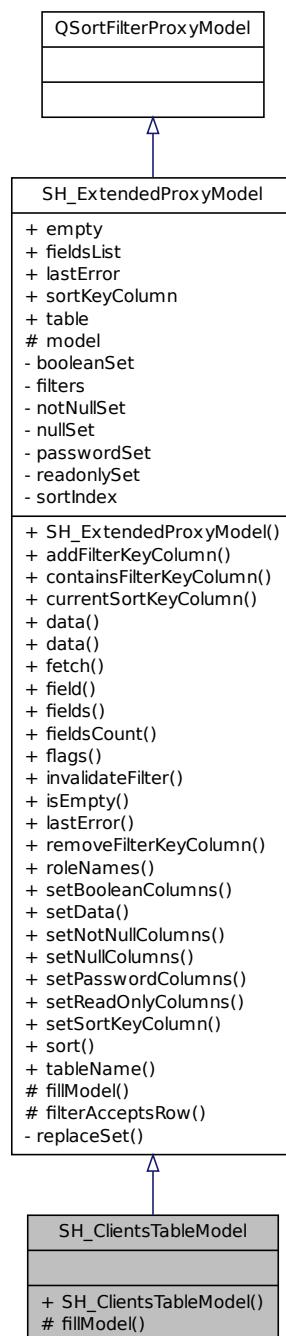
La documentation de cette classe a été générée à partir des fichiers suivants :

- logic/[SH_ClientCreation.h](#)
- logic/[SH_ClientCreation.cpp](#)

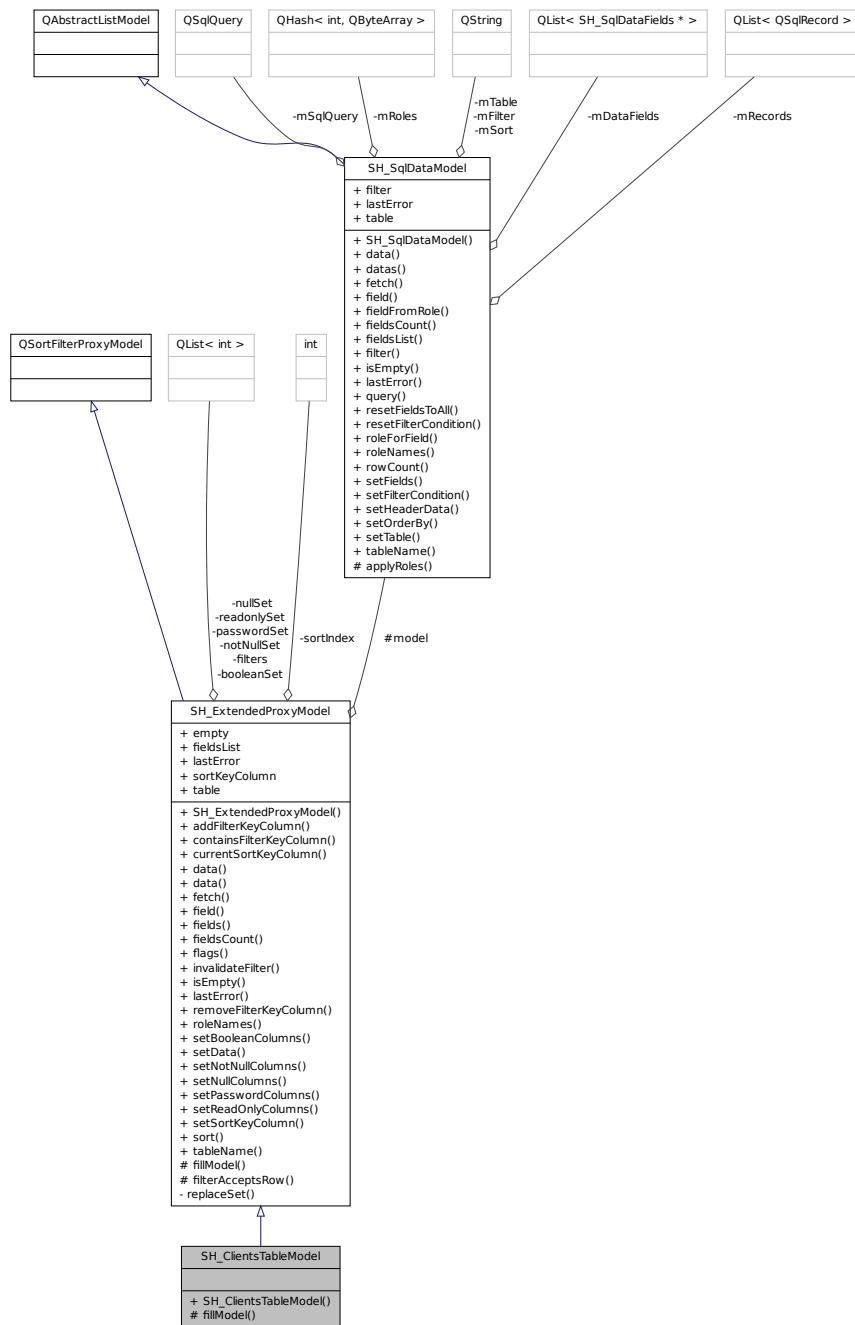
4.25 Référence de la classe SH_ClientsTableModel

```
#include <SH_ClientsTableModel.h>
```

Graphe d'héritage de SH_ClientsTableModel :



Graphe de collaboration de SH_ClientsTableModel :



Signaux

- void `sortChanged ()`

Fonctions membres publiques

- `SH_ClientsTableModel (QObject *parent=0)`
- `Q_INVOKABLE void addFilterKeyColumn (int column)`
- `Q_INVOKABLE bool containsFilterKeyColumn (int column)`
- `const int currentSortKeyColumn () const`

- Q_INVOKABLE QVariant **data** (int row, int column) const
- Q_INVOKABLE QVariant **data** (const QModelIndex &index, int role=Qt::DisplayRole) const
- Q_INVOKABLE bool **fetch** (QString **tableName**="", QString **filter**="", QString **sort**="", QStringList **fields**=QStringList())
- Q_INVOKABLE SH_SqlDataFields * **field** (int i) const
- const QString **fields** () const
- Q_INVOKABLE int **fieldsCount** () const
- Q_INVOKABLE Qt::ItemFlags **flags** (const QModelIndex &index) const
- void **invalidateFilter** ()
- const bool **isEmpty** () const
- const QString **lastError** () const
- Q_INVOKABLE void **removeFilterKeyColumn** (int column)
- virtual Q_INVOKABLE QHash<int, QByteArray > **roleNames** () const
- void **setBooleanColumns** (QList<int> **boolCols**)
- Q_INVOKABLE bool **setData** (const QModelIndex &index, const QVariant &value, int role=Qt::EditRole)
- void **setNotNullColumns** (QList<int> **notNullCols**)
- void **setNullColumns** (QList<int> **nullCols**)
- void **setPasswordColumns** (QList<int> **passwordCols**)
- void **setReadOnlyColumns** (QList<int> **readonlyCols**)
- void **setSortKeyColumn** (int column)
- Q_INVOKABLE void **sort** (int column, Qt::SortOrder newOrder=Qt::AscendingOrder)
- const QString **tableName** () const

Fonctions membres protégées

- void **fillModel** ()
- bool **filterAcceptsRow** (int source_row, const QModelIndex &source_parent) const

Attributs protégés

- SH_SqlDataModel * **model**

Propriétés

- bool **empty**
- QString **fieldsList**
- QString **lastError**
- int **sortKeyColumn**
- QString **table**

4.25.1 Description détaillée

Définition à la ligne 14 du fichier [SH_ClientsTableModel.h](#).

4.25.2 Documentation des constructeurs et destructeur

4.25.2.1 SH_ClientsTableModel : :SH_ClientsTableModel (QObject * parent = 0)

Paramètres

<i>parent</i>	
---------------	--

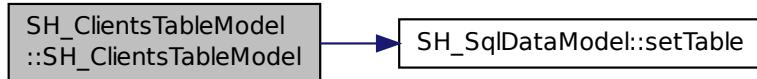
Définition à la ligne 9 du fichier [SH_ClientsTableModel.cpp](#).

Références [SH_ExtendedProxyModel](#) : :model, et [SH_SqlDataModel](#) : :setTable().

```

00009
00010     SH_ExtendedProxyModel (parent)
00011 {
00012     SH_ExtendedProxyModel::model->setTable ("CLIENTS");
00013 }
```

Voici le graphe d'appel pour cette fonction :



4.25.3 Documentation des fonctions membres

4.25.3.1 void SH_ExtendedProxyModel : :addFilterKeyColumn (int column) [inherited]

Définition à la ligne 259 du fichier [SH_ExtendedProxyModel.cpp](#).

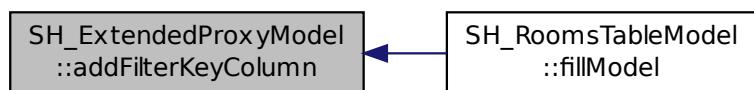
Références [SH_ExtendedProxyModel : :filters](#).

Référencé par [SH_RoomsTableModel : :fillModel\(\)](#).

```

00260 {
00261     this->filters.append\(column\);
00262 }
```

Voici le graphe des appels de cette fonction :



4.25.3.2 bool SH_ExtendedProxyModel : :containsFilterKeyColumn (int column) [inherited]

Définition à la ligne 225 du fichier [SH_ExtendedProxyModel.cpp](#).

Références [SH_ExtendedProxyModel : :filters](#).

```

00226 {
00227     return this->filters.contains\(column\);
00228 }
```

4.25.3.3 const int SH_ExtendedProxyModel : :currentSortKeyColumn () const [inline], [inherited]

Définition à la ligne 38 du fichier [SH_ExtendedSqlProxyModel.h](#).

Références [SH_ExtendedProxyModel : :sortIndex](#).

```
00038 {return this->sortIndex;}
```

4.25.3.4 QVariant SH_ExtendedProxyModel::data(int row, int column) const [inherited]

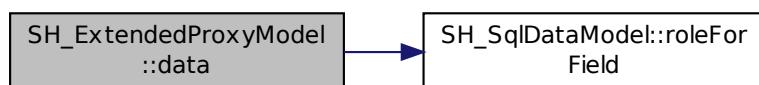
Définition à la ligne 269 du fichier [SH_ExtendedProxyModel.cpp](#).

Références [SH_ExtendedProxyModel::model](#), et [SH_SqlDataModel::roleForField\(\)](#).

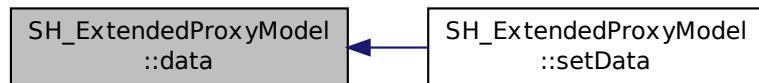
Référencé par [SH_ExtendedProxyModel::setData\(\)](#).

```
00270 {
00271     QModelIndex modelIndex = this->index(row, 0);
00272     return this->data(modelIndex, this->model->roleForField(column));
00273 }
```

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appels de cette fonction :



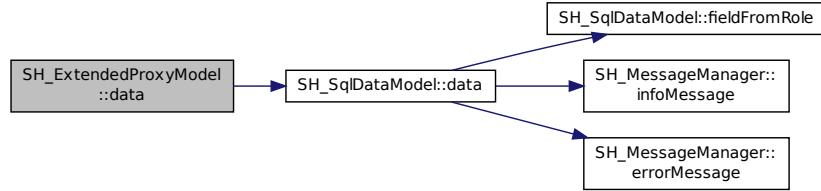
4.25.3.5 QVariant SH_ExtendedProxyModel::data(const QModelIndex & index, int role = Qt::DisplayRole) const [inherited]

Définition à la ligne 127 du fichier [SH_ExtendedProxyModel.cpp](#).

Références [SH_ExtendedProxyModel::booleanSet](#), [SH_SqlDataModel::data\(\)](#), [SH_ExtendedProxyModel::filters](#), [SH_ExtendedProxyModel::model](#), et [SH_ExtendedProxyModel::passwordSet](#).

```
00128 {
00129     if (index.isValid())
00130     {
00131         if (this->booleanSet.contains(role))
00132         {
00133             return index.data(Qt::EditRole).toBool() ? QVariant(Qt::Checked) : QVariant(Qt::Unchecked);
00134         }
00135         else if (this->passwordSet.contains(role))
00136         {
00137             return QVariant("****");
00138         }
00139         else if (!this->filters.contains(role))
00140         {
00141             QModelIndex source_index = QSortFilterProxyModel::mapToSource(index);
00142             if (source_index.isValid())
00143                 return this->model->data(source_index, role);
00144             }
00145         }
00146     }
00147     return QVariant();
00148 }
```

Voici le graphe d'appel pour cette fonction :



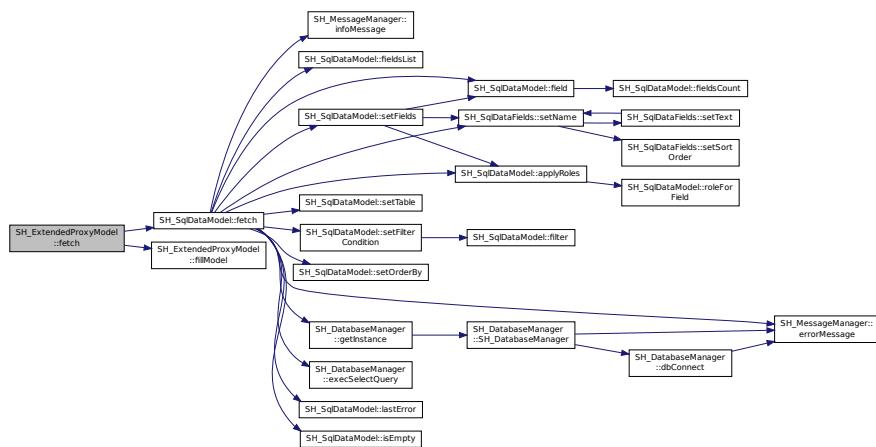
4.25.3.6 bool SH_ExtendedProxyModel ::fetch (QString tableName = "", QString filter = "", QString sort = "", QStringList fields = QStringList()) [inherited]

Définition à la ligne 280 du fichier [SH_ExtendedProxyModel.cpp](#).

Références `SH_SqlDataModel` : `:fetch()`, `SH_ExtendedProxyModel` : `:fillModel()`, et `SH_ExtendedProxyModel` : `:model`.

```
00281 {  
00282     bool fetched = this->model->fetch(tableName, filter, sort,  
        fields);  
00283     if (fetched)  
00284     {  
00285         this->fillModel();  
00286     }  
00287     this->setSourceModel(this->model);  
00288     return fetched;  
00289 }
```

Voici le graphe d'appel pour cette fonction :



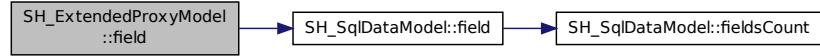
4.25.3.7 **Q_INVOKABLE SH_SqlDataFields* SH_ExtendedProxyModel ::field (int i) const [inline], [inherited]**

Définition à la ligne 82 du fichier [SH_ExtendedSqlProxyModel.h](#).

Références `SH_SqlDataModel` : `:field()`, et `SH_ExtendedProxyModel` : `:model`.

```
00082 { return this->model->field(i); }
```

Voici le graphe d'appel pour cette fonction :



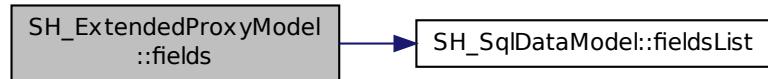
4.25.3.8 const QString SH_ExtendedProxyModel ::fields() const [inline], [inherited]

Définition à la ligne 52 du fichier [SH_ExtendedSqlProxyModel.h](#).

Références [SH_SqlDataModel ::fieldsList\(\)](#), et [SH_ExtendedProxyModel ::model](#).

```
00052 { if(this->model->fieldsList().isEmpty()){ return "*"; } else { return this->model->fieldsList().join(", "); } }
```

Voici le graphe d'appel pour cette fonction :



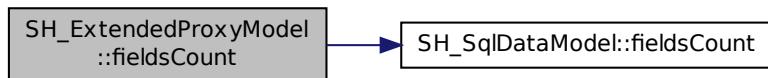
4.25.3.9 Q_INVOKABLE int SH_ExtendedProxyModel ::fieldsCount() const [inline], [inherited]

Définition à la ligne 89 du fichier [SH_ExtendedSqlProxyModel.h](#).

Références [SH_SqlDataModel ::fieldsCount\(\)](#), et [SH_ExtendedProxyModel ::model](#).

```
00089 { return this->model->fieldsCount(); }
```

Voici le graphe d'appel pour cette fonction :



4.25.3.10 void SH_ClientsTableModel : :fillModel () [protected], [virtual]

Implémente [SH_ExtendedProxyModel](#).

Définition à la ligne 21 du fichier [SH_ClientsTableModel.cpp](#).

```
00022 {
00023 }
```

4.25.3.11 bool SH_ExtendedProxyModel : :filterAcceptsRow (int source_row, const QModelIndex & source_parent) const [protected], [inherited]

Définition à la ligne 92 du fichier [SH_ExtendedProxyModel.cpp](#).

Références [SH_ExtendedProxyModel : :notNullSet](#), et [SH_ExtendedProxyModel : :nullSet](#).

```
00093 {
00094     Q_UNUSED(source_parent);
00095
00096     if (!this->notNullSet.isEmpty())
00097     {
00098         QSqlQueryModel *m = static_cast<QSqlQueryModel *>(sourceModel());
00099         foreach(int column, this->notNullSet)
00100         {
00101             if (m->record(source_row).isNull(column))
00102             {
00103                 return false;
00104             }
00105         }
00106     }
00107
00108     if (!this->nullSet.isEmpty())
00109     {
00110         QSqlQueryModel *m = static_cast<QSqlQueryModel *>(sourceModel());
00111         foreach(int column, this->nullSet)
00112         {
00113             if (!m->record(source_row).isNull(column))
00114             {
00115                 return false;
00116             }
00117         }
00118     }
00119     return true;
00120 }
```

4.25.3.12 Qt : :itemFlags SH_ExtendedProxyModel : :flags (const QModelIndex & index) const [inherited]

Définition à la ligne 179 du fichier [SH_ExtendedProxyModel.cpp](#).

Références [SH_ExtendedProxyModel : :booleanSet](#), et [SH_ExtendedProxyModel : :readonlySet](#).

```
00180 {
00181     if (!index.isValid())
00182     {
00183         return Qt::ItemIsEnabled;
00184     }
00185     if (!this->booleanSet.isEmpty())
00186     {
00187         return Qt::ItemIsUserCheckable | Qt::ItemIsSelectable | Qt::ItemIsEnabled;
00188     }
00189     else if (!this->readonlySet.isEmpty())
00190     {
00191         return Qt::ItemIsSelectable;
00192     }
00193     else
00194     {
00195         return QSortFilterProxyModel::flags(index);
00196     }
00197 }
00198 }
```

4.25.3.13 void SH_ExtendedProxyModel : :invalidateFilter () [inherited]

Définition à la ligne 205 du fichier [SH_ExtendedProxyModel.cpp](#).

Références [SH_ExtendedProxyModel : :filters](#).

```
00206 {
00207     this->filters.clear();
00208 }
```

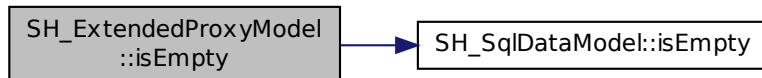
4.25.3.14 const bool SH_ExtendedProxyModel : :isEmpty () const [inline], [inherited]

Définition à la ligne 66 du fichier [SH_ExtendedSqlProxyModel.h](#).

Références [SH_SqlDataModel : :isEmpty\(\)](#), et [SH_ExtendedProxyModel : :model](#).

```
00066 { return this->model->isEmpty(); }
```

Voici le graphe d'appel pour cette fonction :



4.25.3.15 const QString SH_ExtendedProxyModel : :lastError () const [inline], [inherited]

Définition à la ligne 59 du fichier [SH_ExtendedSqlProxyModel.h](#).

Références [SH_SqlDataModel : :lastError](#), et [SH_ExtendedProxyModel : :model](#).

```
00059 { return this->model->lastError(); }
```

4.25.3.16 void SH_ExtendedProxyModel : :removeFilterKeyColumn (int column) [inherited]

Définition à la ligne 215 du fichier [SH_ExtendedProxyModel.cpp](#).

Références [SH_ExtendedProxyModel : :filters](#).

```
00216 {
00217     this->filters.removeAt(this->filters.indexOf(column));
00218 }
```

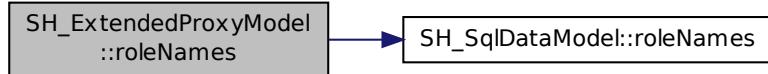
4.25.3.17 virtual Q_INVOKABLE QHash<int, QByteArray> SH_ExtendedProxyModel : :roleNames () const [inline], [virtual], [inherited]

Définition à la ligne 165 du fichier [SH_ExtendedSqlProxyModel.h](#).

Références [SH_ExtendedProxyModel : :model](#), et [SH_SqlDataModel : :roleNames\(\)](#).

```
00165 { return this->model->roleNames(); }
```

Voici le graphe d'appel pour cette fonction :



4.25.3.18 void SH_ExtendedProxyModel ::setBooleanColumns (QList< int > boolCols) [inherited]

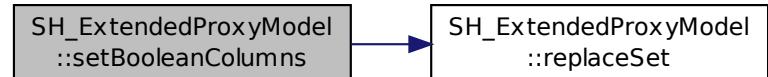
Définition à la ligne 41 du fichier [SH_ExtendedProxyModel.cpp](#).

Références [SH_ExtendedProxyModel ::booleanSet](#), et [SH_ExtendedProxyModel ::replaceSet\(\)](#).

```

00041
00042     replaceSet(this->booleanSet, boolCols);
00043 }
```

Voici le graphe d'appel pour cette fonction :



4.25.3.19 bool SH_ExtendedProxyModel ::setData (const QModelIndex & index, const QVariant & value, int role = Qt::EditRole) [inherited]

Définition à la ligne 156 du fichier [SH_ExtendedProxyModel.cpp](#).

Références [SH_ExtendedProxyModel ::booleanSet](#), et [SH_ExtendedProxyModel ::data\(\)](#).

```

00157 {
00158     if (!index.isValid())
00159         return false;
00160
00161     if (this->booleanSet.contains(role))
00162     {
00163         QVariant data = (value.toInt() == Qt::Checked) ? QVariant(1) : QVariant(0);
00164         return QSortFilterProxyModel::setData(index, data, role);
00165     }
00166     else
00167     {
00168         return QSortFilterProxyModel::setData(index, value, role);
00169     }
00170
00171 }
```

Voici le graphe d'appel pour cette fonction :



4.25.3.20 void SH_ExtendedProxyModel ::setNotNullColumns (QList< int > *notNullCols*) [inherited]

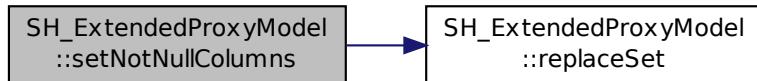
Définition à la ligne 80 du fichier [SH_ExtendedProxyModel.cpp](#).

Références [SH_ExtendedProxyModel ::notNullSet](#), et [SH_ExtendedProxyModel ::replaceSet\(\)](#).

```

00080
00081     if (sourceModel()->inherits("QSqlQueryModel")) {
00082         replaceSet(this->notNullSet, notNullCols);
00083     }
00084 }
```

Voici le graphe d'appel pour cette fonction :



4.25.3.21 void SH_ExtendedProxyModel ::setNullColumns (QList< int > *nullCols*) [inherited]

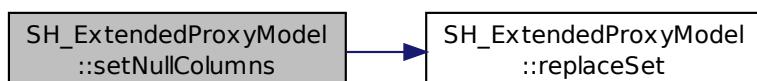
Définition à la ligne 68 du fichier [SH_ExtendedProxyModel.cpp](#).

Références [SH_ExtendedProxyModel ::nullSet](#), et [SH_ExtendedProxyModel ::replaceSet\(\)](#).

```

00068
00069     if (sourceModel()->inherits("QSqlQueryModel")) {
00070         replaceSet(this->nullSet, nullCols);
00071     }
00072 }
```

Voici le graphe d'appel pour cette fonction :



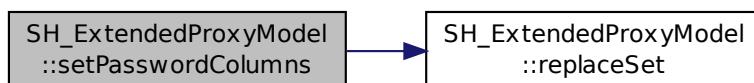
4.25.3.22 void SH_ExtendedProxyModel : setPasswordColumns (QList< int > passwordCols) [inherited]

Définition à la ligne 59 du fichier [SH_ExtendedProxyModel.cpp](#).

Références [SH_ExtendedProxyModel : :passwordSet](#), et [SH_ExtendedProxyModel : :replaceSet\(\)](#).

```
00059
00060     replaceSet(this->passwordSet, passwordCols);
00061 }
```

Voici le graphe d'appel pour cette fonction :



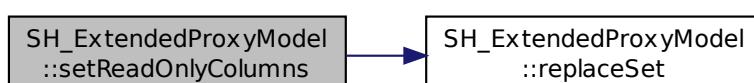
4.25.3.23 void SH_ExtendedProxyModel : :setReadOnlyColumns (QList< int > readonlyCols) [inherited]

Définition à la ligne 50 du fichier [SH_ExtendedProxyModel.cpp](#).

Références [SH_ExtendedProxyModel : :readonlySet](#), et [SH_ExtendedProxyModel : :replaceSet\(\)](#).

```
00050
00051     replaceSet(this->readonlySet, readonlyCols);
00052 }
```

Voici le graphe d'appel pour cette fonction :



4.25.3.24 void SH_ExtendedProxyModel : :setSortKeyColumn (int column) [inherited]

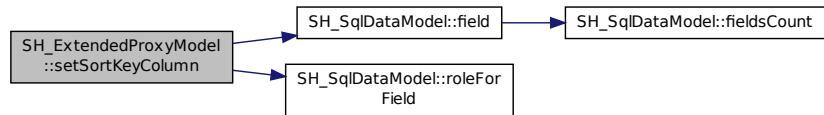
Définition à la ligne 246 du fichier [SH_ExtendedProxyModel.cpp](#).

Références [SH_SqlDataModel : :field\(\)](#), [SH_ExtendedProxyModel : :model](#), [SH_SqlDataModel : :roleForField\(\)](#), [SH_ExtendedProxyModel : :sortChanged\(\)](#), [SH_ExtendedProxyModel : :sortIndex](#), et [SH_SqlDataFields : :sortOrder](#).

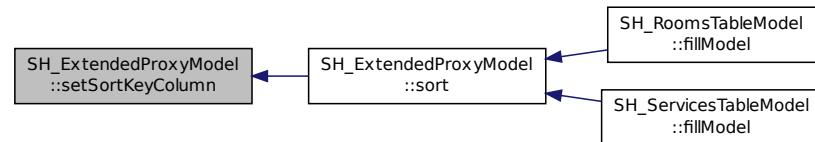
Référencé par [SH_ExtendedProxyModel : :sort\(\)](#).

```
00247 {
00248     this->sortIndex = column;
00249     QSortFilterProxyModel::setSortRole(this->model->roleForField(column));
00250     QSortFilterProxyModel::sort(0, this->model->field(column)->
00251         sortOrder());
00251     emit sortChanged();
00252 }
```

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appels de cette fonction :



4.25.3.25 void SH_ExtendedProxyModel ::sort (int column, Qt ::SortOrder newOrder = Qt ::AscendingOrder) [inherited]

Définition à la ligne 235 du fichier [SH_ExtendedProxyModel.cpp](#).

Références [SH_SqlDataModel ::field\(\)](#), [SH_ExtendedProxyModel ::model\(\)](#), [SH_ExtendedProxyModel ::setSortKeyColumn\(\)](#), et [SH_SqlDataFields ::setSortOrder\(\)](#).

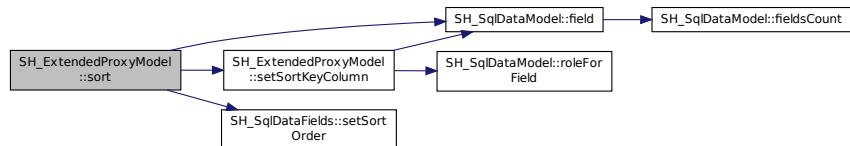
Référencé par [SH_RoomsTableModel ::fillModel\(\)](#), et [SH_ServicesTableModel ::fillModel\(\)](#).

```

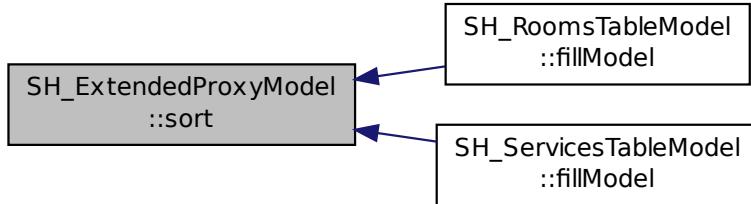
00236 {
00237     this->model->field(column)->setSortOrder(newOrder);
00238     SH_ExtendedProxyModel::setSortKeyColumn(column);
00239 }

```

Voici le graphe d'appel pour cette fonction :



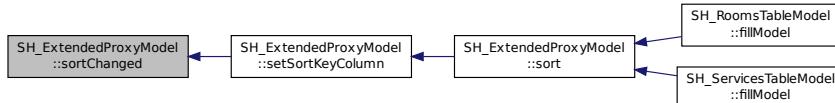
Voici le graphe des appelants de cette fonction :



4.25.3.26 void SH_ExtendedProxyModel ::sortChanged() [signal], [inherited]

Référencé par [SH_ExtendedProxyModel ::setSortKeyColumn\(\)](#).

Voici le graphe des appelants de cette fonction :



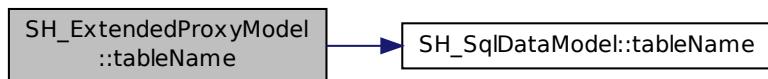
4.25.3.27 const QString SH_ExtendedProxyModel ::tableName() const [inline], [inherited]

Définition à la ligne 45 du fichier [SH_ExtendedSqlProxyModel.h](#).

Références [SH_ExtendedProxyModel ::model](#), et [SH_SqlDataModel ::tableName\(\)](#).

```
00045 { return this->model->tableName(); }
```

Voici le graphe d'appel pour cette fonction :



4.25.4 Documentation des données membres

4.25.4.1 SH_SqlDataModel* SH_ExtendedProxyModel ::model [protected], [inherited]

Définition à la ligne 241 du fichier [SH_ExtendedSqlProxyModel.h](#).

Référencé par [SH_ExtendedProxyModel](#) : `:data()`, [SH_ExtendedProxyModel](#) : `:fetch()`, [SH_ExtendedProxyModel](#) : `:field()`, [SH_ExtendedProxyModel](#) : `:fields()`, [SH_ExtendedProxyModel](#) : `:fieldsCount()`, [SH_BillingsTableModel](#) : `:fillModel()`, [SH_RoomsTableModel](#) : `:fillModel()`, [SH_BookingsTableModel](#) : `:fillModel()`, [SH_ExtendedProxyModel](#) : `:isEmpty()`, [SH_ExtendedProxyModel](#) : `:lastError()`, [SH_ExtendedProxyModel](#) : `:roleNames()`, [SH_ExtendedProxyModel](#) : `:setSortKeyColumn()`, [SH_BillingsTableModel](#) : `:SH_BillingsTableModel()`, [SH_BillsTableModel](#) : `:SH_BillsTableModel()`, [SH_BookingsTableModel](#) : `:SH_BookingsTableModel()`, [SH_ClientsTableModel](#) : `:SH_ClientsTableModel()`, [SH_ExtendedProxyModel](#) : `:SH_ExtendedProxyModel()`, [SH_GroupsTableModel](#) : `:SH_GroupsTableModel()`, [SH_RoomsTableModel](#) : `:SH_RoomsTableModel()`, [SH_ServicesTableModel](#) : `:SH_ServicesTableModel()`, [SH_ExtendedProxyModel](#) : `:sort()`, et [SH_ExtendedProxyModel](#) : `:tableName()`.

4.25.5 Documentation des propriétés

4.25.5.1 `bool SH_ExtendedProxyModel::empty [read], [inherited]`

Définition à la ligne 21 du fichier [SH_ExtendedSqlProxyModel.h](#).

4.25.5.2 `QString SH_ExtendedProxyModel::fieldsList [read], [inherited]`

Définition à la ligne 18 du fichier [SH_ExtendedSqlProxyModel.h](#).

4.25.5.3 `QString SH_ExtendedProxyModel::lastError [read], [inherited]`

Définition à la ligne 19 du fichier [SH_ExtendedSqlProxyModel.h](#).

4.25.5.4 `int SH_ExtendedProxyModel::sortKeyColumn [read], [write], [inherited]`

Définition à la ligne 20 du fichier [SH_ExtendedSqlProxyModel.h](#).

4.25.5.5 `QString SH_ExtendedProxyModel::table [read], [inherited]`

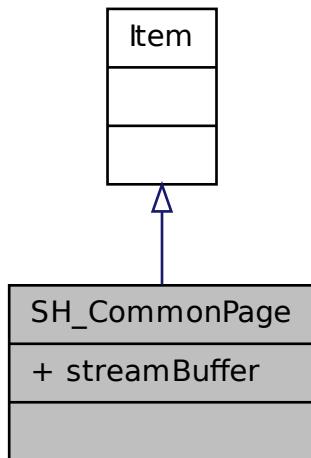
Définition à la ligne 17 du fichier [SH_ExtendedSqlProxyModel.h](#).

La documentation de cette classe a été générée à partir des fichiers suivants :

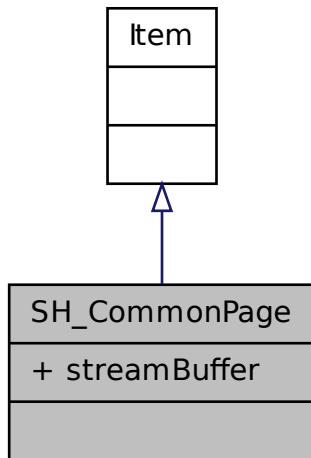
- [models/SH_ClientsTableModel.h](#)
- [models/SH_ClientsTableModel.cpp](#)

4.26 Référence de la classe SH_CommonPage

Graphe d'héritage de SH_CommonPage :



Graphe de collaboration de SH_CommonPage :



Signaux

- void `cancelProcess ()`
- void `cancelReplace ()`
- void `confirm ()`
- void `keySelected (string selectedKey)`

- void `quit()`
- void `reload()`
- void `replace(string field)`
- void `selected(string selectedItem)`
- void `validate()`

Propriétés

- QtObject `streamBuffer`

4.26.1 Description détaillée

Définition à la ligne 4 du fichier [SH_CommonPage.qml](#).

4.26.2 Documentation des fonctions membres

4.26.2.1 void `SH_CommonPage::cancelProcess()` [signal]

4.26.2.2 void `SH_CommonPage::cancelReplace()` [signal]

4.26.2.3 void `SH_CommonPage::confirm()` [signal]

4.26.2.4 void `SH_CommonPage::keySelected(string selectedKey)` [signal]

4.26.2.5 void `SH_CommonPage::quit()` [signal]

4.26.2.6 void `SH_CommonPage::reload()` [signal]

4.26.2.7 void `SH_CommonPage::replace(string field)` [signal]

4.26.2.8 void `SH_CommonPage::selected(string selectedItem)` [signal]

4.26.2.9 void `SH_CommonPage::validate()` [signal]

4.26.3 Documentation des propriétés

4.26.3.1 QtObject `SH_CommonPage::streamBuffer`

Définition à la ligne 7 du fichier [SH_CommonPage.qml](#).

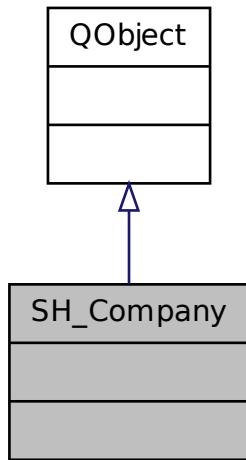
La documentation de cette classe a été générée à partir du fichier suivant :

- [views/qml/SH_CommonPage.qml](#)

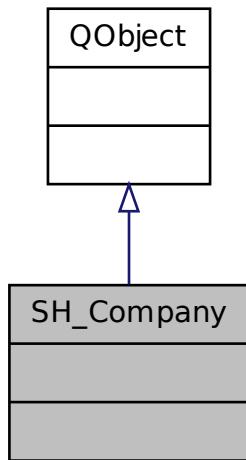
4.27 Référence de la classe SH_Company

```
#include <SH_Company.h>
```

Graphe d'héritage de SH_Company :



Graphe de collaboration de SH_Company :



4.27.1 Description détaillée

Définition à la ligne 12 du fichier [SH_Company.h](#).

La documentation de cette classe a été générée à partir du fichier suivant :

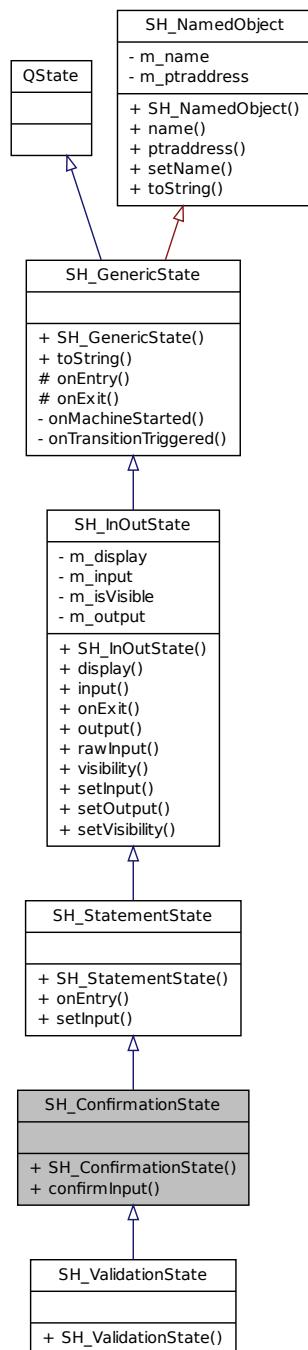
- [models/SH_Company.h](#)

4.28 Référence de la classe SH_ConfirmationState

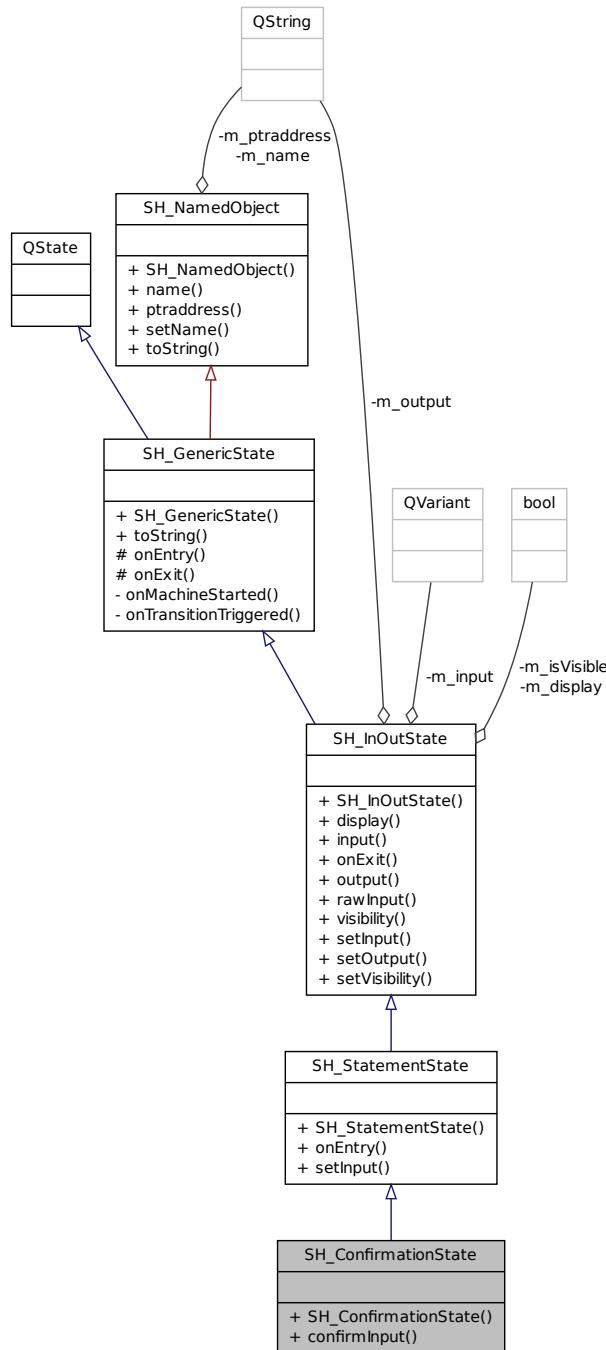
La class ConfirmationState représente un état dans lequel le système attend que l'utilisateur appuie sur une touche de confirmation.

```
#include <SH_ConfirmationState.h>
```

Graphe d'héritage de SH_ConfirmationState :



Graphe de collaboration de SH_ConfirmationState :



Connecteurs publics

- void `confirmInput()`
- virtual void `setOutput` (const `QString` &`output`)
- virtual void `setVisibility` (`bool` `isVisible`)

Signaux

- void `next ()`
- void `resendInput (QVariant input)`
- void `sendOutput (QVariant output)`

Fonctions membres publiques

- `SH_ConfirmationState (QString output, QString name, QState *parent=0)`
- void `display (bool canDisplay)`
- virtual QVariant `input () const`
- void `onEntry (QEvent *event)`
- void `onExit (QEvent *event)`
- virtual QString `output () const`
- virtual QVariant `rawInput () const`
- void `setInput (const QVariant &input)`
- QString `toString ()`
- bool `visibility ()`

4.28.1 Description détaillée

La class ConfirmationState représente un état dans lequel le système attend que l'utilisateur appuie sur une touche de confirmation.

L'état ConfirmationState correspond à un état de quasi-déclaration où la seule entrée acceptée de l'utilisateur est une confirmation.

Définition à la ligne 13 du fichier `SH_ConfirmationState.h`.

4.28.2 Documentation des constructeurs et destructeur

4.28.2.1 SH_ConfirmationState : :SH_ConfirmationState (`QString output, QString name, QState * parent = 0`)

Paramètres

<code>output</code>	
<code>name</code>	
<code>parent</code>	

Définition à la ligne 10 du fichier `SH_ConfirmationState.cpp`.

```
00010
00011     SH_StatementState(output, name, parent)
00012 {
00013     qDebug() << "confirmation !";
00014 }
```

:

4.28.3 Documentation des fonctions membres

4.28.3.1 void SH_ConfirmationState : :confirmInput () [slot]

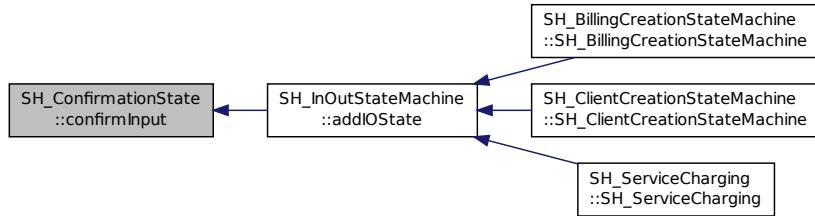
Définition à la ligne 21 du fichier `SH_ConfirmationState.cpp`.

Références `SH_GenericState : :next()`.

Référencé par `SH_InOutStateMachine : :addIOState()`.

```
00022 {
00023     emit next();
00024 }
```

Voici le graphe des appelants de cette fonction :



4.28.3.2 void SH_InOutState ::display (bool canDisplay) [inherited]

Définition à la ligne 95 du fichier [SH_IOState.cpp](#).

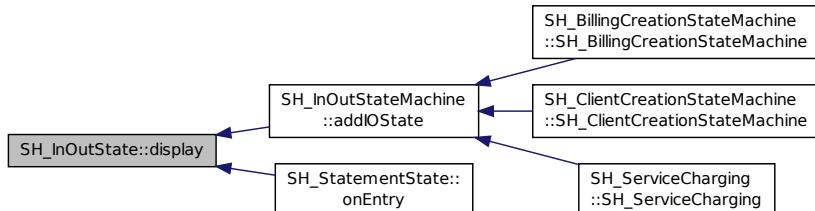
Références [SH_InOutState ::m_display](#), [SH_InOutState ::m_isVisible](#), [SH_InOutState ::m_output](#), et [SH_InOutState ::sendOutput\(\)](#).

Référencé par [SH_InOutStateMachine ::addIOState\(\)](#), et [SH_StatementState ::onEntry\(\)](#).

```

00096 {
00097     m_display=canDisplay;
00098     if(m_display && !m_output.isEmpty() && m_isVisible) {
00099         qDebug() << "resalut !" << QVariant(m_output);
00100         emit sendOutput(QVariant(m_output));
00101     }
00102 }
```

Voici le graphe des appelants de cette fonction :



4.28.3.3 QVariant SH_InOutState ::input() const [virtual], [inherited]

Définition à la ligne 20 du fichier [SH_IOState.cpp](#).

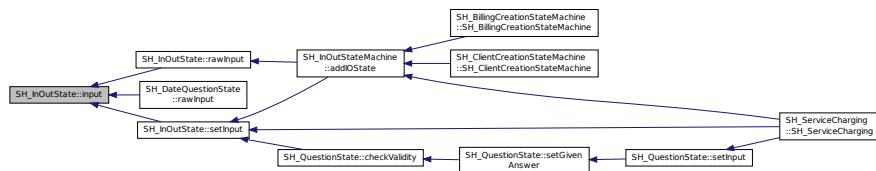
Références [SH_InOutState ::m_input](#).

Référencé par [SH_InOutState ::rawInput\(\)](#), [SH_DateQuestionState ::rawInput\(\)](#), et [SH_InOutState ::setInput\(\)](#).

```

00021 {
00022     return m_input;
00023 }
```

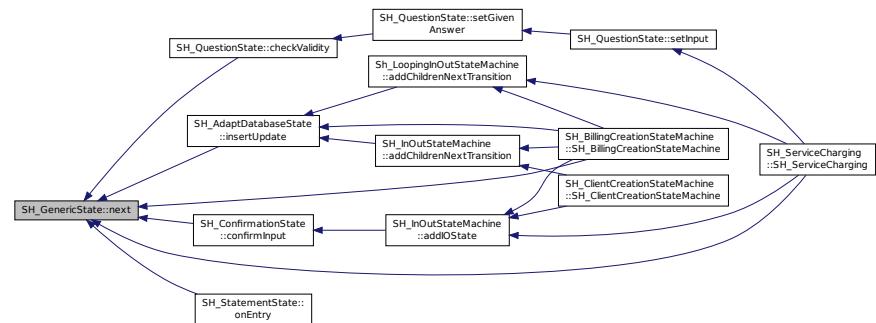
Voici le graphe des appelants de cette fonction :



4.28.3.4 void SH_GenericState ::next() [signal], [inherited]

Référencé par [SH_QuestionState ::checkValidity\(\)](#), [confirmInput\(\)](#), [SH_AdaptDatabaseState ::insertUpdate\(\)](#), [SH_StatementState ::onEntry\(\)](#), [SH_BillingCreationStateMachine ::SH_BillingCreationStateMachine\(\)](#), et [SH_ServiceCharging ::SH_ServiceCharging\(\)](#).

Voici le graphe des appelants de cette fonction :



4.28.3.5 void SH_StatementState ::onEntry(QEvent * event) [inherited]

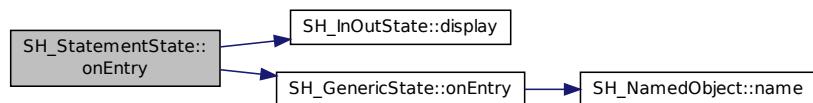
Définition à la ligne 33 du fichier [SH_StatementState.cpp](#).

Références [SH_InOutState ::display\(\)](#), [SH_GenericState ::next\(\)](#), et [SH_GenericState ::onEntry\(\)](#).

```

00034 {
00035     SH_GenericState::onEntry(event);
00036     display(true);
00037     emit next();
00038 }
  
```

Voici le graphe d'appel pour cette fonction :



4.28.3.6 void SH_InOutState ::onExit (QEvent * event) [inherited]

Définition à la ligne 110 du fichier [SH_IOState.cpp](#).

Références [SH_InOutState ::m_input](#), [SH_InOutState ::m_isVisible](#), [SH_GenericState ::onExit\(\)](#), et [SH_InOutState ::resendInput\(\)](#).

```
00111 {
00112     SH_GenericState::onExit(event);
00113     if(m_isVisible) {
00114         emit resendInput(m_input);
00115     }
00116 }
```

Voici le graphe d'appel pour cette fonction :



4.28.3.7 QString SH_InOutState ::output () const [virtual], [inherited]

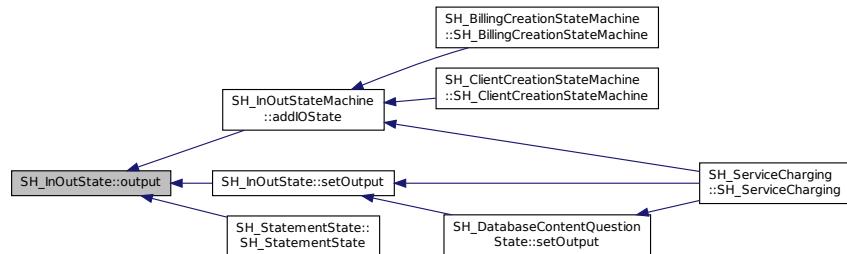
Définition à la ligne 56 du fichier [SH_IOState.cpp](#).

Références [SH_InOutState ::m_output](#).

Référencé par [SH_InOutStateMachine ::addIOState\(\)](#), [SH_InOutState ::setOutput\(\)](#), et [SH_StatementState ::SH_StatementState\(\)](#).

```
00057 {
00058     return m_output;
00059 }
```

Voici le graphe des appels de cette fonction :



4.28.3.8 QVariant SH_InOutState ::rawInput () const [virtual], [inherited]

Réimplémentée dans [SH_DateQuestionState](#), et [SH_DatabaseContentQuestionState](#).

Définition à la ligne 30 du fichier [SH_IOState.cpp](#).

Références [SH_InOutState ::input\(\)](#).

Référencé par [SH_InOutStateMachine ::addIOState\(\)](#).

```

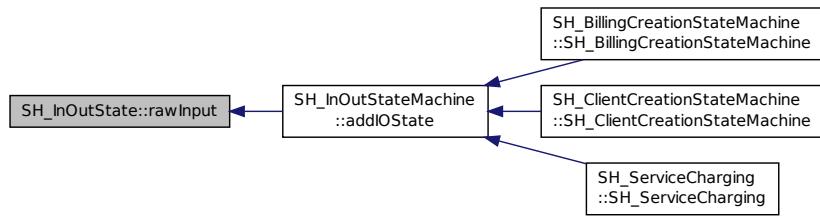
00031 {
00032     return input();
00033 }

```

Voici le graphe d'appel pour cette fonction :



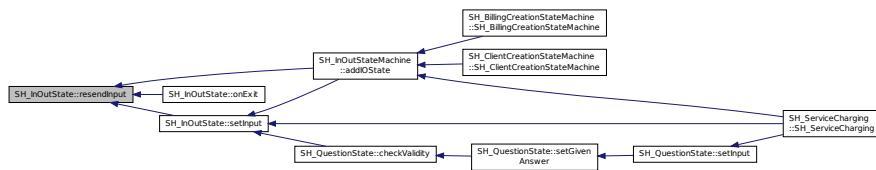
Voici le graphe des appelants de cette fonction :



4.28.3.9 void SH_InOutState ::resendInput (QVariant *input*) [signal], [inherited]

Référencé par [SH_InOutStateMachine ::addIOState\(\)](#), [SH_InOutState ::onExit\(\)](#), et [SH_InOutState ::setInput\(\)](#).

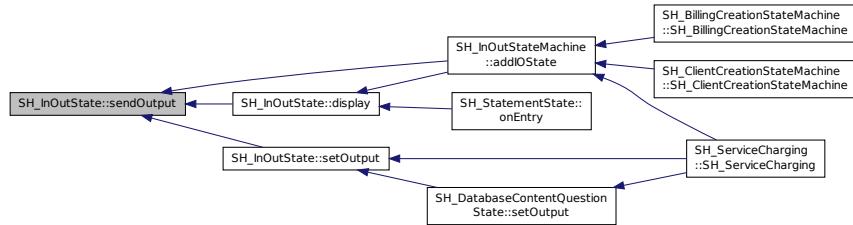
Voici le graphe des appelants de cette fonction :



4.28.3.10 void SH_InOutState ::sendOutput (QVariant *output*) [signal], [inherited]

Référencé par [SH_InOutStateMachine ::addIOState\(\)](#), [SH_InOutState ::display\(\)](#), et [SH_InOutState ::setOutput\(\)](#).

Voici le graphe des appelants de cette fonction :



4.28.3.11 void SH_StatementState ::setInput (const QVariant & input) [virtual], [inherited]

Réimplémentée à partir de [SH_InOutState](#).

Définition à la ligne [21](#) du fichier [SH_StatementState.cpp](#).

```

00022 {
00023     Q_UNUSED(input);
00024     /*DO NOTHING*/
00025 }
  
```

4.28.3.12 void SH_InOutState ::setOutput (const QString & output) [virtual], [slot], [inherited]

Réimplémentée dans [SH_DatabaseContentQuestionState](#).

Définition à la ligne [68](#) du fichier [SH_IOState.cpp](#).

Références [SH_InOutState ::m_isVisible](#), [SH_InOutState ::m_output](#), [SH_InOutState ::output\(\)](#), et [SH_InOutState ::sendOutput\(\)](#).

Référencé par [SH_DatabaseContentQuestionState ::setOutput\(\)](#), et [SH_ServiceCharging ::SH_ServiceCharging\(\)](#).

```

00069 {
00070     m_output = output;
00071     if(m_isVisible) {
00072         emit sendOutput(QVariant(m_output));
00073     }
00074 }
  
```

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :



4.28.3.13 void SH_InOutState ::setVisibility (bool isVisible) [virtual], [slot], [inherited]

Définition à la ligne 81 du fichier [SH_IOState.cpp](#).

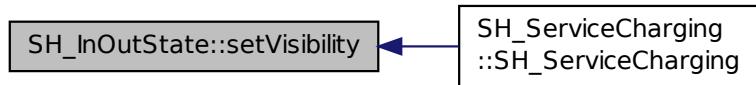
Références [SH_InOutState ::m_isVisible](#).

Référencé par [SH_ServiceCharging ::SH_ServiceCharging\(\)](#).

```

00082 {
00083     m_isVisible = isVisible;
00084 }
```

Voici le graphe des appelants de cette fonction :



4.28.3.14 QString SH_GenericState ::toString () [virtual], [inherited]

Réimplémentée à partir de [SH_NamedObject](#).

Définition à la ligne 21 du fichier [SH_GenericDebugableState.cpp](#).

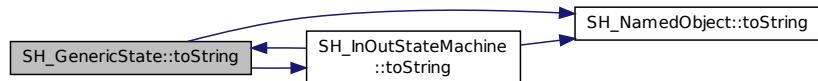
Références [SH_NamedObject ::toString\(\)](#), et [SH_InOutStateMachine ::toString\(\)](#).

Référencé par [SH_InOutStateMachine ::addChildsNextTransition\(\)](#), [SH_DateQuestionState ::rawInput\(\)](#), et [SH_InOutStateMachine ::toString\(\)](#).

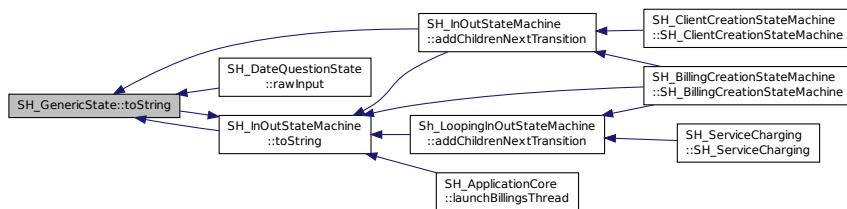
```

00022 {
00023     QStateMachine* machine = this->machine();
00024     SH_InOutStateMachine* mach = qobject_cast<
00025         SH_InOutStateMachine *>(machine);
00026     if(mach) {
00027         return SH_NamedObject::toString() + " [in "+mach->
00028             toString()+"] ";
00029     } else {
00030         return SH_NamedObject::toString();
00031     }
00032 }
```

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appels de cette fonction :



4.28.3.15 bool SH_InOutState ::visibility() [inherited]

Définition à la ligne 91 du fichier [SH_IOState.cpp](#).

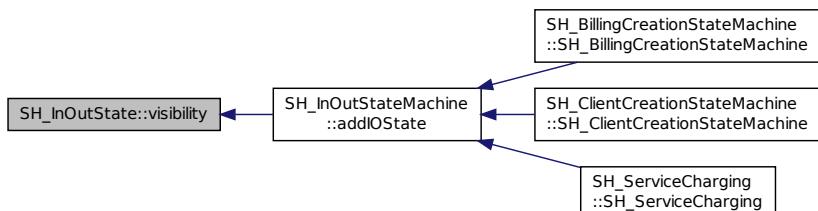
Références [SH_InOutState ::m_isVisible](#).

Référencé par [SH_InOutStateMachine ::addIOState\(\)](#).

```

00091
00092     return m_isVisible;
00093 }
  
```

Voici le graphe des appels de cette fonction :

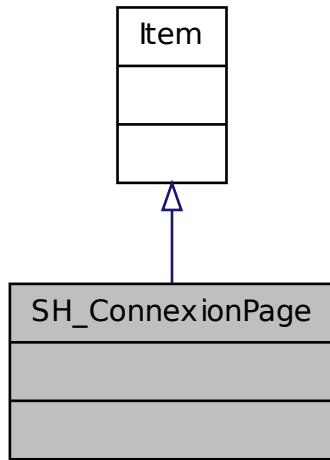


La documentation de cette classe a été générée à partir des fichiers suivants :

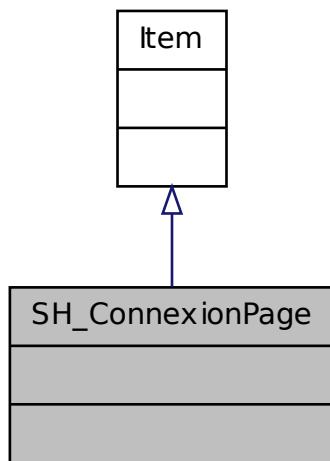
- logic/[SH_ConfirmationState.h](#)
- logic/[SH_ConfirmationState.cpp](#)

4.29 Référence de la classe SH_ConexionPage

Graphe d'héritage de SH_ConexionPage :



Graphe de collaboration de SH_ConexionPage :



Signaux

- void `checkUsername` (string name)
- void `loggedin` ()
- void `login` (string login, string password)
- void `reload` ()

4.29.1 Description détaillée

Définition à la ligne 4 du fichier [SH_ConexionPage.qml](#).

4.29.2 Documentation des fonctions membres

4.29.2.1 `void SH_ConexionPage ::checkUsername (string name) [signal]`

4.29.2.2 `void SH_ConexionPage ::loggedin () [signal]`

4.29.2.3 `void SH_ConexionPage ::login (string login, string password) [signal]`

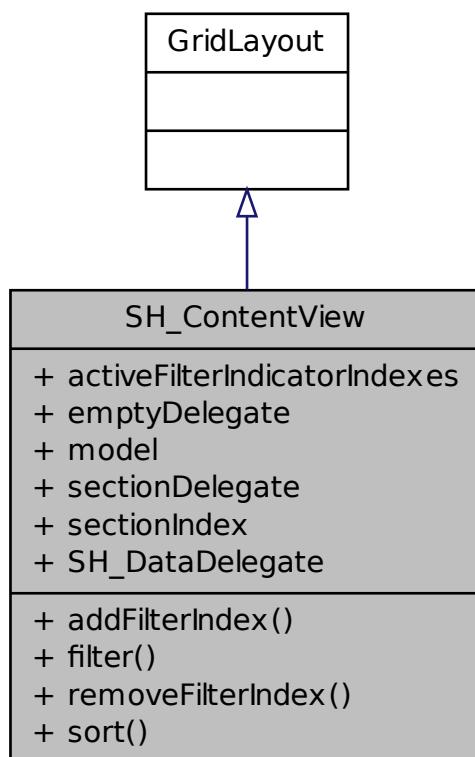
4.29.2.4 `void SH_ConexionPage ::reload () [signal]`

La documentation de cette classe a été générée à partir du fichier suivant :

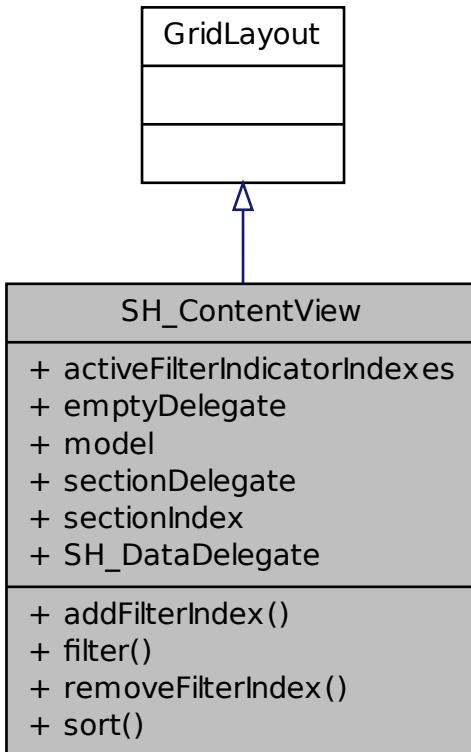
- [views/qml/SH_ConexionPage.qml](#)

4.30 Référence de la classe SH_ContentView

Graphe d'héritage de SH_ContentView :



Graphe de collaboration de SH_ContentView :



Signaux

- void **selected** (string selectedItem)

Fonctions membres publiques

- void **addFilterIndex** (index)
- void **filter** (index, remove)
- void **removeFilterIndex** (index)
- void **sort** (index)

Propriétés

- variant **activeFilterIndicatorIndexes**
- string **emptyDelegate**
- alias **model**
- string **sectionDelegate**
- alias **sectionIndex**
- string **SH_DataDelegate**

4.30.1 Description détaillée

Définition à la ligne 4 du fichier [SH_ContentView.qml](#).

4.30.2 Documentation des fonctions membres

- 4.30.2.1 void SH_ContentView : :addFilterIndex(index)
- 4.30.2.2 void SH_ContentView : :filter(index , remove)
- 4.30.2.3 void SH_ContentView : :removeFilterIndex(index)
- 4.30.2.4 void SH_ContentView : :selected(string selectedItem) [signal]
- 4.30.2.5 void SH_ContentView : :sort(index)

4.30.3 Documentation des propriétés

- 4.30.3.1 variant SH_ContentView : :activeFilterIndicatorIndexes

Définition à la ligne 13 du fichier [SH_ContentView.qml](#).

- 4.30.3.2 string SH_ContentView : :emptyDelegate

Définition à la ligne 9 du fichier [SH_ContentView.qml](#).

- 4.30.3.3 alias SH_ContentView : :model

Définition à la ligne 17 du fichier [SH_ContentView.qml](#).

- 4.30.3.4 string SH_ContentView : :sectionDelegate

Définition à la ligne 11 du fichier [SH_ContentView.qml](#).

- 4.30.3.5 alias SH_ContentView : :sectionIndex

Définition à la ligne 15 du fichier [SH_ContentView.qml](#).

- 4.30.3.6 string SH_ContentView : :SH_DataDelegate

Définition à la ligne 7 du fichier [SH_ContentView.qml](#).

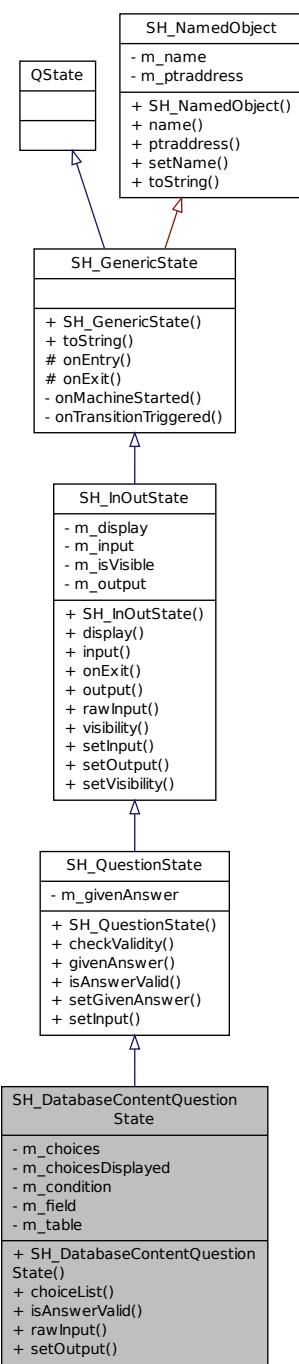
La documentation de cette classe a été générée à partir du fichier suivant :

- views/qml/[SH_ContentView.qml](#)

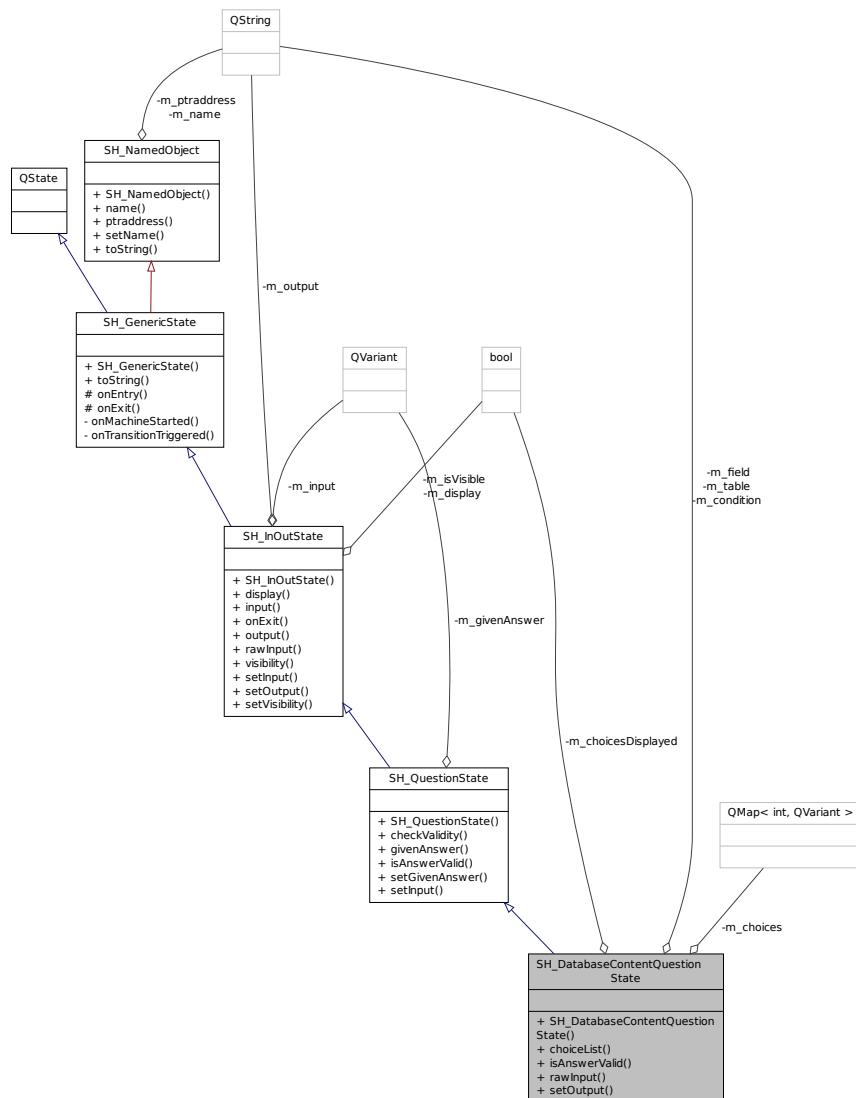
4.31 Référence de la classe SH_DatabaseContentQuestionState

```
#include <SH_DatabaseContentQuestionState.h>
```

Graphe d'héritage de SH_DatabaseContentQuestionState :



Graphe de collaboration de SH_DatabaseContentQuestionState :



Connecteurs publics

- `virtual void setVisibility (bool isVisible)`

Signaux

- `void answerInvalid ()`
 answerInvalid
- `void answerValid ()`
 answerValid
- `void displayChoiceList ()`
- `void next ()`
- `void resendInput (QVariant input)`
- `void sendOutput (QVariant output)`

Fonctions membres publiques

- `SH_DatabaseContentQuestionState` (QString *question*, QString *name*, QString *databaseTable*, QString *tableField*, QString *databaseCondition*=`" "`, QState **parent*=0)
- bool `checkValidity` ()
- QMap< int, QVariant > `choiceList` ()
- void `display` (bool *canDisplay*)
- virtual QVariant `givenAnswer` () const
- virtual QVariant `input` () const
- virtual bool `isValid` (const QVariant &*givenAnswer*)
- void `onExit` (QEvent **event*)
- virtual QString `output` () const
- virtual QVariant `rawInput` () const
- virtual void `setGivenAnswer` (const QVariant &*givenAnswer*)
- virtual void `setInput` (const QVariant &*input*)
- void `setOutput` (const QString &*output*)
- QString `toString` ()
- bool `visibility` ()

Fonctions membres protégées

- void `onEntry` (QEvent **event*)

Attributs privés

- QMap< int, QVariant > *m_choices*
- m_choices*
- bool *m_choicesDisplayed*
- m_choicesDisplayed*
- QString *m_condition*
- m_condition*
- QString *m_field*
- m_field*
- QString *m_table*
- m_table*

4.31.1 Description détaillée

Définition à la ligne 10 du fichier `SH_DatabaseContentQuestionState.h`.

4.31.2 Documentation des constructeurs et destructeur

4.31.2.1 `SH_DatabaseContentQuestionState` : `:SH_DatabaseContentQuestionState (QString question, QString name, QString databaseTable, QString tableField, QString databaseCondition = " ", QState * parent = 0)`

Paramètres

<i>question</i>	
<i>name</i>	
<i>databaseTable</i>	
<i>tableField</i>	
<i>databaseCondition</i>	
<i>parent</i>	

Définition à la ligne 10 du fichier `SH_DatabaseContentQuestionState.cpp`.

Références `SH_SqlDataModel` : `:datas()`, `SH_SqlDataModel` : `:fetch()`, *m_choices*, *m_condition*, *m_field*, et *m_table*.

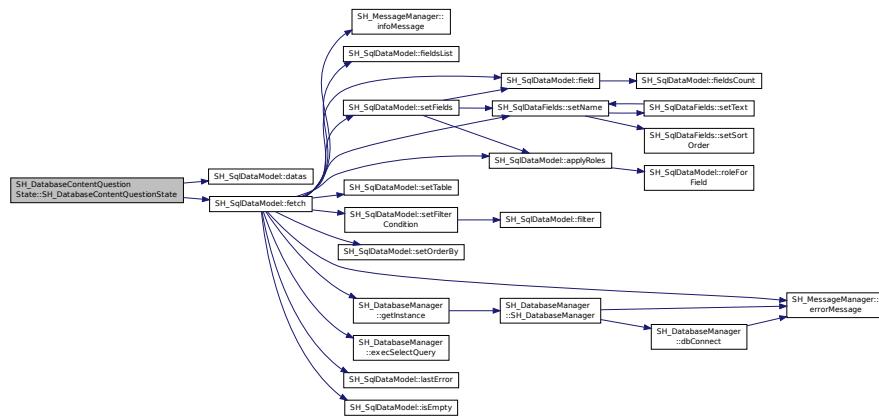
```
00010
00011      SH_QuestionState(question, name, parent), m_table(databaseTable),
           m_condition(databaseCondition), m_field(tableField)
```

```

00012 {
00013     qDebug() << "multiple choice list with datas from " << databaseTable << "!";
00014     SH_SqlDataModel *sqlDatas = new SH_SqlDataModel();
00015     QStringList fields;
00016     fields << "ID" << m_field;
00017     sqlDatas->fetch(m_table, m_condition, "", fields);
00018     QVariantMap results = sqlDatas->datas();
00019     QVariantList idValues = results.values("ID");
00020     QVariantList fieldsValues = results.values(m_field);
00021     for(int i = 0; i < idValues.length(); i++) {
00022         qDebug() << "new choice " << idValues.at(i).toString() << ":" << fieldsValues.at(i).toString();
00023         m_choices.insert(idValues.at(i).toInt(), fieldsValues.at(i));
00024     }
00025 }

```

Voici le graphe d'appel pour cette fonction :



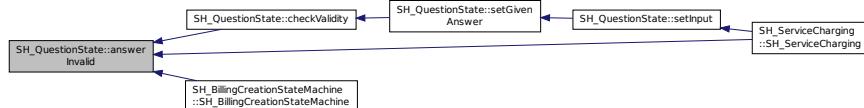
4.31.3 Documentation des fonctions membres

4.31.3.1 void SH_QuestionState ::answerInvalid() [signal], [inherited]

answerInvalid

Référencé par [SH_QuestionState ::checkValidity\(\)](#), [SH_BillingCreationStateMachine ::SH_BillingCreationStateMachine\(\)](#), et [SH_ServiceCharging ::SH_ServiceCharging\(\)](#).

Voici le graphe des appels de cette fonction :

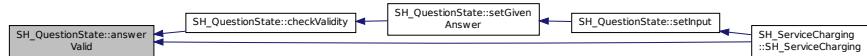


4.31.3.2 void SH_QuestionState ::answerValid() [signal], [inherited]

answerValid

Référencé par [SH_QuestionState ::checkValidity\(\)](#), et [SH_ServiceCharging ::SH_ServiceCharging\(\)](#).

Voici le graphe des appelants de cette fonction :



4.31.3.3 bool SH_QuestionState ::checkValidity () [inherited]

Définition à la ligne 20 du fichier [SH_QuestionState.cpp](#).

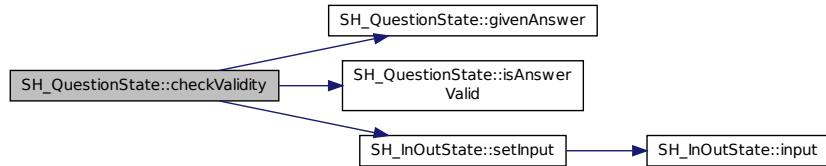
Références [SH_QuestionState ::answerInvalid\(\)](#), [SH_QuestionState ::answerValid\(\)](#), [SH_QuestionState ::givenAnswer\(\)](#), [SH_QuestionState ::isAnswerValid\(\)](#), [SH_GenericState ::next\(\)](#), et [SH_InOutState ::setInput\(\)](#).

Référencé par [SH_QuestionState ::setGivenAnswer\(\)](#).

```

00021 {
00022     bool ok = this->isAnswerValid(this->givenAnswer());
00023     if(ok) {
00024         qDebug() << this->givenAnswer() << " answer valid !";
00025         qDebug() << this->givenAnswer();
00026         SH_InOutState::setInput(this->givenAnswer());
00027         emit answerValid();
00028         emit next();
00029     } else {
00030         qDebug() << this->givenAnswer() << " answer invalid :-(";
00031         qDebug() << this->givenAnswer();
00032         emit answerInvalid();
00033     }
00034     return ok;
00035 }
```

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :



4.31.3.4 QMap< int, QVariant > SH_DatabaseContentQuestionState ::choiceList ()

Définition à la ligne 67 du fichier [SH_DatabaseContentQuestionState.cpp](#).

Références [m_choices](#), et [m_choicesDisplayed](#).

```
00067 {
```

```

00068     if(m_choicesDisplayed) {
00069         return m_choices;
00070     }
00071     return QMap<int,QVariant>();
00072 }
```

4.31.3.5 void SH_InOutState ::display (bool canDisplay) [inherited]

Définition à la ligne 95 du fichier [SH_IOState.cpp](#).

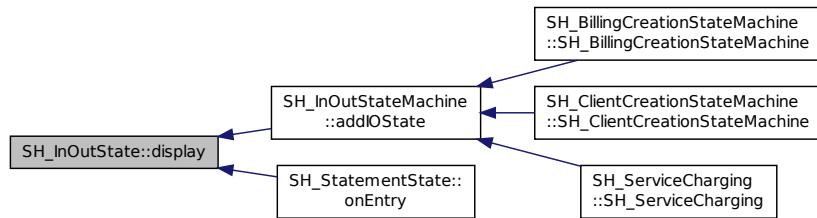
Références [SH_InOutState ::m_display](#), [SH_InOutState ::m_isVisible](#), [SH_InOutState ::m_output](#), et [SH_InOutState ::sendOutput\(\)](#).

Référencé par [SH_InOutStateMachine ::addIOState\(\)](#), et [SH_StatementState ::onEntry\(\)](#).

```

00096 {
00097     m_display=canDisplay;
00098     if(m_display && !m_output.isEmpty() && m_isVisible) {
00099         qDebug() << "resalut !" << QVariant(m_output);
00100         emit sendOutput(QVariant(m_output));
00101     }
00102 }
```

Voici le graphe des appelants de cette fonction :



4.31.3.6 void SH_DatabaseContentQuestionState ::displayChoiceList () [signal]

Référencé par [setOutput\(\)](#).

Voici le graphe des appelants de cette fonction :



4.31.3.7 QVariant SH_QuestionState ::givenAnswer () const [virtual], [inherited]

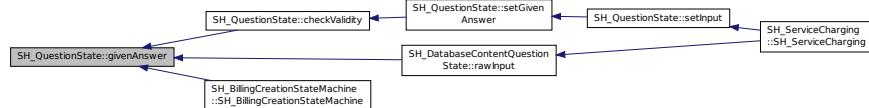
Définition à la ligne 55 du fichier [SH_QuestionState.cpp](#).

Références [SH_QuestionState ::m_givenAnswer](#).

Référencé par [SH_QuestionState ::checkValidity\(\)](#), [rawInput\(\)](#), et [SH_BillingCreationStateMachine ::SH_BillingCreationStateMachine\(\)](#).

```
00056 {
00057     return this->m_givenAnswer;
00058 }
```

Voici le graphe des appelants de cette fonction :



4.31.3.8 QVariant SH_InOutState ::input() const [virtual], [inherited]

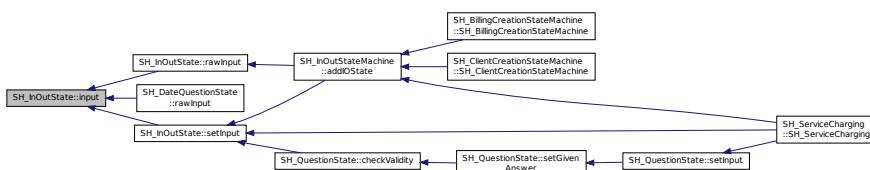
Définition à la ligne 20 du fichier [SH_IOState.cpp](#).

Références [SH_InOutState ::m_input](#).

Référencé par [SH_InOutState ::rawInput\(\)](#), [SH_DateQuestionState ::rawInput\(\)](#), et [SH_InOutState ::setInput\(\)](#).

```
00021 {
00022     return m_input;
00023 }
```

Voici le graphe des appelants de cette fonction :



4.31.3.9 bool SH_DatabaseContentQuestionState ::isValid(const QVariant & givenAnswer) [virtual]

Implémente [SH_QuestionState](#).

Définition à la ligne 33 du fichier [SH_DatabaseContentQuestionState.cpp](#).

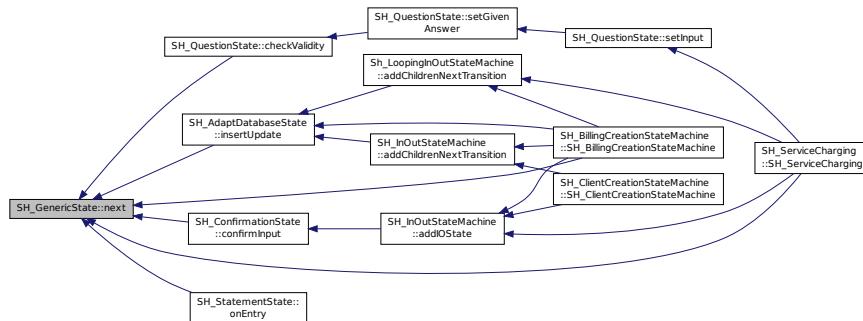
Références [m_choices](#).

```
00034 {
00035     qDebug() << m_choices.values();
00036     return m_choices.isEmpty() || m_choices.values().contains(
00037         givenAnswer);
```

4.31.3.10 void SH_GenericState ::next() [signal], [inherited]

Référencé par [SH_QuestionState ::checkValidity\(\)](#), [SH_ConfirmationState ::confirmInput\(\)](#), [SH_AdaptDatabaseState ::insertUpdate\(\)](#), [SH_StatementState ::onEntry\(\)](#), [SH_BillingCreationStateMachine ::SH_BillingCreationStateMachine\(\)](#), et [SH_ServiceCharging ::SH_ServiceCharging\(\)](#).

Voici le graphe des appelants de cette fonction :



4.31.3.11 void SH_GenericState ::onEntry (QEvent * event) [protected], [inherited]

Définition à la ligne 62 du fichier [SH_GenericDebugableState.cpp](#).

Références [SH_NamedObject ::name\(\)](#).

Référencé par [SH_StatementState ::onEntry\(\)](#).

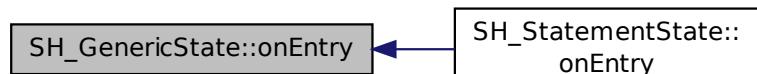
```

00063 {
00064     Q_UNUSED(event);
00065     qDebug() << "Machine: " << machine()->objectName() << " entered " << name();
00066 }
```

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :



4.31.3.12 void SH_InOutState ::onExit (QEvent * event) [inherited]

Définition à la ligne 110 du fichier [SH_IOState.cpp](#).

Références [SH_InOutState ::m_input](#), [SH_InOutState ::m_isVisible](#), [SH_GenericState ::onExit\(\)](#), et [SH_InOutState ::resendInput\(\)](#).

```

00111 {
00112     SH_GenericState::onExit(event);
00113     if(m_isVisible) {
00114         emit resendInput(m_input);
00115     }
00116 }
```

Voici le graphe d'appel pour cette fonction :



4.31.3.13 QString SH_InOutState::output() const [virtual], [inherited]

Définition à la ligne 56 du fichier [SH_IState.cpp](#).

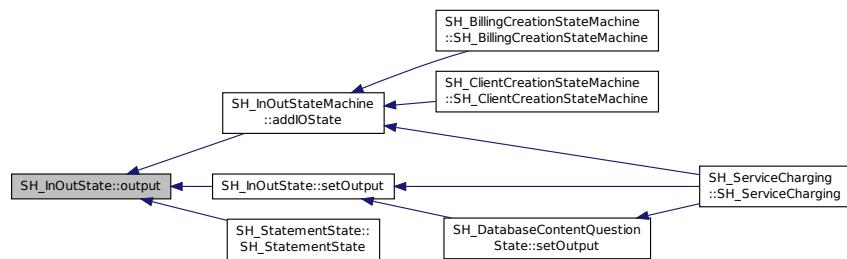
Références [SH_InOutState::m_output](#).

Référencé par [SH_InOutStateMachine::addIOState\(\)](#), [SH_InOutState::setOutput\(\)](#), et [SH_StatementState::SH_StatementState\(\)](#).

```

00057 {
00058     return m_output;
00059 }
```

Voici le graphe des appels de cette fonction :



4.31.3.14 QVariant SH_DatabaseContentQuestionState::rawInput() const [virtual]

Réimplémentée à partir de [SH_InOutState](#).

Définition à la ligne 57 du fichier [SH_DatabaseContentQuestionState.cpp](#).

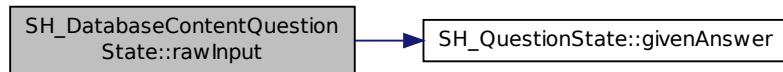
Références [SH_QuestionState::givenAnswer\(\)](#), et [m_choices](#).

Référencé par [SH_ServiceCharging::SH_ServiceCharging\(\)](#).

```

00058 {
00059     return m_choices.key(this->givenAnswer());
00060 }
```

Voici le graphe d'appel pour cette fonction :



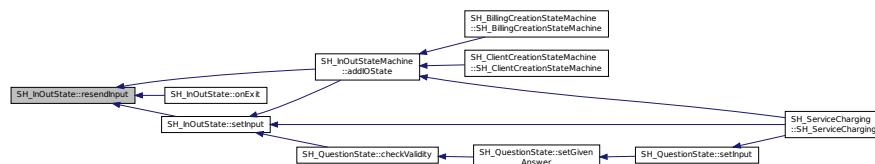
Voici le graphe des appelants de cette fonction :



4.31.3.15 void SH_InOutState ::resendInput (QVariant *input*) [signal], [inherited]

Référencé par [SH_InOutStateMachine ::addIOState\(\)](#), [SH_InOutState ::onExit\(\)](#), et [SH_InOutState ::setInput\(\)](#).

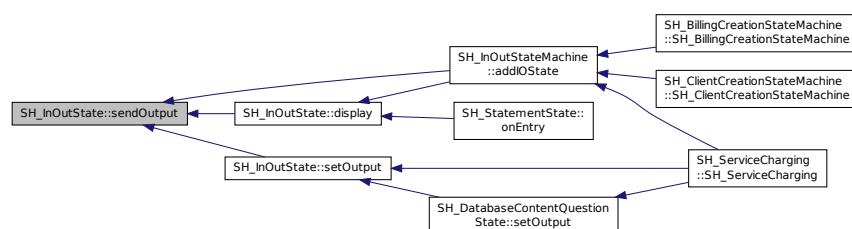
Voici le graphe des appelants de cette fonction :



4.31.3.16 void SH_InOutState ::sendOutput (QVariant *output*) [signal], [inherited]

Référencé par [SH_InOutStateMachine ::addIOState\(\)](#), [SH_InOutState ::display\(\)](#), et [SH_InOutState ::setOutput\(\)](#).

Voici le graphe des appelants de cette fonction :



4.31.3.17 void SH_QuestionState : :setGivenAnswer (const QVariant & givenAnswer) [virtual], [inherited]

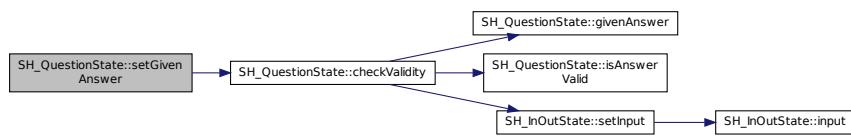
Définition à la ligne 66 du fichier [SH_QuestionState.cpp](#).

Références [SH_QuestionState : :checkValidity\(\)](#), et [SH_QuestionState : :m_givenAnswer](#).

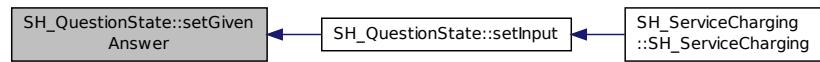
Référencé par [SH_QuestionState : :setInput\(\)](#).

```
00067 {
00068     this->m_givenAnswer = givenAnswer;
00069     this->checkValidity();
00070 }
```

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appels de cette fonction :



4.31.3.18 void SH_QuestionState : :setInput (const QVariant & input) [virtual], [inherited]

Réimplémentée à partir de [SH_InOutState](#).

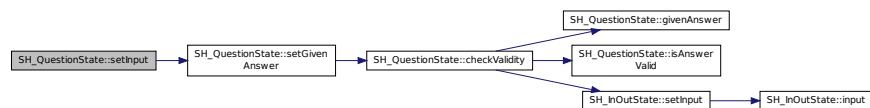
Définition à la ligne 43 du fichier [SH_QuestionState.cpp](#).

Références [SH_QuestionState : :setGivenAnswer\(\)](#).

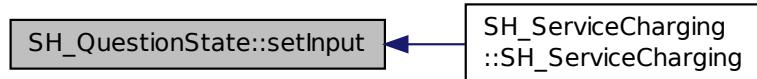
Référencé par [SH_ServiceCharging : :SH_ServiceCharging\(\)](#).

```
00044 {
00045     qDebug() << "new answer " << input.toString();
00046     this->setGivenAnswer(input);
00047 }
```

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :



4.31.3.19 void SH_DatabaseContentQuestionState ::setOutput(const QString & output) [virtual]

Réimplémentée à partir de [SH_InOutState](#).

Définition à la ligne 43 du fichier [SH_DatabaseContentQuestionState.cpp](#).

Références [displayChoiceList\(\)](#), [m_choices](#), [m_choicesDisplayed](#), et [SH_InOutState ::setOutput\(\)](#).

Référencé par [SH_ServiceCharging ::SH_ServiceCharging\(\)](#).

```

00044 {
00045     SH_QuestionState::setOutput(output);
00046     if(m_choices.size() < 8) {
00047         m_choicesDisplayed = true;
00048         emit displayChoiceList();
00049     }
00050 }

```

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :



4.31.3.20 void SH_InOutState ::setVisibility(bool isVisible) [virtual], [slot], [inherited]

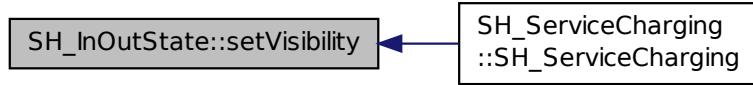
Définition à la ligne 81 du fichier [SH_IOState.cpp](#).

Références [SH_InOutState ::m_isVisible](#).

Référencé par [SH_ServiceCharging ::SH_ServiceCharging\(\)](#).

```
00082 {
00083     m_isVisible = isVisible;
00084 }
```

Voici le graphe des appels de cette fonction :



4.31.3.21 QString SH_GenericState : toString() [virtual], [inherited]

Réimplémentée à partir de [SH_NamedObject](#).

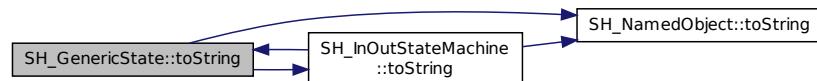
Définition à la ligne 21 du fichier [SH_GenericDebugableState.cpp](#).

Références [SH_NamedObject : toString\(\)](#), et [SH_InOutStateMachine : toString\(\)](#).

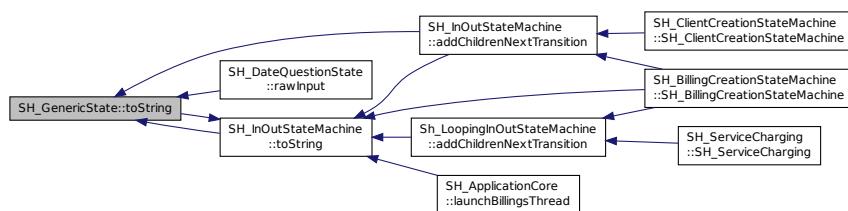
Référencé par [SH_InOutStateMachine : addChildrenNextTransition\(\)](#), [SH_DateQuestionState : rawInput\(\)](#), et [SH_InOutStateMachine : toString\(\)](#).

```
00022 {
00023     QStateMachine* machine = this->machine();
00024     SH_InOutStateMachine* mach = qobject_cast<
00025         SH_InOutStateMachine * >(machine);
00026     if(mach) {
00027         return SH_NamedObject::toString() + " [in "+mach->
00028             toString()+"] ";
00029     } else {
00030         return SH_NamedObject::toString();
00031     }
00030 }
```

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appels de cette fonction :



4.31.3.22 bool SH_InOutState ::visibility() [inherited]

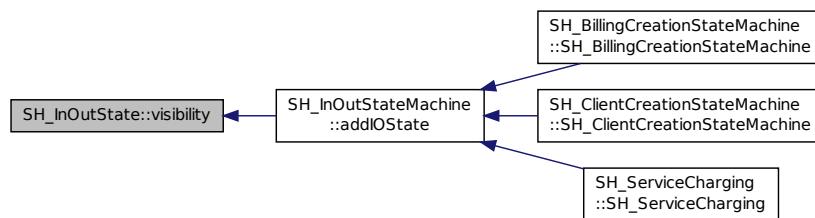
Définition à la ligne 91 du fichier [SH_IOState.cpp](#).

Références [SH_InOutState ::m_isVisible](#).

Référencé par [SH_InOutStateMachine ::addIOState\(\)](#).

```
00091     return m_isVisible;
00092 }
00093 }
```

Voici le graphe des appels de cette fonction :



4.31.4 Documentation des données membres

4.31.4.1 QMap<int, QVariant> SH_DatabaseContentQuestionState ::m_choices [private]

m_choices

Définition à la ligne 80 du fichier [SH_DatabaseContentQuestionState.h](#).

Référencé par [choiceList\(\)](#), [isValid\(\)](#), [rawInput\(\)](#), [setOutput\(\)](#), et [SH_DatabaseContentQuestionState\(\)](#).

4.31.4.2 bool SH_DatabaseContentQuestionState ::m_choicesDisplayed [private]

m_choicesDisplayed

Définition à la ligne 84 du fichier [SH_DatabaseContentQuestionState.h](#).

Référencé par [choiceList\(\)](#), et [setOutput\(\)](#).

4.31.4.3 QString SH_DatabaseContentQuestionState ::m_condition [private]

m_condition

Définition à la ligne 72 du fichier [SH_DatabaseContentQuestionState.h](#).

Référencé par [SH_DatabaseContentQuestionState\(\)](#).

4.31.4.4 QString SH_DatabaseContentQuestionState ::m_field [private]

m_field

Définition à la ligne 76 du fichier [SH_DatabaseContentQuestionState.h](#).

Référencé par [SH_DatabaseContentQuestionState\(\)](#).

4.31.4.5 QString SH_DatabaseContentQuestionState ::m_table [private]

m_table

Définition à la ligne 68 du fichier [SH_DatabaseContentQuestionState.h](#).

Référencé par [SH_DatabaseContentQuestionState\(\)](#).

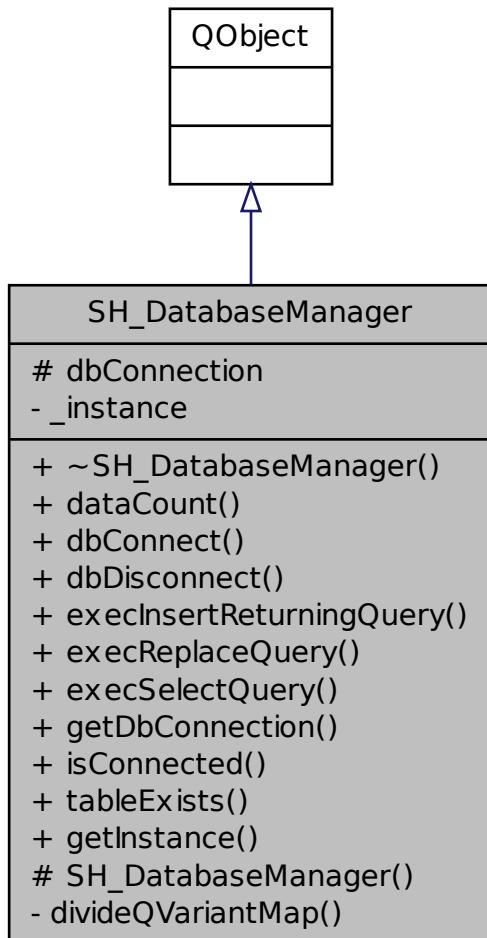
La documentation de cette classe a été générée à partir des fichiers suivants :

- logic/[SH_DatabaseContentQuestionState.h](#)
- logic/[SH_DatabaseContentQuestionState.cpp](#)

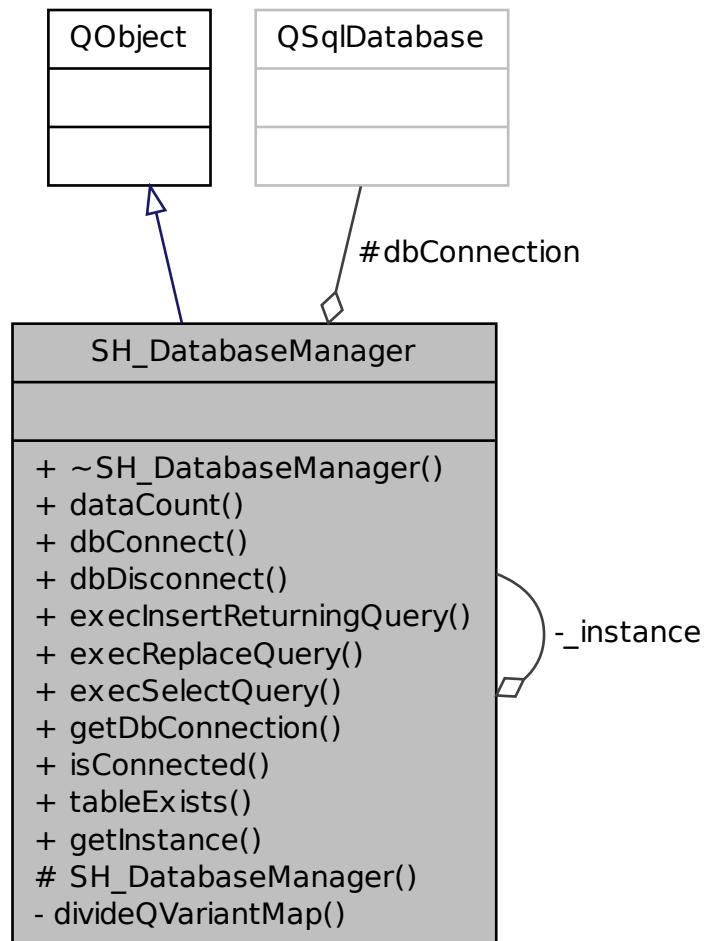
4.32 Référence de la classe SH_DatabaseManager

```
#include <SH_DatabaseManager.h>
```

Graphe d'héritage de SH_DatabaseManager :



Graphe de collaboration de SH_DatabaseManager :



Fonctions membres publiques

- `~SH_DatabaseManager ()`
- `int dataCount (QString tableName, QString filter)`
- `bool dbConnect ()`
- `bool dbDisconnect ()`
- `QVariant execInsertReturningQuery (QString tableName, QVariantMap values, QString returningField)`
- `bool execReplaceQuery (QString tableName, QVariantMap values)`
- `QSqlQuery execSelectQuery (QString tableName, QStringList fields=QStringList("*"), QString condition="", QString ordering="")`
- `QSqlDatabase getDbConnection ()`
- `bool isConnected ()`
- `bool tableExists (QString tableName)`

Fonctions membres publiques statiques

- `static SH_DatabaseManager * getInstance ()`

Fonctions membres protégées

- [SH_DatabaseManager \(\)](#)

Attributs protégés

- [QSqlDatabase dbConnection](#)
dbConnection

Fonctions membres privées

- [void divideQVariantMap \(QVariantMap values, QString &fields, QString &vals\)](#)

Attributs privés statiques

- [static SH_DatabaseManager * _instance = 0](#)

4.32.1 Description détaillée

Définition à la ligne [62](#) du fichier [SH_DatabaseManager.h](#).

4.32.2 Documentation des constructeurs et destructeur**4.32.2.1 SH_DatabaseManager : :SH_DatabaseManager () [protected]**

Définition à la ligne [44](#) du fichier [SH_DatabaseManager.cpp](#).

Références [dbConnect\(\)](#), [dbConnection](#), [dbDriverNotExistStr](#), [dbDriverStr](#), [dbFileNameStr](#), [dbFilePathStr](#), [dbPasswordStr](#), [dbUsernameStr](#), et [SH_MessageManager : :errorMessage\(\)](#).

Référencé par [getInstance\(\)](#).

```

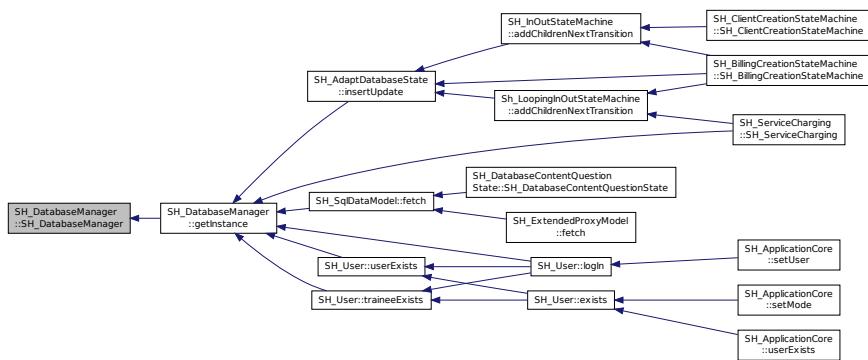
00045 {
00046     /*
00047      *Check the existence of the database driver.
00048     */
00049     if (! QSqlDatabase::isDriverAvailable(dbDriverStr))
00050     {
00051         /*
00052          *Gui message that informs that the driver does not exist
00053         */
00054         SH_MessageManager::errorMessage(
00055             dbDriverNotExistStr);
00056         qDebug() << dbConnection.lastError();
00057         for (int i = 0; i < dbConnection.drivers().count(); i++)
00058         {
00059             qDebug() << "AVAILABLE DRIVERS : " << dbConnection.drivers()[i] << endl;
00060         }
00061         exit(1);
00062     }
00063     /*
00064      *Connect to the database with the following driver.
00065     */
00066     dbConnection = QSqlDatabase::addDatabase(dbDriverStr);
00067     if (dbDriverStr == "QIBASE")
00068     {
00069         dbConnection.setDatabaseName(dbFilePathStr);
00070     } else {
00071         dbConnection.setDatabaseName(dbFileNameStr);
00072     }
00073     dbConnection.setUserName(dbUsernameStr);
00074     dbConnection.setPassword(dbPasswordStr);
00075     dbConnect();
00076 }
```

```
00077
00078 }
```

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appels de cette fonction :



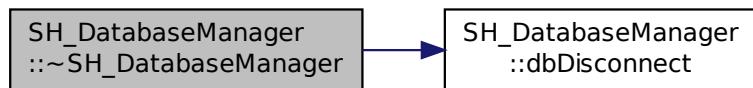
4.32.2.2 SH_DatabaseManager ::~SH_DatabaseManager ()

Définition à la ligne 33 du fichier [SH_DatabaseManager.cpp](#).

Références [dbDisconnect\(\)](#).

```
00034 {
00035     dbDisconnect();
00036 }
```

Voici le graphe d'appel pour cette fonction :



4.32.3 Documentation des fonctions membres

4.32.3.1 int SH_DatabaseManager ::dataCount (QString tableName, QString filter)

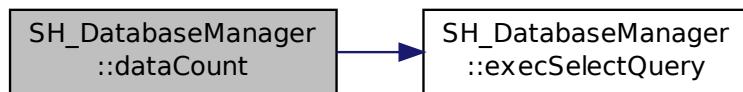
Définition à la ligne 170 du fichier [SH_DatabaseManager.cpp](#).

Références [dbConnection](#), et [execSelectQuery\(\)](#).

```

00170
00171     if (!tableName.isEmpty() && !filter.isEmpty()) {
00172         QSqlQuery result = execSelectQuery(tableName, QStringList("COUNT(*) AS MATCH"),
00173                                             filter);
00174         if (dbConnection.driver() ->hasFeature(QSqlDriver::QuerySize)) {
00175             return result.size();
00176         } else {
00177             if (result.next()) {
00178                 QSqlRecord rec = result.record();
00179                 if (!rec.isEmpty() && result.isValid()) {
00180                     return rec.value(rec.indexOf("MATCH")).toInt();
00181                 }
00182             }
00183         }
00184     }
00185 }
```

Voici le graphe d'appel pour cette fonction :



4.32.3.2 bool SH_DatabaseManager ::dbConnect ()

Définition à la ligne 89 du fichier [SH_DatabaseManager.cpp](#).

Références [dbCannotOpenStr](#), [dbConnection](#), et [SH_MessageManager ::errorMessage\(\)](#).

Référencé par [SH_DatabaseManager\(\)](#).

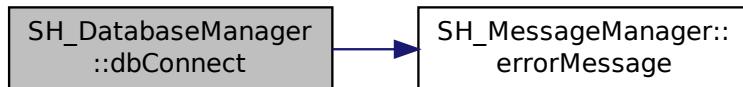
```

00090 {
00091     /*
00092      *Open database, if the database cannot open for
00093      *any reason print a warning.
00094      */
00095     if (!dbConnection.open())
00096     {
00097         /*
00098          *Gui message that informs that the database cannot open
00099          */
00100         SH_MessageManager::errorMessage(
00101             dbCannotOpenStr);
00102         qDebug() << dbConnection.lastError();
00103         /*
00104          *@return false if database connection failed.
00105          */
00106         return false;
00107     }
00108
00109     /*
00110      *@return true if database connection successed
00111      */

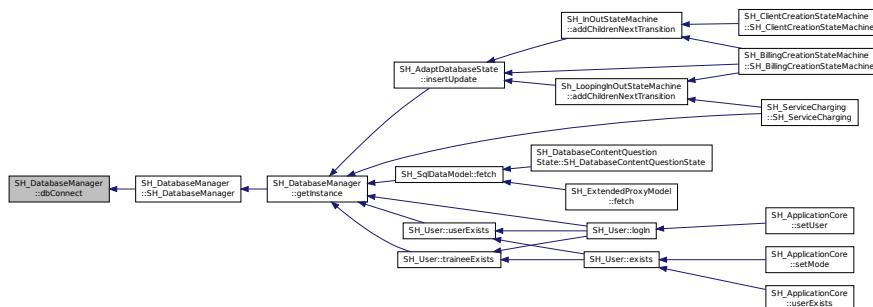
```

```
00112     return dbConnection.isOpen();
00113 }
```

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :



4.32.3.3 bool SH_DatabaseManager ::dbDisconnect ()

Définition à la ligne 124 du fichier [SH_DatabaseManager.cpp](#).

Références [dbConnection](#).

Référencé par [~SH_DatabaseManager\(\)](#).

```
00125 {
00126     /*
00127      *close database
00128      */
00129     dbConnection.close();
00130     return (!dbConnection.isOpen());
00131 }
```

Voici le graphe des appelants de cette fonction :



4.32.3.4 void SH_DatabaseManager ::divideQVariantMap (QVariantMap values, QString & fields, QString & vals)
[private]

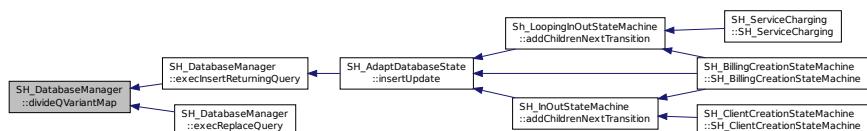
Définition à la ligne 260 du fichier [SH_DatabaseManager.cpp](#).

Référencé par [execInsertReturningQuery\(\)](#), et [execReplaceQuery\(\)](#).

```

00260
00261     for(auto field : values.keys())
00262     {
00263         fields += field+",";
00264         QVariant val = values.value(field);
00265         bool ok;
00266         int intValue = val.toInt(&ok);
00267         if(ok) {
00268             vals += QString::number(intValue)+",";
00269         }
00270         double dblVal = val.toDouble(&ok);
00271         if(ok) {
00272             vals += QString::number(dblVal)+",";
00273         }
00274         /*bool boolVal = val.toBool();
00275
00276             if(boolVal) {
00277                 &vals += "'"+1+"'','";
00278
00279             }*/
00280         QDate dateVal = val.toDate();
00281         if(dateVal.isValid()) {
00282             vals += "'"+dateVal.toString()+"',"; /*FIXME adapt date format*/
00283         }
00284         QDateTime dateTimeVal = val.toDateTime();
00285         if(dateTimeVal.isValid()) {
00286             vals += "'"+dateTimeVal.toString()+"',"; /*FIXME adapt datetime format*/
00287         }
00288         QString stringVal = val.toString();
00289         vals += "'"+stringVal+"',";
00290     }
00291     fields = fields.left(fields.lastIndexOf(',')-1);
00292     vals = vals.left(vals.lastIndexOf(',')-1);
00293 }
```

Voici le graphe des appelants de cette fonction :



4.32.3.5 QVariant SH_DatabaseManager ::execInsertReturningQuery (QString tableName, QVariantMap values, QString returningField)

Définition à la ligne 237 du fichier [SH_DatabaseManager.cpp](#).

Références [dbConnection](#), et [divideQVariantMap\(\)](#).

Référencé par [SH_AdaptDatabaseState ::insertUpdate\(\)](#).

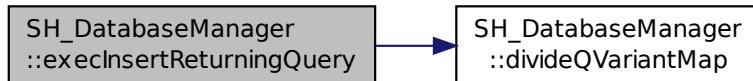
```

00237
00238     {
00239         QString fields;
00240         QString vals;
00241         divideQVariantMap(values, fields, vals);
00242         QString query;
00243         if(dbConnection.driverName() == "QIBASE") {
00244             query = QString("UPDATE OR INSERT INTO %1(%2) VALUES (%3) MATCHING(ID) RETURNING %4").arg(tableName)
00245             .arg(fields).arg(vals).arg(returningField);
00246         }
00247         QSqlQuery result = dbConnection.exec(query);
00248 }
```

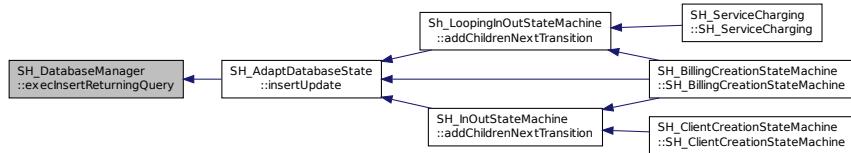
```

00246     qDebug() << result.executedQuery() << " > " << result.isValid() << " << result.isActive();
00247     if(result.next()) {
00248         QSqlRecord rec = result.record();
00249         if(!rec.isEmpty() && result.isValid()) {
00250             return rec.value(rec.indexOf(returningField));
00251         }
00252     }
00253     return QVariant();
00254 }
```

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appels de cette fonction :



4.32.3.6 bool SH_DatabaseManager ::execReplaceQuery (QString tableName, QVariantMap values)

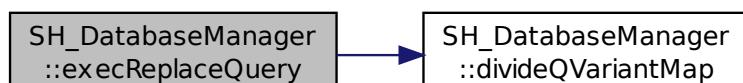
Définition à la ligne 220 du fichier [SH_DatabaseManager.cpp](#).

Références [dbConnection](#), et [divideQVariantMap\(\)](#).

```

00220
00221     QString fields;
00222     QString vals;
00223     divideQVariantMap(values, fields, vals);
00224     QString query;
00225     if(dbConnection.driverName() == "QIBASE") {
00226         query = QString("UPDATE OR INSERT INTO %1(%2) VALUES(%3) MATCHING(ID)").arg(tableName).arg(fields).
00227             arg(vals);
00228     }
00229     QSqlQuery result = dbConnection.exec(query);
00230     qDebug() << result.executedQuery() << " > " << result.isValid() << " << result.isActive();
00231 }
```

Voici le graphe d'appel pour cette fonction :



4.32.3.7 QSqlQuery SH_DatabaseManager : execSelectQuery (QString tableName, QStringList fields = QStringList ("*"), QString condition = " ", QString ordering = " ")

Définition à la ligne 193 du fichier [SH_DatabaseManager.cpp](#).

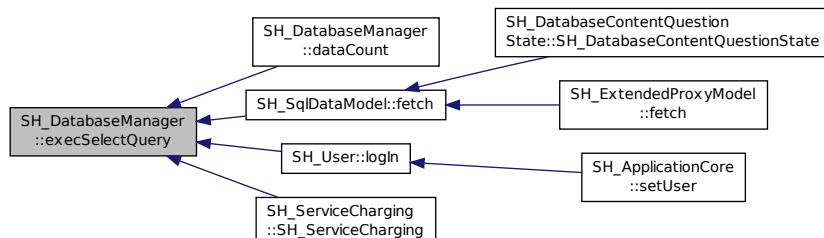
Références [dbConnection](#).

Référencé par [dataCount\(\)](#), [SH_SqlDataModel : fetch\(\)](#), [SH_User : login\(\)](#), et [SH_ServiceCharging : SH_ServiceCharging\(\)](#).

```

00193
00194     {
00195         if(fields.isEmpty()) {
00196             fields.append("*");
00197         }
00198         QString query;
00199         if(dbConnection.driverName() == "QIBASE") {
00200             query = QString("SELECT %1 FROM %2").arg(fields.join(", ")).arg(tableName);
00201             if(!condition.isEmpty()) {
00202                 query = QString("%1 WHERE %2").arg(query).arg(condition);
00203             }
00204             if(!ordering.isEmpty()) {
00205                 query = QString("%1 ORDER BY %2").arg(query).arg(ordering);
00206             }
00207         }
00208         qDebug() << query;
00209         QSqlQuery result;
00210         result.exec(query);
00211         qDebug() << result.executedQuery() << " > " << result.isValid() << " << result.isActive();
00212         return result;
00213 }
```

Voici le graphe des appels de cette fonction :



4.32.3.8 QSqlDatabase SH_DatabaseManager : getDbConnection ()

Définition à la ligne 150 du fichier [SH_DatabaseManager.cpp](#).

Références [dbConnection](#).

```

00151 {
00152     return dbConnection;
00153 }
```

4.32.3.9 SH_DatabaseManager * SH_DatabaseManager : getInstance () [static]

Définition à la ligne 17 du fichier [SH_DatabaseManager.cpp](#).

Références [_instance](#), [dbFilePathStr](#), et [SH_DatabaseManager\(\)](#).

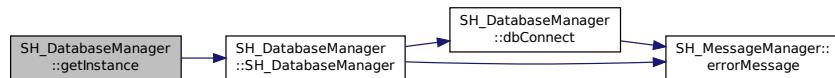
Référencé par [SH_SqlDataModel : fetch\(\)](#), [SH_AdaptDatabaseState : insertUpdate\(\)](#), [SH_User : login\(\)](#), [SH_ServiceCharging : SH_ServiceCharging\(\)](#), [SH_User : traineeExists\(\)](#), et [SH_User : userExists\(\)](#).

```

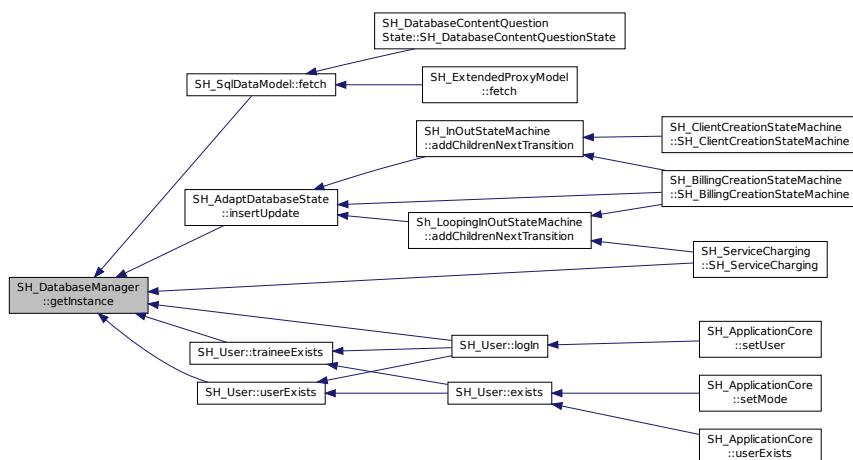
00018 {
00019     if (_instance == 0)
00020     {
00021         _instance = new SH_DatabaseManager;
00022     }
00023     qDebug() << dbFilePathStr;
00024     return _instance;
00025 }

```

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appels de cette fonction :



4.32.3.10 bool SH_DatabaseManager ::isConnected()

Définition à la ligne 139 du fichier [SH_DatabaseManager.cpp](#).

Références [dbConnection](#).

```

00140 {
00141     return dbConnection.isOpen();
00142 }

```

4.32.3.11 bool SH_DatabaseManager ::tableExists (QString tableName)

Définition à la ligne 160 du fichier [SH_DatabaseManager.cpp](#).

Références [dbConnection](#).

```

00161 {
00162     return dbConnection.tables(SqlView).contains(tableName.toUpper(), Qt::CaseInsensitive)
        || dbConnection.tables(SqlTable).contains(tableName.toUpper(), Qt::CaseInsensitive);
00163 }

```

4.32.4 Documentation des données membres

4.32.4.1 **SH_DatabaseManager * SH_DatabaseManager ::_instance = 0** [static], [private]

[SH_DatabaseManager ::_instance](#)

Définition à la ligne 66 du fichier [SH_DatabaseManager.h](#).

Référencé par [getInstance\(\)](#).

4.32.4.2 **QSqlDatabase SH_DatabaseManager ::dbConnection** [protected]

[dbConnection](#)

Définition à la ligne 87 du fichier [SH_DatabaseManager.h](#).

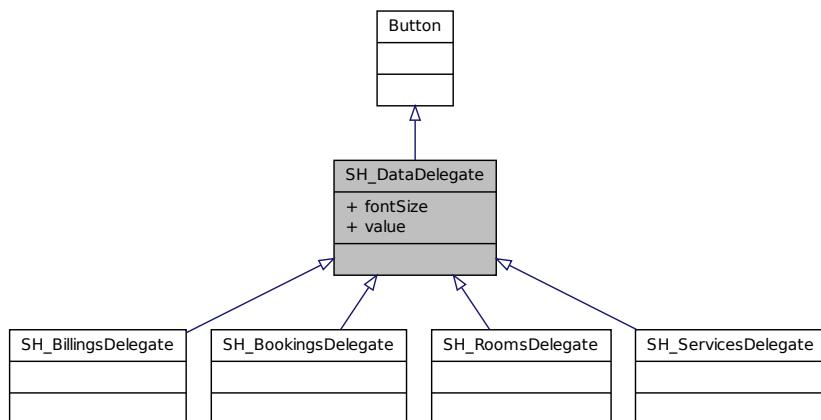
Référencé par [dataCount\(\)](#), [dbConnect\(\)](#), [dbDisconnect\(\)](#), [execInsertReturningQuery\(\)](#), [execReplaceQuery\(\)](#), [execSelectQuery\(\)](#), [getDbConnection\(\)](#), [isConnected\(\)](#), [SH_DatabaseManager\(\)](#), et [tableExists\(\)](#).

La documentation de cette classe a été générée à partir des fichiers suivants :

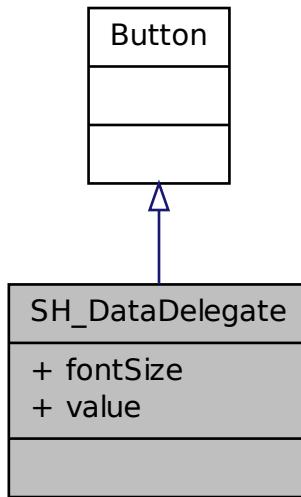
- [SH_DatabaseManager.h](#)
- [SH_DatabaseManager.cpp](#)

4.33 Référence de la classe SH_DataDelegate

Graphe d'héritage de SH_DataDelegate :



Graphe de collaboration de SH_DataDelegate :



Propriétés

- int `fontSize`
- string `value`

4.33.1 Description détaillée

Définition à la ligne 4 du fichier [SH_DataDelegate.qml](#).

4.33.2 Documentation des propriétés

4.33.2.1 int SH_DataDelegate ::fontSize

Définition à la ligne 9 du fichier [SH_DataDelegate.qml](#).

4.33.2.2 string SH_DataDelegate ::value

Définition à la ligne 7 du fichier [SH_DataDelegate.qml](#).

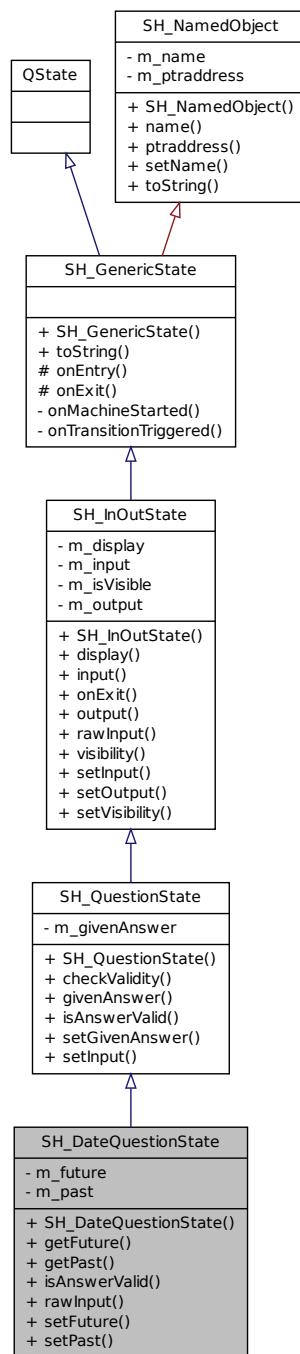
La documentation de cette classe a été générée à partir du fichier suivant :

- [views/qml/SH_DataDelegate.qml](#)

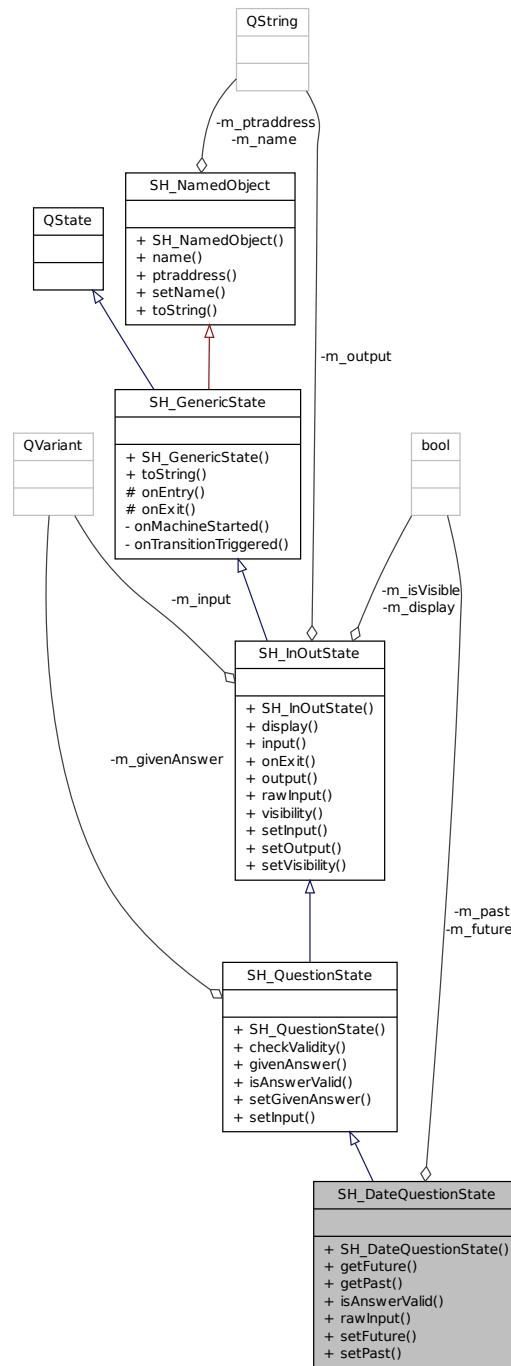
4.34 Référence de la classe SH_DateQuestionState

```
#include <SH_DateQuestionState.h>
```

Graphe d'héritage de SH_DateQuestionState :



Graphe de collaboration de SH_DateQuestionState :



Connecteurs publics

- `virtual void setOutput (const QString &output)`
- `virtual void setVisibility (bool isVisible)`

Signaux

- `void answerInvalid ()`

```

    answerInvalid
- void answerValid ()
    answerValid
- void next ()
- void resendInput (QVariant input)
- void sendOutput (QVariant output)

```

Fonctions membres publiques

```

- SH_DateQuestionState (QString question, QString name, bool past=true, bool future=false, QState *parent=0)
- bool checkValidity ()
- void display (bool canDisplay)
- bool getFuture () const
- bool getPast () const
- virtual QVariant givenAnswer () const
- virtual QVariant input () const
- virtual bool isAnswerValid (const QVariant &givenAnswer)
- void onExit (QEvent *event)
- virtual QString output () const
- QVariant rawInput () const
- void setFuture (bool value)
- virtual void setGivenAnswer (const QVariant &givenAnswer)
- virtual void setInput (const QVariant &input)
- void setPast (bool value)
- QString toString ()
- bool visibility ()

```

Fonctions membres protégées

```
- void onEntry (QEvent *event)
```

Attributs privés

```

- bool m_future
    m_future
- bool m_past
    m_past

```

4.34.1 Description détaillée

Définition à la ligne 10 du fichier [SH_DateQuestionState.h](#).

4.34.2 Documentation des constructeurs et destructeur

4.34.2.1 SH_DateQuestionState : :SH_DateQuestionState (QString *question*, QString *name*, bool *past* = true, bool *future* = false, QState * *parent* = 0)

Paramètres

<i>question</i>	
<i>name</i>	
<i>past</i>	
<i>future</i>	
<i>parent</i>	

Définition à la ligne 9 du fichier [SH_DateQuestionState.cpp](#).

```

00009
:
00010      SH_QuestionState(question+" (au format jj-mm-aaaa)", name, parent),
    m_past(past), m_future(future)
00011 {
00012 }

```

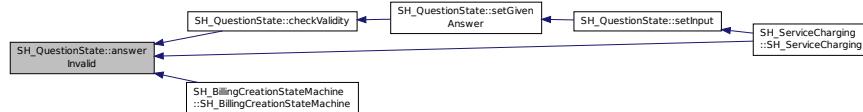
4.34.3 Documentation des fonctions membres

4.34.3.1 void SH_QuestionState ::answerInvalid() [signal], [inherited]

answerInvalid

Référencé par [SH_QuestionState ::checkValidity\(\)](#), [SH_BillingCreationStateMachine ::SH_BillingCreationStateMachine\(\)](#), et [SH_ServiceCharging ::SH_ServiceCharging\(\)](#).

Voici le graphe des appelants de cette fonction :

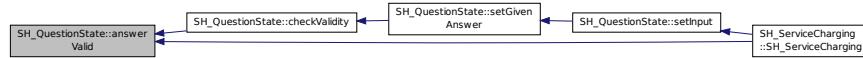


4.34.3.2 void SH_QuestionState ::answerValid() [signal], [inherited]

answerValid

Référencé par [SH_QuestionState ::checkValidity\(\)](#), et [SH_ServiceCharging ::SH_ServiceCharging\(\)](#).

Voici le graphe des appelants de cette fonction :



4.34.3.3 bool SH_QuestionState ::checkValidity() [inherited]

Définition à la ligne 20 du fichier [SH_QuestionState.cpp](#).

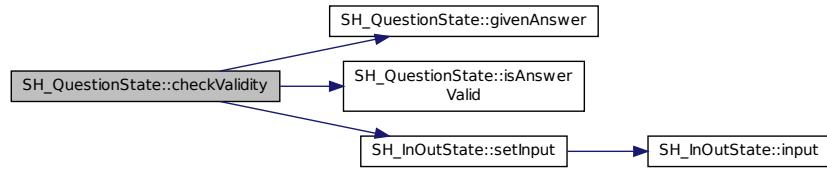
Références [SH_QuestionState ::answerInvalid\(\)](#), [SH_QuestionState ::answerValid\(\)](#), [SH_QuestionState ::givenAnswer\(\)](#), [SH_QuestionState ::isAnswerValid\(\)](#), [SH_GenericState ::next\(\)](#), et [SH_InOutState ::setInput\(\)](#).

Référencé par [SH_QuestionState ::setGivenAnswer\(\)](#).

```

00021 {
00022     bool ok = this->isAnswerValid(this->givenAnswer());
00023     if(ok) {
00024         qDebug() << this->givenAnswer() << " answer valid !";
00025         qDebug() << this->givenAnswer();
00026         SH_InOutState::setInput(this->givenAnswer());
00027         emit answerValid();
00028         emit next();
00029     } else {
00030         qDebug() << this->givenAnswer() << " answer invalid :-(";
00031         qDebug() << this->givenAnswer();
00032         emit answerInvalid();
00033     }
00034     return ok;
00035 }
  
```

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appels de cette fonction :



4.34.3.4 void SH_InOutState ::display (bool canDisplay) [inherited]

Définition à la ligne 95 du fichier [SH_IOSState.cpp](#).

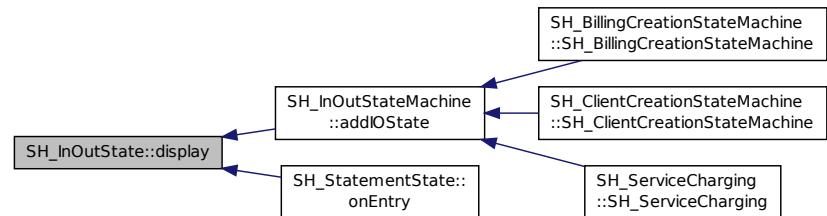
Références [SH_InOutState ::m_display](#), [SH_InOutState ::m_isVisible](#), [SH_InOutState ::m_output](#), et [SH_InOutState ::sendOutput\(\)](#).

Référencé par [SH_InOutStateMachine ::addIOSState\(\)](#), et [SH_StatementState ::onEntry\(\)](#).

```

00096 {
00097     m_display=canDisplay;
00098     if(m_display && !m_output.isEmpty() && m_isVisible) {
00099         qDebug() << "resalut !" << QVariant(m_output);
00100         emit sendOutput(QVariant(m_output));
00101     }
00102 }
```

Voici le graphe des appels de cette fonction :



4.34.3.5 bool SH_DateQuestionState ::getFuture () const

Définition à la ligne 59 du fichier [SH_DateQuestionState.cpp](#).

Références [m_future](#).

```
00060 {
00061     return m_future;
00062 }
```

4.34.3.6 bool SH_DateQuestionState ::getPast () const

Définition à la ligne 37 du fichier [SH_DateQuestionState.cpp](#).

Références [m_past](#).

```
00038 {
00039     return m_past;
00040 }
```

4.34.3.7 QVariant SH_QuestionState ::givenAnswer () const [virtual], [inherited]

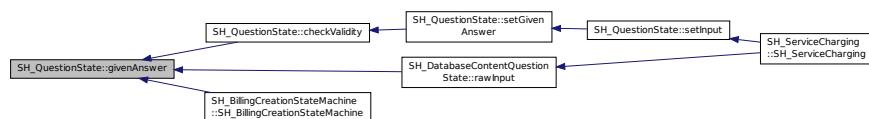
Définition à la ligne 55 du fichier [SH_QuestionState.cpp](#).

Références [SH_QuestionState ::m_givenAnswer](#).

Référencé par [SH_QuestionState ::checkValidity\(\)](#), [SH_DatabaseContentQuestionState ::rawInput\(\)](#), et [SH_BillingCreationStateMachine ::SH_BillingCreationStateMachine\(\)](#).

```
00056 {
00057     return this->m_givenAnswer;
00058 }
```

Voici le graphe des appelants de cette fonction :



4.34.3.8 QVariant SH_InOutState ::input () const [virtual], [inherited]

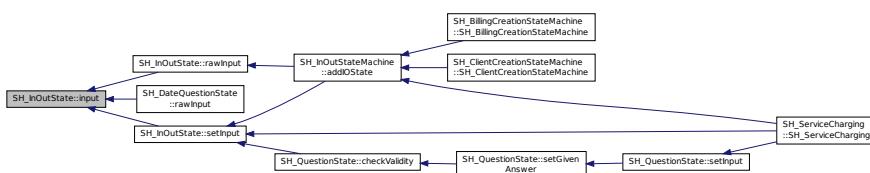
Définition à la ligne 20 du fichier [SH_IOState.cpp](#).

Références [SH_InOutState ::m_input](#).

Référencé par [SH_InOutState ::rawInput\(\)](#), [rawInput\(\)](#), et [SH_InOutState ::setInput\(\)](#).

```
00021 {
00022     return m_input;
00023 }
```

Voici le graphe des appelants de cette fonction :



4.34.3.9 bool SH_DateQuestionState ::isValid (const QVariant & givenAnswer) [virtual]

Implémente [SH_QuestionState](#).

Définition à la ligne 20 du fichier [SH_DateQuestionState.cpp](#).

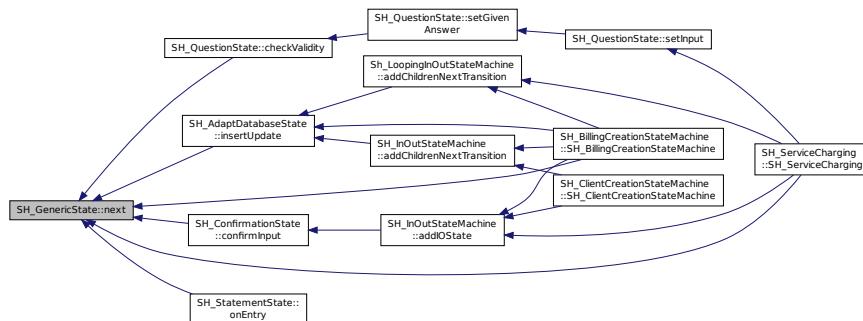
Références [m_future](#), et [m_past](#).

```
00021 {
00022     QDate answer = QDate::fromString(givenAnswer.toString(),QString("dd-MM-yyyy"));
00023     if(answer.isValid()) {
00024         qDebug() << "date conforme";
00025         return ((m_future && answer >= QDate::currentDate()) || (m_past && answer <=
QDate::currentDate()));
00026     } else {
00027         return false;
00028     }
00029 }
```

4.34.3.10 void SH_GenericState ::next() [signal], [inherited]

Référencé par [SH_QuestionState ::checkValidity\(\)](#), [SH_ConfirmationState ::confirmInput\(\)](#), [SH_AdaptDatabaseState ::insertUpdate\(\)](#), [SH_StatementState ::onEntry\(\)](#), [SH_BillingCreationStateMachine ::SH_BillingCreationStateMachine\(\)](#), et [SH_ServiceCharging ::SH_ServiceCharging\(\)](#).

Voici le graphe des appelleurs de cette fonction :



4.34.3.11 void SH_GenericState ::onEntry (QEvent * event) [protected], [inherited]

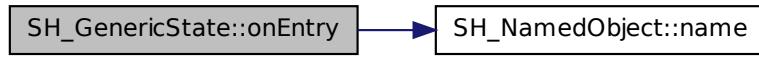
Définition à la ligne 62 du fichier [SH_GenericDebugableState.cpp](#).

Références [SH_NamedObject ::name\(\)](#).

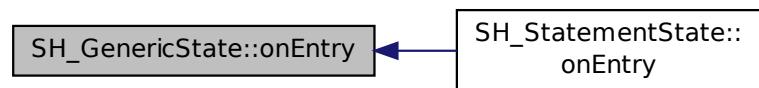
Référencé par [SH_StatementState ::onEntry\(\)](#).

```
00063 {
00064     Q_UNUSED(event);
00065     qDebug() << "Machine: " << machine()->objectName() << " entered " << name();
00066 }
```

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :



4.34.3.12 void SH_InOutState ::onExit(QEvent * event) [inherited]

Définition à la ligne 110 du fichier [SH_IOState.cpp](#).

Références [SH_InOutState ::m_input](#), [SH_InOutState ::m_isVisible](#), [SH_GenericState ::onExit\(\)](#), et [SH_InOutState ::resendInput\(\)](#).

```

00111 {
00112     SH_GenericState::onExit(event);
00113     if(m_isVisible) {
00114         emit resendInput(m_input);
00115     }
00116 }
```

Voici le graphe d'appel pour cette fonction :



4.34.3.13 QString SH_InOutState ::output() const [virtual], [inherited]

Définition à la ligne 56 du fichier [SH_IOState.cpp](#).

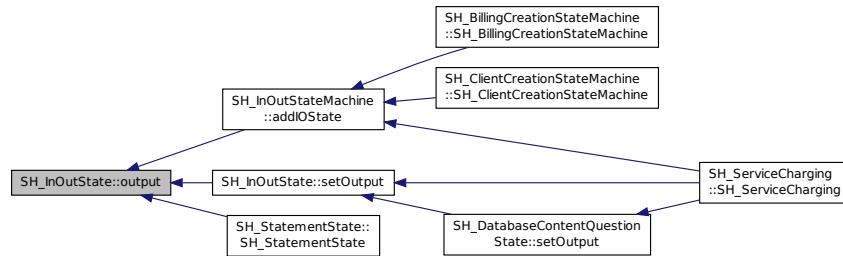
Références [SH_InOutState ::m_output](#).

Référencé par [SH_InOutStateMachine ::addIOState\(\)](#), [SH_InOutState ::setOutput\(\)](#), et [SH_StatementState ::SH_StatementState\(\)](#).

```

00057 {
00058     return m_output;
00059 }
```

Voici le graphe des appelants de cette fonction :



4.34.3.14 QVariant SH_DateQuestionState ::rawInput() const [virtual]

Réimplémentée à partir de [SH_InOutState](#).

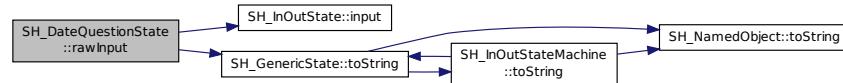
Définition à la ligne 80 du fichier [SH_DateQuestionState.cpp](#).

Références [SH_InOutState ::input\(\)](#), et [SH_GenericState ::toString\(\)](#).

```

00081 {
00082     return QVariant(input().toDate().toString()); /*TODO set format*/
00083 }
  
```

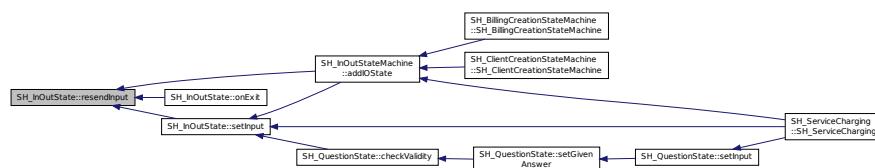
Voici le graphe d'appel pour cette fonction :



4.34.3.15 void SH_InOutState ::resendInput(QVariant input) [signal], [inherited]

Référencé par [SH_InOutStateMachine ::addIOState\(\)](#), [SH_InOutState ::onExit\(\)](#), et [SH_InOutState ::setInput\(\)](#).

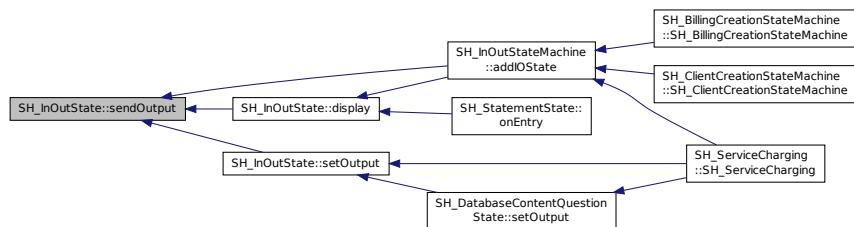
Voici le graphe des appelants de cette fonction :



4.34.3.16 void SH_InOutState ::sendOutput(QVariant output) [signal], [inherited]

Référencé par [SH_InOutStateMachine ::addIOState\(\)](#), [SH_InOutState ::display\(\)](#), et [SH_InOutState ::setOutput\(\)](#).

Voici le graphe des appelants de cette fonction :



4.34.3.17 void SH_DateQuestionState::setFuture (bool value)

Définition à la ligne 70 du fichier [SH_DateQuestionState.cpp](#).

Références [m_future](#).

```

00071 {
00072     m_future = value;
00073 }
```

4.34.3.18 void SH_QuestionState::setGivenAnswer (const QVariant & givenAnswer) [virtual], [inherited]

Définition à la ligne 66 du fichier [SH_QuestionState.cpp](#).

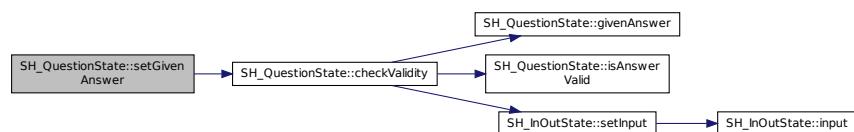
Références [SH_QuestionState::checkValidity\(\)](#), et [SH_QuestionState::m_givenAnswer](#).

Référencé par [SH_QuestionState::setInput\(\)](#).

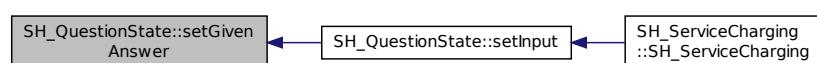
```

00067 {
00068     this->m_givenAnswer = givenAnswer;
00069     this->checkValidity();
00070 }
```

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :



4.34.3.19 void SH_QuestionState :: setInput (const QVariant & input) [virtual], [inherited]

Réimplémentée à partir de [SH_InOutState](#).

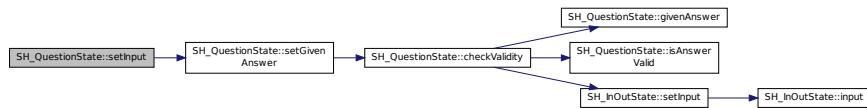
Définition à la ligne 43 du fichier [SH_QuestionState.cpp](#).

Références [SH_QuestionState ::setGivenAnswer\(\)](#).

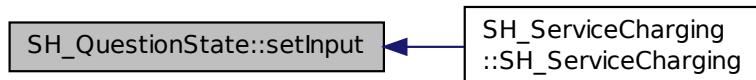
Référencé par [SH_ServiceCharging ::SH_ServiceCharging\(\)](#).

```
00044 {
00045     qDebug() << "new answer " << input.toString();
00046     this->setGivenAnswer(input);
00047 }
```

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appels de cette fonction :



4.34.3.20 void SH_InOutState :: setOutput (const QString & output) [virtual], [slot], [inherited]

Réimplémentée dans [SH_DatabaseContentQuestionState](#).

Définition à la ligne 68 du fichier [SH_IOState.cpp](#).

Références [SH_InOutState ::m_isVisible](#), [SH_InOutState ::m_output](#), [SH_InOutState ::output\(\)](#), et [SH_InOutState ::sendOutput\(\)](#).

Référencé par [SH_DatabaseContentQuestionState ::setOutput\(\)](#), et [SH_ServiceCharging ::SH_ServiceCharging\(\)](#).

```
00069 {
00070     m_output = output;
00071     if(m_isVisible) {
00072         emit sendOutput(QVariant(m_output));
00073     }
00074 }
```

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :



4.34.3.21 void SH_DateQuestionState ::setPast (bool value)

Définition à la ligne 48 du fichier [SH_DateQuestionState.cpp](#).

Références [m_past](#).

```

00049 {
00050     m_past = value;
00051 }
```

4.34.3.22 void SH_InOutState ::setVisibility (bool isVisible) [virtual], [slot], [inherited]

Définition à la ligne 81 du fichier [SH_IOState.cpp](#).

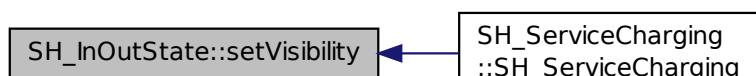
Références [SH_InOutState ::m_isVisible](#).

Référencé par [SH_ServiceCharging ::SH_ServiceCharging\(\)](#).

```

00082 {
00083     m_isVisible = isVisible;
00084 }
```

Voici le graphe des appelants de cette fonction :



4.34.3.23 QString SH_GenericState::toString() [virtual], [inherited]

Réimplémentée à partir de [SH_NamedObject](#).

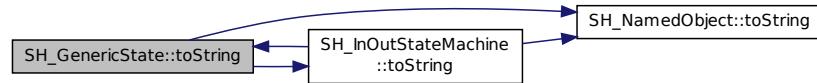
Définition à la ligne 21 du fichier [SH_GenericDebugableState.cpp](#).

Références [SH_NamedObject::toString\(\)](#), et [SH_InOutStateMachine::toString\(\)](#).

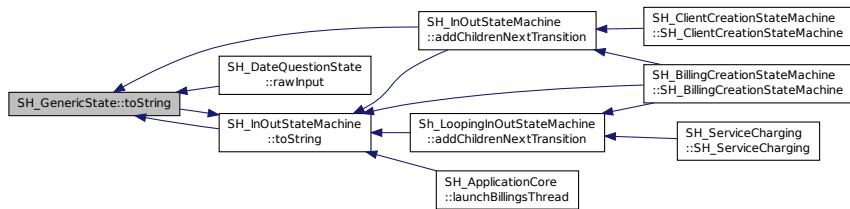
Référencé par [SH_InOutStateMachine::addChildsNextTransition\(\)](#), [rawInput\(\)](#), et [SH_InOutStateMachine::toString\(\)](#).

```
00022 {
00023     QStateMachine* machine = this->machine();
00024     SH_InOutStateMachine* mach = qobject_cast<
00025         SH_InOutStateMachine * >(machine);
00026     if(mach) {
00027         return SH_NamedObject::toString() + " [" + mach->
00028             toString() + "]";
00029     } else {
00030         return SH_NamedObject::toString();
00031     }
00030 }
```

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appels de cette fonction :



4.34.3.24 bool SH_InOutState::visibility() [inherited]

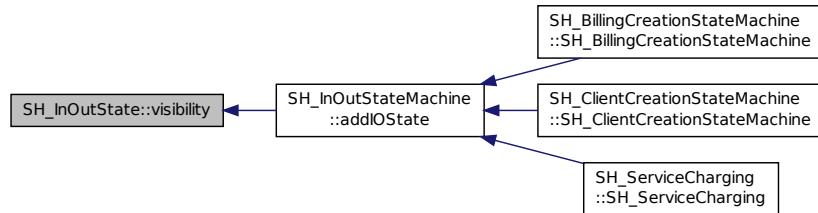
Définition à la ligne 91 du fichier [SH_IOState.cpp](#).

Références [SH_InOutState::m_isVisible](#).

Référencé par [SH_InOutStateMachine::addIOState\(\)](#).

```
00091 {
00092     return m_isVisible;
00093 }
```

Voici le graphe des appelants de cette fonction :



4.34.4 Documentation des données membres

4.34.4.1 bool SH_DateQuestionState ::m_future [private]

m_future

Définition à la ligne 82 du fichier [SH_DateQuestionState.h](#).

Référencé par [getFuture\(\)](#), [isAnswerValid\(\)](#), et [setFuture\(\)](#).

4.34.4.2 bool SH_DateQuestionState ::m_past [private]

m_past

Définition à la ligne 78 du fichier [SH_DateQuestionState.h](#).

Référencé par [getPast\(\)](#), [isAnswerValid\(\)](#), et [setPast\(\)](#).

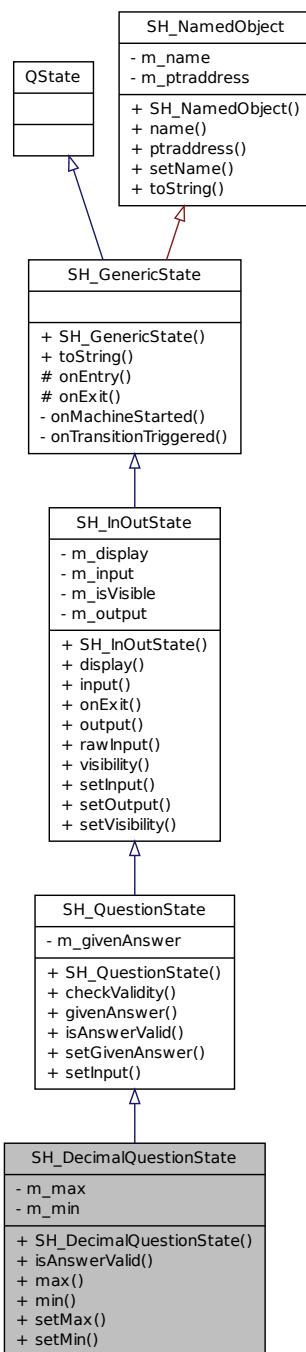
La documentation de cette classe a été générée à partir des fichiers suivants :

- logic/[SH_DateQuestionState.h](#)
- logic/[SH_DateQuestionState.cpp](#)

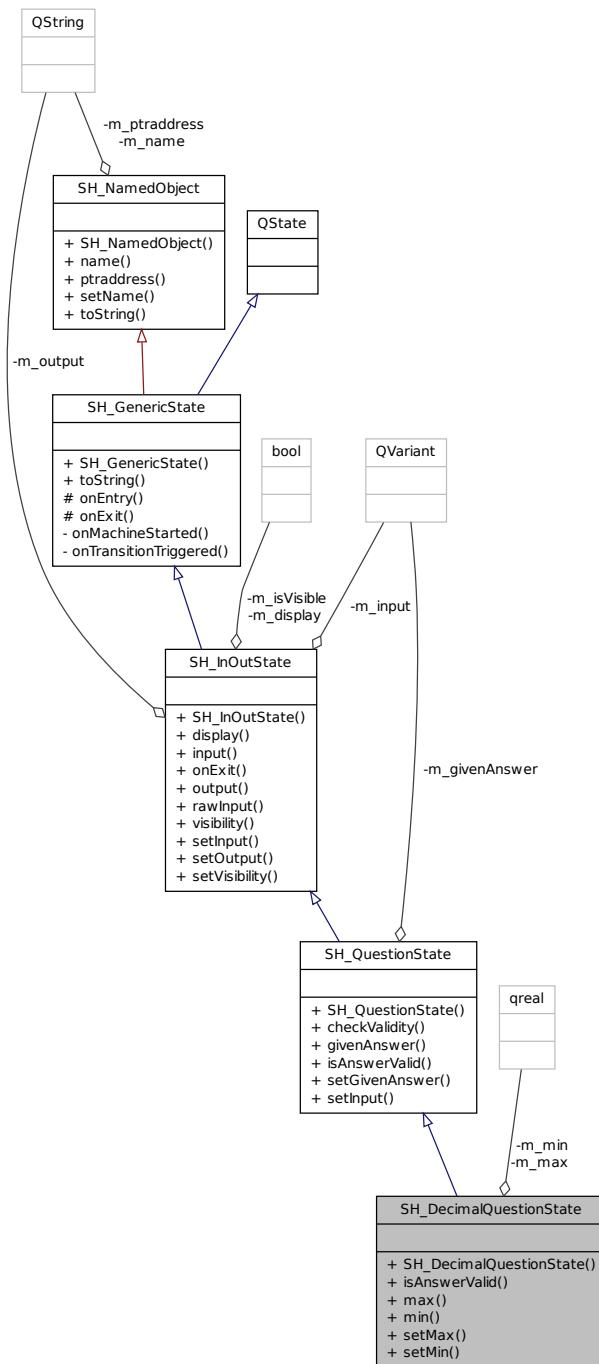
4.35 Référence de la classe SH_DecimalQuestionState

```
#include <SH_DecimalQuestionState.h>
```

Graphe d'héritage de SH_DecimalQuestionState :



Graphe de collaboration de SH_DecimalQuestionState :



Connecteurs publics

- `virtual void setOutput (const QString &output)`
- `virtual void setVisibility (bool isVisible)`

Signaux

- `void answerInvalid ()`

```

    answerInvalid
- void answerValid ()
    answerValid
- void next ()
- void resendInput (QVariant input)
- void sendOutput (QVariant output)

```

Fonctions membres publiques

```

- SH_DecimalQuestionState (QString question, QString name, qreal min=0, qreal max=-1, QState *parent=0)
- bool checkValidity ()
- void display (bool canDisplay)
- virtual QVariant givenAnswer () const
- virtual QVariant input () const
- virtual bool isValid (const QVariant &givenAnswer)
- qreal max () const
- qreal min () const
- void onExit (QEvent *event)
- virtual QString output () const
- virtual QVariant rawInput () const
- virtual void setGivenAnswer (const QVariant &givenAnswer)
- virtual void setInput (const QVariant &input)
- void setMax (const qreal &max)
- void setMin (const qreal &min)
- QString toString ()
- bool visibility ()

```

Fonctions membres protégées

```
- void onEntry (QEvent *event)
```

Attributs privés

```

- qreal m_max
    m_max
- qreal m_min
    m_min

```

4.35.1 Description détaillée

Définition à la ligne 11 du fichier [SH_DecimalQuestionState.h](#).

4.35.2 Documentation des constructeurs et destructeur

4.35.2.1 SH_DecimalQuestionState : :SH_DecimalQuestionState (QString *question*, QString *name*, qreal *min* = 0, qreal *max* = -1, QState * *parent* = 0)

Paramètres

<i>question</i>	
<i>name</i>	
<i>min</i>	
<i>max</i>	
<i>parent</i>	

Définition à la ligne 10 du fichier [SH_DecimalQuestionState.cpp](#).

```

00010
:
00011      SH_QuestionState(question, name, parent), m_min(min),
        m_max(max)
00012 {
00013

```

```
00014 }
```

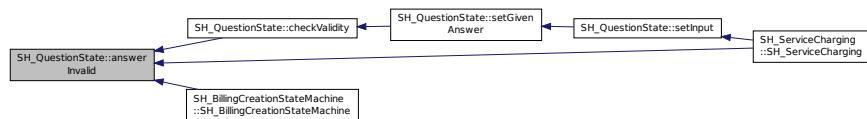
4.35.3 Documentation des fonctions membres

4.35.3.1 void SH_QuestionState ::answerInvalid() [signal], [inherited]

answerInvalid

Référencé par [SH_QuestionState ::checkValidity\(\)](#), [SH_BillingCreationStateMachine ::SH_BillingCreationStateMachine\(\)](#), et [SH_ServiceCharging ::SH_ServiceCharging\(\)](#).

Voici le graphe des appelants de cette fonction :

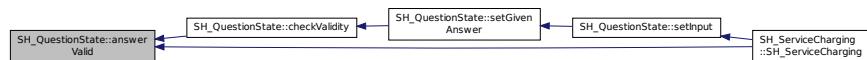


4.35.3.2 void SH_QuestionState ::answerValid() [signal], [inherited]

answerValid

Référencé par [SH_QuestionState ::checkValidity\(\)](#), et [SH_ServiceCharging ::SH_ServiceCharging\(\)](#).

Voici le graphe des appelants de cette fonction :



4.35.3.3 bool SH_QuestionState ::checkValidity() [inherited]

Définition à la ligne 20 du fichier [SH_QuestionState.cpp](#).

Références [SH_QuestionState ::answerInvalid\(\)](#), [SH_QuestionState ::answerValid\(\)](#), [SH_QuestionState ::givenAnswer\(\)](#), [SH_QuestionState ::isValid\(\)](#), [SH_GenericState ::next\(\)](#), et [SH_InOutState ::setInput\(\)](#).

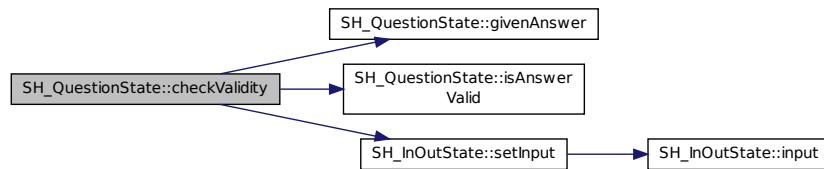
Référencé par [SH_QuestionState ::setGivenAnswer\(\)](#).

```

00021 {
00022     bool ok = this->isValid\(this->givenAnswer\(\)\);
00023     if(ok) {
00024         qDebug() << this->givenAnswer() << " answer valid !";
00025         qDebug() << this->givenAnswer();
00026         SH\_InOutState::setInput\(this->givenAnswer\(\)\);
00027         emit answerValid\(\);
00028         emit next\(\);
00029     } else {
00030         qDebug() << this->givenAnswer() << " answer invalid :-(";
00031         qDebug() << this->givenAnswer();
00032         emit answerInvalid\(\);
00033     }
00034     return ok;
00035 }

```

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appels de cette fonction :



4.35.3.4 void SH_InOutState ::display (bool canDisplay) [inherited]

Définition à la ligne 95 du fichier [SH_IOState.cpp](#).

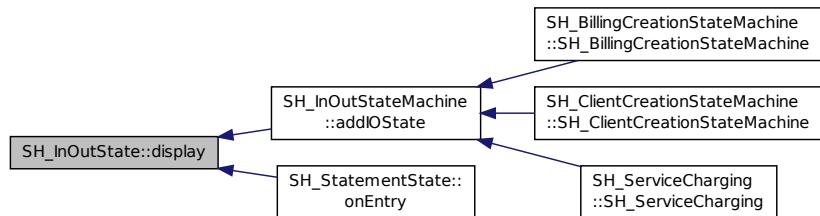
Références [SH_InOutState ::m_display](#), [SH_InOutState ::m_isVisible](#), [SH_InOutState ::m_output](#), et [SH_InOutState ::sendOutput\(\)](#).

Référencé par [SH_InOutStateMachine ::addIOState\(\)](#), et [SH_StatementState ::onEntry\(\)](#).

```

00096 {
00097     m_display=canDisplay;
00098     if(m_display && !m_output.isEmpty() && m_isVisible) {
00099         qDebug() << "resalut !" << QVariant(m_output);
00100         emit sendOutput(QVariant(m_output));
00101     }
00102 }
```

Voici le graphe des appels de cette fonction :



4.35.3.5 QVariant SH_QuestionState ::givenAnswer () const [virtual], [inherited]

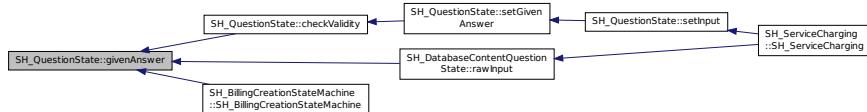
Définition à la ligne 55 du fichier [SH_QuestionState.cpp](#).

Références [SH_QuestionState ::m_givenAnswer](#).

Référencé par [SH_QuestionState ::checkValidity\(\)](#), [SH_DatabaseContentQuestionState ::rawInput\(\)](#), et [SH_BillingCreationStateMachine ::SH_BillingCreationStateMachine\(\)](#).

```
00056 {
00057     return this->m_givenAnswer;
00058 }
```

Voici le graphe des appelants de cette fonction :



4.35.3.6 QVariant SH_InOutState ::input() const [virtual], [inherited]

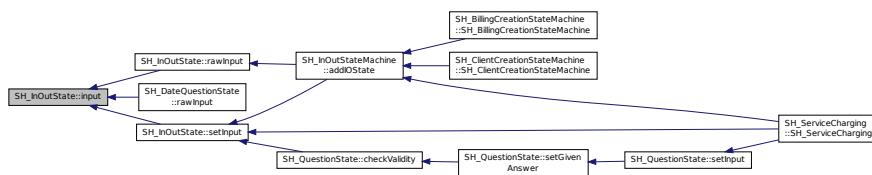
Définition à la ligne 20 du fichier [SH_IOState.cpp](#).

Références [SH_InOutState ::m_input](#).

Référencé par [SH_InOutState ::rawInput\(\)](#), [SH_DateQuestionState ::rawInput\(\)](#), et [SH_InOutState ::setInput\(\)](#).

```
00021 {
00022     return m_input;
00023 }
```

Voici le graphe des appelants de cette fonction :



4.35.3.7 bool SH_DecimalQuestionState ::isValid(const QVariant & givenAnswer) [virtual]

Implémente [SH_QuestionState](#).

Définition à la ligne 22 du fichier [SH_DecimalQuestionState.cpp](#).

Références [m_max](#), et [m_min](#).

```
00023 {
00024     bool ok;
00025     qreal answer = givenAnswer.toReal(&ok);
00026     if(ok) {
00027         return ((m_max <= m_min || answer <= m_max) && answer >=
00028             m_min);
00029     } else {
00030         return false;
00031     }
  
```

4.35.3.8 qreal SH_DecimalQuestionState ::max() const

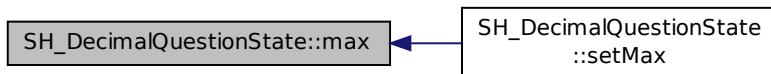
Définition à la ligne 61 du fichier [SH_DecimalQuestionState.cpp](#).

Références [m_max](#).

Référencé par [setMax\(\)](#).

```
00062 {  
00063     return m_max;  
00064 }
```

Voici le graphe des appelants de cette fonction :



4.35.3.9 qreal SH_DecimalQuestionState ::min() const

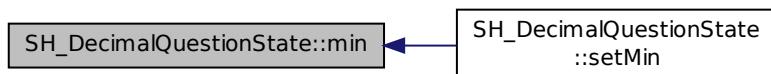
Définition à la ligne 39 du fichier [SH_DecimalQuestionState.cpp](#).

Références [m_min](#).

Référencé par [setMax\(\)](#).

```
00040 {  
00041     return m_min;  
00042 }
```

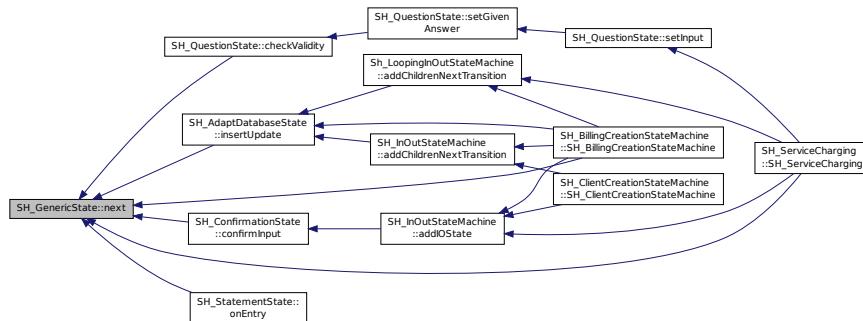
Voici le graphe des appelants de cette fonction :



4.35.3.10 void SH_GenericState ::next() [signal], [inherited]

Référencé par [SH_QuestionState ::checkValidity\(\)](#), [SH_ConfirmationState ::confirmInput\(\)](#), [SH_AdaptDatabase-State ::insertUpdate\(\)](#), [SH_StatementState ::onEntry\(\)](#), [SH_BillingCreationStateMachine ::SH_BillingCreation-StateMachine\(\)](#), et [SH_ServiceCharging ::SH_ServiceCharging\(\)](#).

Voici le graphe des appelants de cette fonction :



4.35.3.11 void SH_GenericState ::onEntry (QEvent * event) [protected], [inherited]

Définition à la ligne 62 du fichier [SH_GenericDebugableState.cpp](#).

Références [SH_NamedObject ::name\(\)](#).

Référencé par [SH_StatementState ::onEntry\(\)](#).

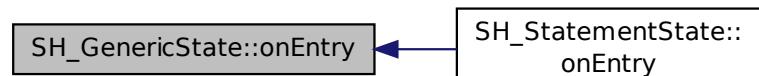
```

00063 {
00064     Q_UNUSED(event);
00065     qDebug() << "Machine: " << machine()->objectName() << " entered " << name();
00066 }
```

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :



4.35.3.12 void SH_InOutState ::onExit (QEvent * event) [inherited]

Définition à la ligne 110 du fichier [SH_IOState.cpp](#).

Références [SH_InOutState ::m_input](#), [SH_InOutState ::m_isVisible](#), [SH_GenericState ::onExit\(\)](#), et [SH_InOutState ::resendInput\(\)](#).

```

00111 {
00112     SH_GenericState::onExit(event);
00113     if(m_isVisible) {
00114         emit resendInput(m_input);
00115     }
00116 }
```

Voici le graphe d'appel pour cette fonction :



4.35.3.13 QString SH_InOutState::output() const [virtual], [inherited]

Définition à la ligne 56 du fichier [SH_IOState.cpp](#).

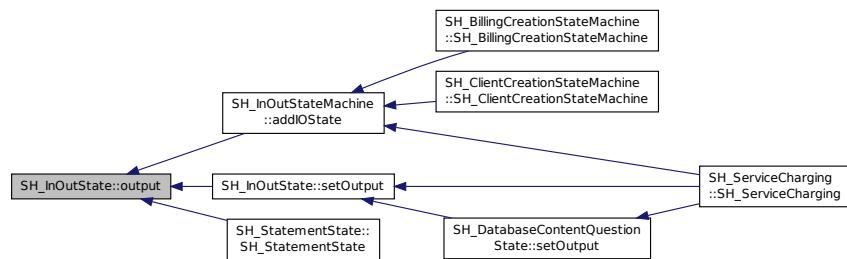
Références [SH_InOutState::m_output](#).

Référencé par [SH_InOutStateMachine::addIOState\(\)](#), [SH_InOutState::setOutput\(\)](#), et [SH_StatementState::SH_StatementState\(\)](#).

```

00057 {
00058     return m_output;
00059 }
```

Voici le graphe des appels de cette fonction :



4.35.3.14 QVariant SH_InOutState::rawInput() const [virtual], [inherited]

Réimplémentée dans [SH_DateQuestionState](#), et [SH_DatabaseContentQuestionState](#).

Définition à la ligne 30 du fichier [SH_IOState.cpp](#).

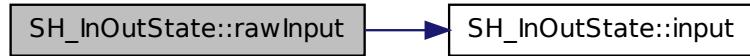
Références [SH_InOutState::input\(\)](#).

Référencé par [SH_InOutStateMachine::addIOState\(\)](#).

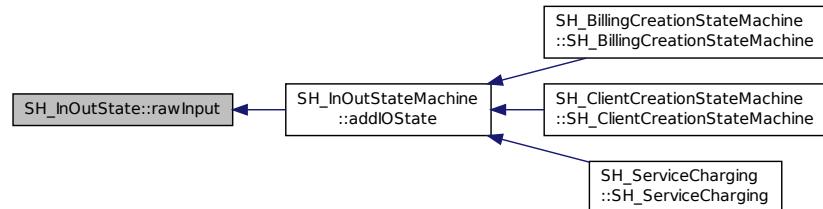
```

00031 {
00032     return input();
00033 }
```

Voici le graphe d'appel pour cette fonction :



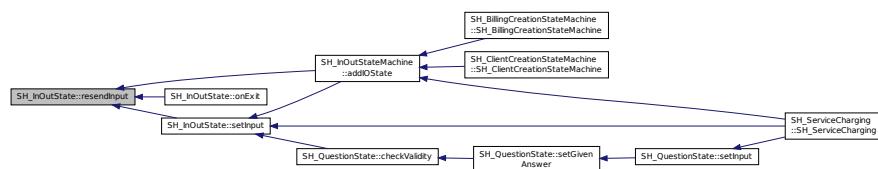
Voici le graphe des appels de cette fonction :



4.35.3.15 void SH_InOutState ::resendInput (QVariant *input*) [signal], [inherited]

Référencé par [SH_InOutStateMachine ::addIOState\(\)](#), [SH_InOutState ::onExit\(\)](#), et [SH_InOutState ::setInput\(\)](#).

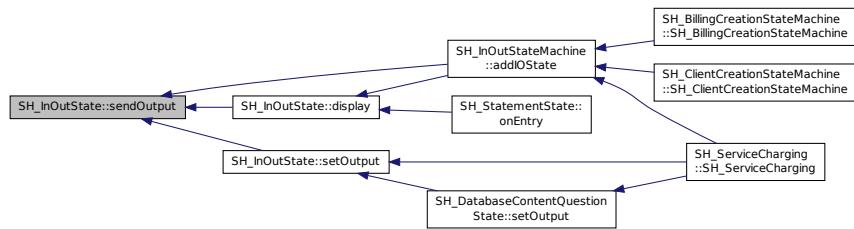
Voici le graphe des appels de cette fonction :



4.35.3.16 void SH_InOutState ::sendOutput (QVariant *output*) [signal], [inherited]

Référencé par [SH_InOutStateMachine ::addIOState\(\)](#), [SH_InOutState ::display\(\)](#), et [SH_InOutState ::setOutput\(\)](#).

Voici le graphe des appelants de cette fonction :



4.35.3.17 void SH_QuestionState : :setGivenAnswer (const QVariant & givenAnswer) [virtual], [inherited]

Définition à la ligne 66 du fichier [SH_QuestionState.cpp](#).

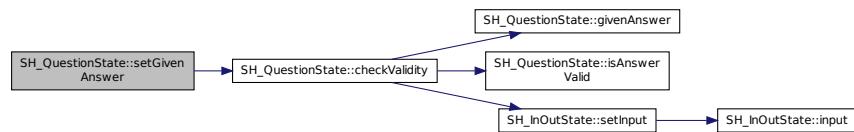
Références [SH_QuestionState : :checkValidity\(\)](#), et [SH_QuestionState : :m_givenAnswer](#).

Référencé par [SH_QuestionState : :setInput\(\)](#).

```

00067 {
00068     this->m_givenAnswer = givenAsnwer;
00069     this->checkValidity();
00070 }
  
```

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :



4.35.3.18 void SH_QuestionState : :setInput (const QVariant & input) [virtual], [inherited]

Réimplémentée à partir de [SH_InOutState](#).

Définition à la ligne 43 du fichier [SH_QuestionState.cpp](#).

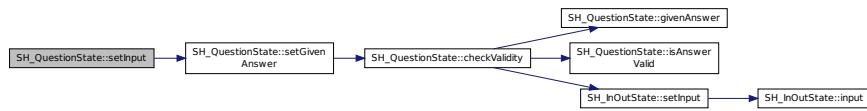
Références [SH_QuestionState : :setGivenAnswer\(\)](#).

Référencé par [SH_ServiceCharging : :SH_ServiceCharging\(\)](#).

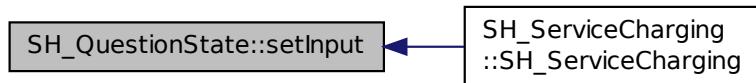
```

00044 {
00045     qDebug() << "new answer " << input.toString();
00046     this->setGivenAnswer(input);
00047 }
  
```

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appels de cette fonction :



4.35.3.19 void SH_DecimalQuestionState ::setMax (const qreal & max)

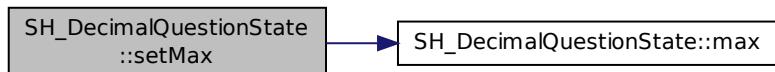
Définition à la ligne 72 du fichier [SH_DecimalQuestionState.cpp](#).

Références [m_max](#), et [max\(\)](#).

```

00073 {
00074     m_max = max;
00075 }
```

Voici le graphe d'appel pour cette fonction :



4.35.3.20 void SH_DecimalQuestionState ::setMin (const qreal & min)

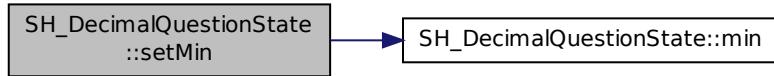
Définition à la ligne 50 du fichier [SH_DecimalQuestionState.cpp](#).

Références [m_min](#), et [min\(\)](#).

```

00051 {
00052     m_min = min;
00053 }
```

Voici le graphe d'appel pour cette fonction :



4.35.3.21 void SH_InOutState ::setOutput (const QString & output) [virtual], [slot], [inherited]

Réimplémentée dans [SH_DatabaseContentQuestionState](#).

Définition à la ligne 68 du fichier [SH_IOState.cpp](#).

Références [SH_InOutState ::m_isVisible](#), [SH_InOutState ::m_output](#), [SH_InOutState ::output\(\)](#), et [SH_InOutState ::sendOutput\(\)](#).

Référencé par [SH_DatabaseContentQuestionState ::setOutput\(\)](#), et [SH_ServiceCharging ::SH_ServiceCharging\(\)](#).

```

00069 {
00070     m_output = output;
00071     if(m_isVisible) {
00072         emit sendOutput(QVariant(m_output));
00073     }
00074 }
```

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :



4.35.3.22 void SH_InOutState ::setVisibility (bool isVisible) [virtual], [slot], [inherited]

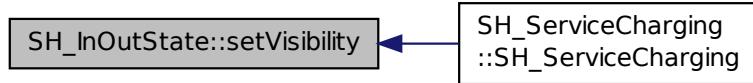
Définition à la ligne 81 du fichier [SH_IOState.cpp](#).

Références [SH_InOutState ::m_isVisible](#).

Référencé par [SH_ServiceCharging ::SH_ServiceCharging\(\)](#).

```
00082 {
00083     m_isVisible = isVisible;
00084 }
```

Voici le graphe des appels de cette fonction :



4.35.3.23 QString SH_GenericState : toString() [virtual], [inherited]

Réimplémentée à partir de [SH_NamedObject](#).

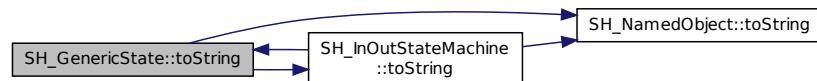
Définition à la ligne 21 du fichier [SH_GenericDebugableState.cpp](#).

Références [SH_NamedObject : toString\(\)](#), et [SH_InOutStateMachine : toString\(\)](#).

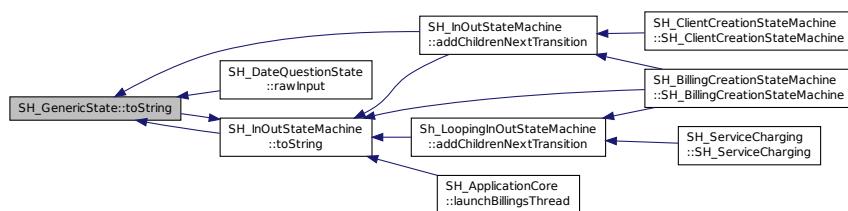
Référencé par [SH_InOutStateMachine : addChildrenNextTransition\(\)](#), [SH_DateQuestionState : rawInput\(\)](#), et [SH_InOutStateMachine : toString\(\)](#).

```
00022 {
00023     QStateMachine* machine = this->machine();
00024     SH_InOutStateMachine* mach = qobject_cast<
00025         SH_InOutStateMachine * >(machine);
00026     if(mach) {
00027         return SH_NamedObject::toString() + " [in "+mach->
00028             toString()+"] ";
00029     } else {
00030         return SH_NamedObject::toString();
00031     }
00030 }
```

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appels de cette fonction :



4.35.3.24 bool SH_InOutState ::visibility() [inherited]

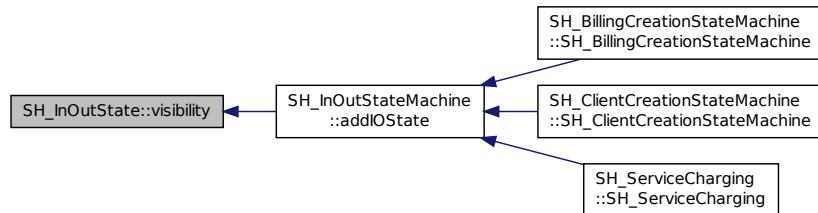
Définition à la ligne 91 du fichier [SH_IoState.cpp](#).

Références [SH_InOutState ::m_isVisible](#).

Référencé par [SH_InOutStateMachine ::addIOState\(\)](#).

```
00091     {
00092         return m_isVisible;
00093     }
```

Voici le graphe des appels de cette fonction :



4.35.4 Documentation des données membres

4.35.4.1 qreal SH_DecimalQuestionState ::m_max [private]

m_max

Définition à la ligne 76 du fichier [SH_DecimalQuestionState.h](#).

Référencé par [isAnswerValid\(\)](#), [max\(\)](#), et [setMax\(\)](#).

4.35.4.2 qreal SH_DecimalQuestionState ::m_min [private]

m_min

Définition à la ligne 72 du fichier [SH_DecimalQuestionState.h](#).

Référencé par [isAnswerValid\(\)](#), [min\(\)](#), et [setMin\(\)](#).

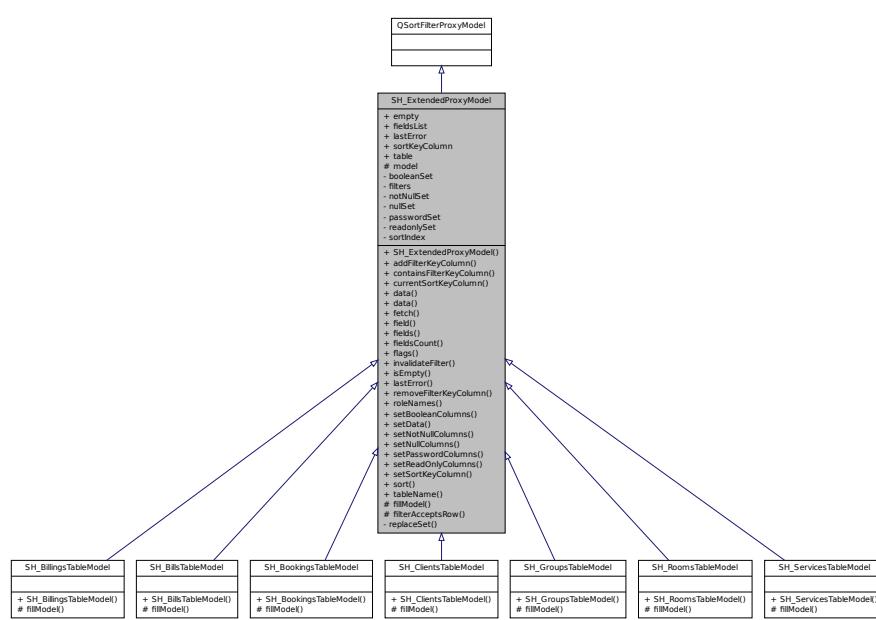
La documentation de cette classe a été générée à partir des fichiers suivants :

- logic/[SH_DecimalQuestionState.h](#)
- logic/[SH_DecimalQuestionState.cpp](#)

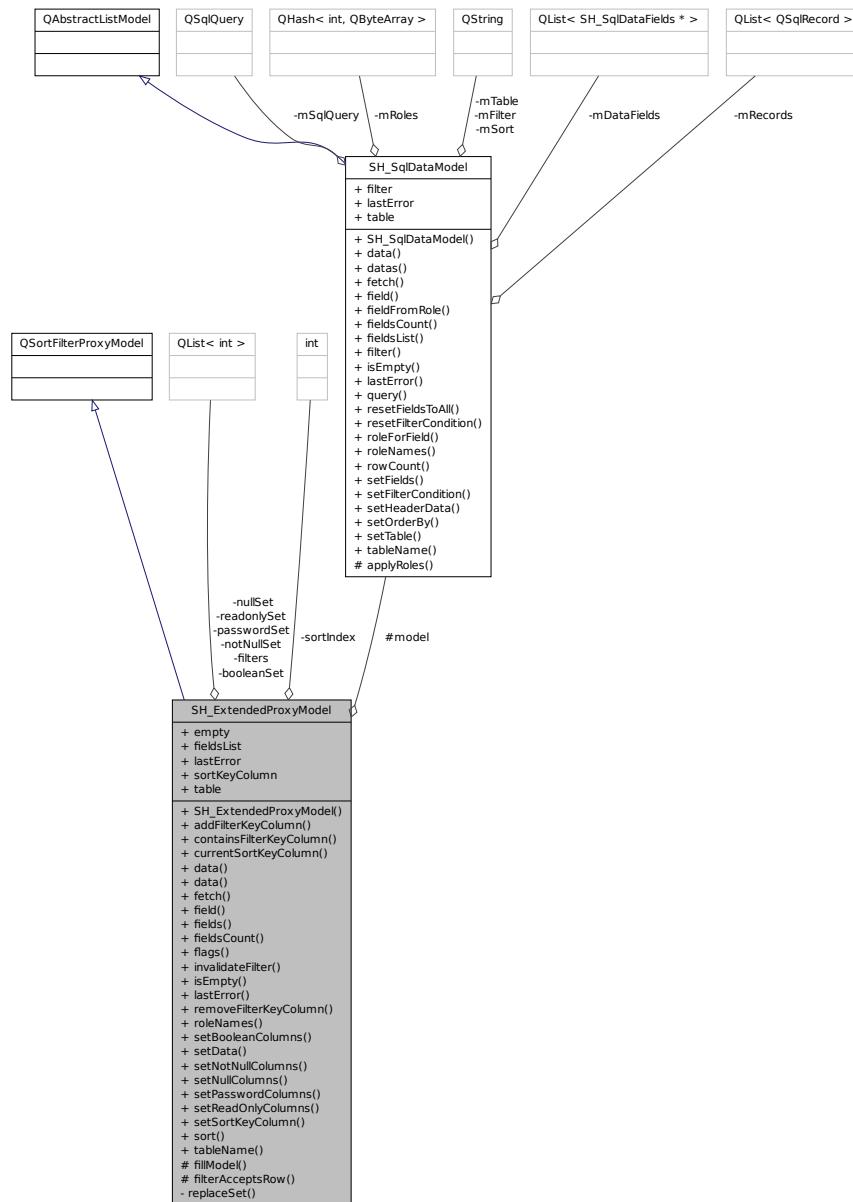
4.36 Référence de la classe SH_ExtendedProxyModel

```
#include <SH_ExtendedSqlProxyModel.h>
```

Graphe d'héritage de SH_ExtendedProxyModel :



Graphe de collaboration de SH_ExtendedProxyModel :



Signaux

- void `sortChanged ()`

Fonctions membres publiques

- `SH_ExtendedProxyModel (QObject *parent=0)`
- `Q_INVOKABLE void addFilterKeyColumn (int column)`
- `Q_INVOKABLE bool containsFilterKeyColumn (int column)`
- `const int currentSortKeyColumn () const`
- `Q_INVOKABLE QVariant data (int row, int column) const`
- `Q_INVOKABLE QVariant data (const QModelIndex &index, int role=Qt::DisplayRole) const`
- `Q_INVOKABLE bool fetch (QString tableName="", QString filter="", QString sort="", QStringList fields=QStringList())`
- `Q_INVOKABLE SH_SqlDataFields * field (int i) const`

- const QString **fields** () const
- Q_INVOKABLE int **fieldsCount** () const
- Q_INVOKABLE Qt : :ItemFlags **flags** (const QModelIndex &index) const
- void **invalidateFilter** ()
- const bool **isEmpty** () const
- const QString **lastError** () const
- Q_INVOKABLE void **removeFilterKeyColumn** (int column)
- virtual Q_INVOKABLE QHash< int,
 QByteArray > **roleNames** () const
- void **setBooleanColumns** (QList< int > boolCols)
- Q_INVOKABLE bool **setData** (const QModelIndex &index, const QVariant &value, int role=Qt : :EditRole)
- void **setNotNullColumns** (QList< int > notNullCols)
- void **setNullColumns** (QList< int > nullCols)
- void **setPasswordColumns** (QList< int > passwordCols)
- void **setReadOnlyColumns** (QList< int > readonlyCols)
- void **setSortKeyColumn** (int column)
- Q_INVOKABLE void **sort** (int column, Qt : :SortOrder newOrder=Qt : :AscendingOrder)
- const QString **tableName** () const

Fonctions membres protégées

- virtual void **fillModel** ()=0
- bool **filterAcceptsRow** (int source_row, const QModelIndex &source_parent) const

Attributs protégés

- SH_SqlDataModel * **model**

Propriétés

- bool **empty**
- QString **fieldsList**
- QString **lastError**
- int **sortKeyColumn**
- QString **table**

Fonctions membres privées

- void **replaceSet** (QList< int > &originalSet, QList< int > newSet)

Attributs privés

- QList< int > **booleanSet**
 booleanSet
- QList< int > **filters**
 filters
- QList< int > **notNullSet**
 notNullSet
- QList< int > **nullSet**
 nullSet
- QList< int > **passwordSet**
 passwordSet
- QList< int > **readonlySet**
 readonlySet
- int **sortIndex**
 sortIndex

4.36.1 Description détaillée

Définition à la ligne 13 du fichier [SH_ExtendedSqlProxyModel.h](#).

4.36.2 Documentation des constructeurs et destructeur

4.36.2.1 `SH_ExtendedProxyModel ::SH_ExtendedProxyModel (QObject * parent = 0)`

Définition à la ligne 12 du fichier [SH_ExtendedProxyModel.cpp](#).

Références [model](#), et [sortIndex](#).

```
00012
00013     QSortFilterProxyModel(parent)
00014 {
00015     this->setDynamicSortFilter(false);
00016     this->model = new SH_SqlDataModel(parent);
00017     this->setSourceModel(this->model);
00018     this->sortIndex = 0;
00019     /*connect (this->model, tableChanged(), tableName());*/
00020 }
```

4.36.3 Documentation des fonctions membres

4.36.3.1 `void SH_ExtendedProxyModel ::addFilterKeyColumn (int column)`

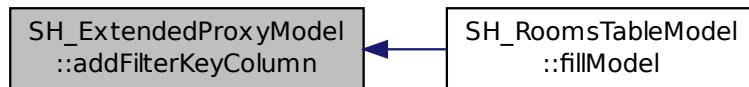
Définition à la ligne 259 du fichier [SH_ExtendedProxyModel.cpp](#).

Références [filters](#).

Référencé par [SH_RoomsTableModel ::fillModel\(\)](#).

```
00260 {
00261     this->filters.append(column);
00262 }
```

Voici le graphe des appelants de cette fonction :



4.36.3.2 `bool SH_ExtendedProxyModel ::containsFilterKeyColumn (int column)`

Définition à la ligne 225 du fichier [SH_ExtendedProxyModel.cpp](#).

Références [filters](#).

```
00226 {
00227     return this->filters.contains(column);
00228 }
```

4.36.3.3 `const int SH_ExtendedProxyModel ::currentSortKeyColumn () const [inline]`

Définition à la ligne 38 du fichier [SH_ExtendedSqlProxyModel.h](#).

Références [sortIndex](#).

```
00038 {return this->sortIndex;}
```

4.36.3.4 QVariant SH_ExtendedProxyModel::data (int row, int column) const

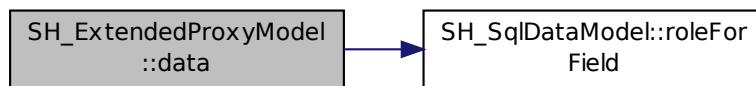
Définition à la ligne 269 du fichier [SH_ExtendedProxyModel.cpp](#).

Références [model](#), et [SH_SqlDataModel::roleForField\(\)](#).

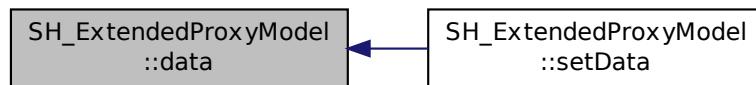
Référencé par [setData\(\)](#).

```
00270 {
00271     QModelIndex modelIndex = this->index(row, 0);
00272     return this->data(modelIndex, this->model->roleForField(column));
00273 }
```

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appels de cette fonction :



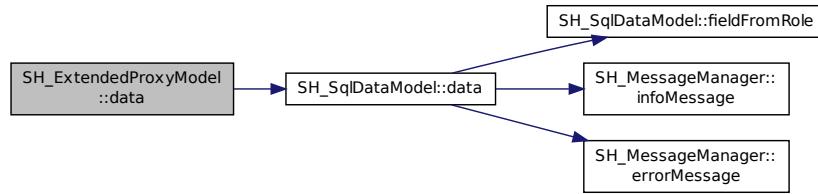
4.36.3.5 QVariant SH_ExtendedProxyModel::data (const QModelIndex & index, int role = Qt::DisplayRole) const

Définition à la ligne 127 du fichier [SH_ExtendedProxyModel.cpp](#).

Références [booleanSet](#), [SH_SqlDataModel::data\(\)](#), [filters](#), [model](#), et [passwordSet](#).

```
00128 {
00129     if (index.isValid())
00130     {
00131         if (this->booleanSet.contains(role))
00132         {
00133             return index.data(Qt::EditRole).toBool() ? QVariant(Qt::Checked) : QVariant(Qt::Unchecked);
00134         }
00135         else if (this->passwordSet.contains(role))
00136         {
00137             return QVariant("****");
00138         }
00139         else if (!this->filters.contains(role))
00140         {
00141             QModelIndex source_index = QSortFilterProxyModel::mapToSource(index);
00142             if (source_index.isValid())
00143                 return this->model->data(source_index, role);
00144             }
00145         }
00146     }
00147     return QVariant();
00148 }
```

Voici le graphe d'appel pour cette fonction :



4.36.3.6 bool SH_ExtendedProxyModel ::fetch (QString tableName = " ", QString filter = " ", QString sort = " ", QStringList fields = QStringList ())

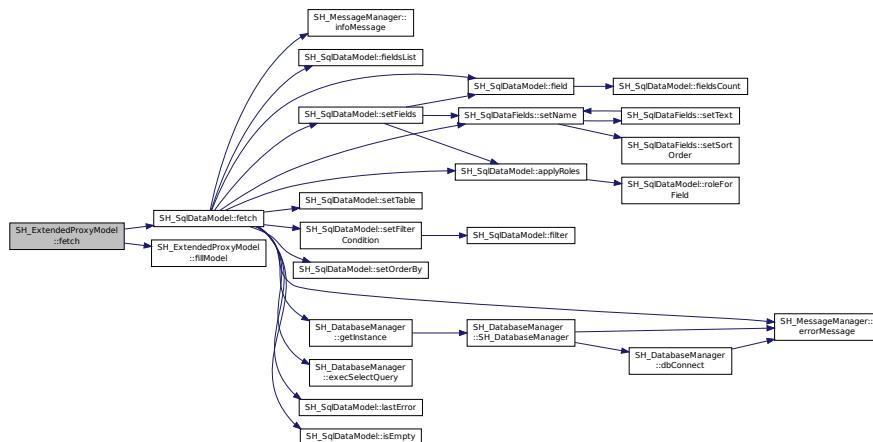
Définition à la ligne 280 du fichier [SH_ExtendedProxyModel.cpp](#).

Références [SH_SqlDataModel ::fetch\(\)](#), [fillModel\(\)](#), et [model](#).

```

00281 {
00282     bool fetched = this->model->fetch(tableName, filter, sort,
00283         fields);
00284     if (fetched)
00285         this->fillModel();
00286     this->setSourceModel(this->model);
00287     return fetched;
00288 }
00289 }
```

Voici le graphe d'appel pour cette fonction :



4.36.3.7 Q_INVOKABLE SH_SqlDataFields* SH_ExtendedProxyModel ::field (int i) const [inline]

Définition à la ligne 82 du fichier [SH_ExtendedSqlProxyModel.h](#).

Références [SH_SqlDataModel ::field\(\)](#), et [model](#).

```
00082 { return this->model->field(i); }
```

Voici le graphe d'appel pour cette fonction :



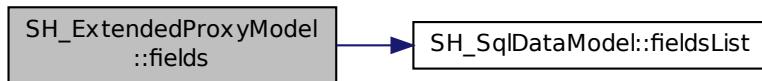
4.36.3.8 const QString SH_ExtendedProxyModel ::fields() const [inline]

Définition à la ligne 52 du fichier [SH_ExtendedSqlProxyModel.h](#).

Références [SH_SqlDataModel ::fieldsList\(\)](#), et [model](#).

```
00052 { if(this->model->fieldsList().isEmpty()){ return "*"; } else { return this->
model->fieldsList().join(", "); } }
```

Voici le graphe d'appel pour cette fonction :



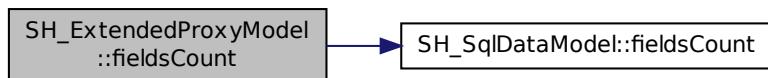
4.36.3.9 Q_INVOKABLE int SH_ExtendedProxyModel ::fieldsCount() const [inline]

Définition à la ligne 89 du fichier [SH_ExtendedSqlProxyModel.h](#).

Références [SH_SqlDataModel ::fieldsCount\(\)](#), et [model](#).

```
00089 { return this->model->fieldsCount(); }
```

Voici le graphe d'appel pour cette fonction :

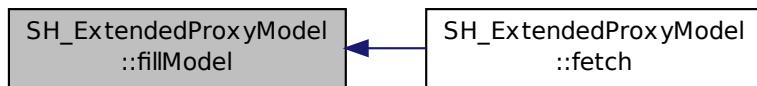


4.36.3.10 virtual void SH_ExtendedProxyModel ::fillModel() [protected], [pure virtual]

Implémenté dans [SH_BookingsTableModel](#), [SH_ClientsTableModel](#), [SH_GroupsTableModel](#), [SH_BillingsTableModel](#), [SH_BillsTableModel](#), [SH_RoomsTableModel](#), et [SH_ServicesTableModel](#).

Référencé par [fetch\(\)](#).

Voici le graphe des appels de cette fonction :



4.36.3.11 bool SH_ExtendedProxyModel::filterAcceptsRow (int source_row, const QModelIndex & source_parent) const [protected]

Définition à la ligne [92](#) du fichier [SH_ExtendedProxyModel.cpp](#).

Références [notNullSet](#), et [nullSet](#).

```

00093 {
00094     Q_UNUSED(source_parent);
00095
00096     if (!this->notNullSet.isEmpty())
00097     {
00098         QSqlQueryModel *m = static_cast<QSqlQueryModel *>(sourceModel());
00099         foreach(int column, this->notNullSet)
00100         {
00101             if (m->record(source_row).isNull(column))
00102             {
00103                 return false;
00104             }
00105         }
00106     }
00107
00108     if (!this->nullSet.isEmpty())
00109     {
00110         QSqlQueryModel *m = static_cast<QSqlQueryModel *>(sourceModel());
00111         foreach(int column, this->nullSet)
00112         {
00113             if (!m->record(source_row).isNull(column))
00114             {
00115                 return false;
00116             }
00117         }
00118     }
00119     return true;
00120 }

```

4.36.3.12 Qt::ItemFlags SH_ExtendedProxyModel::flags (const QModelIndex & index) const

Définition à la ligne [179](#) du fichier [SH_ExtendedProxyModel.cpp](#).

Références [booleanSet](#), et [readonlySet](#).

```

00180 {
00181     if (!index.isValid())
00182     {
00183         return Qt::ItemIsEnabled;
00184     }
00185     if (!this->booleanSet.isEmpty())
00186     {
00187         return Qt::ItemIsUserCheckable | Qt::ItemIsSelectable | Qt::ItemIsEnabled;
00188     }
00189     else if (!this->readonlySet.isEmpty())
00190     {
00191         return Qt::ItemIsSelectable;
00192     }
00193     else

```

```

00194     {
00195         return QSortFilterProxyModel::flags(index);
00196     }
00197
00198 }
```

4.36.3.13 void SH_ExtendedProxyModel ::invalidateFilter ()

Définition à la ligne 205 du fichier [SH_ExtendedProxyModel.cpp](#).

Références [filters](#).

```

00206 {
00207     this->filters.clear();
00208 }
```

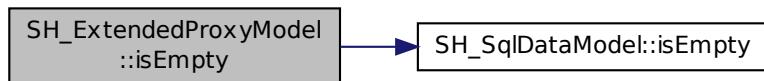
4.36.3.14 const bool SH_ExtendedProxyModel ::isEmpty () const [inline]

Définition à la ligne 66 du fichier [SH_ExtendedSqlProxyModel.h](#).

Références [SH_SqlDataModel ::isEmpty\(\)](#), et [model](#).

```
00066 { return this->model->isEmpty(); }
```

Voici le graphe d'appel pour cette fonction :



4.36.3.15 const QString SH_ExtendedProxyModel ::lastError () const [inline]

Définition à la ligne 59 du fichier [SH_ExtendedSqlProxyModel.h](#).

Références [SH_SqlDataModel ::lastError](#), et [model](#).

```
00059 { return this->model->lastError(); }
```

4.36.3.16 void SH_ExtendedProxyModel ::removeFilterKeyColumn (int column)

Définition à la ligne 215 du fichier [SH_ExtendedProxyModel.cpp](#).

Références [filters](#).

```

00216 {
00217     this->filters.removeAt(this->filters.indexOf(column));
00218 }
```

4.36.3.17 void SH_ExtendedProxyModel ::replaceSet (QList< int > & originalSet, QList< int > newSet) [private]

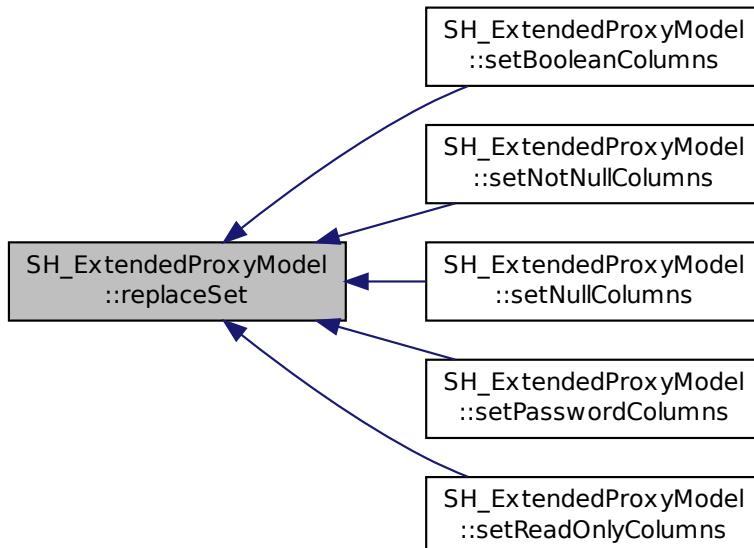
Définition à la ligne 27 du fichier [SH_ExtendedProxyModel.cpp](#).

Référencé par [setBooleanColumns\(\)](#), [setNotNullColumns\(\)](#), [setNullColumns\(\)](#), [setPasswordColumns\(\)](#), et [setReadOnlyColumns\(\)](#).

```

00027
00028     originalSet.clear();
00029     foreach(int col, newSet) {
00030         if(!originalSet.contains(col)) {
00031             originalSet.append(col);
00032         }
00033     }
00034 }
```

Voici le graphe des appels de cette fonction :



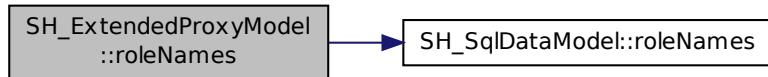
4.36.3.18 virtual Q_INVOKABLE QHash<int, QByteArray> SH_ExtendedProxyModel ::roleNames () const [inline], [virtual]

Définition à la ligne 165 du fichier [SH_ExtendedSqlProxyModel.h](#).

Références [model](#), et [SH_SqlDataManager ::roleNames\(\)](#).

```
00165 { return this->model->roleNames(); }
```

Voici le graphe d'appel pour cette fonction :



4.36.3.19 void SH_ExtendedProxyModel ::setBooleanColumns (QList< int > boolCols)

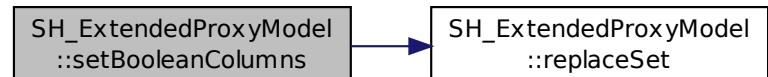
Définition à la ligne 41 du fichier [SH_ExtendedProxyModel.cpp](#).

Références [booleanSet](#), et [replaceSet\(\)](#).

```

00041
00042     replaceSet(this->booleanSet, boolCols);
00043 }
```

Voici le graphe d'appel pour cette fonction :



4.36.3.20 bool SH_ExtendedProxyModel ::setData (const QModelIndex & index, const QVariant & value, int role = Qt::EditRole)

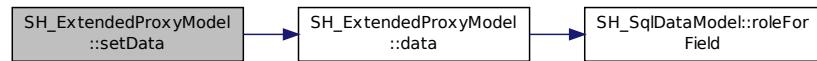
Définition à la ligne 156 du fichier [SH_ExtendedProxyModel.cpp](#).

Références [booleanSet](#), et [data\(\)](#).

```

00157 {
00158     if (!index.isValid())
00159         return false;
00160
00161     if (this->booleanSet.contains(role))
00162     {
00163         QVariant data = (value.toInt() == Qt::Checked) ? QVariant(1) : QVariant(0);
00164         return QSortFilterProxyModel::setData(index, data, role);
00165     }
00166     else
00167     {
00168         return QSortFilterProxyModel::setData(index, value, role);
00169     }
00170 }
00171 }
```

Voici le graphe d'appel pour cette fonction :



4.36.3.21 void SH_ExtendedProxyModel ::setNotNullColumns (QList< int > notNullCols)

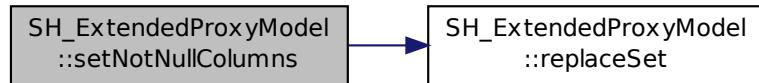
Définition à la ligne 80 du fichier [SH_ExtendedProxyModel.cpp](#).

Références [notNullSet](#), et [replaceSet\(\)](#).

```

00080
00081     if (sourceModel()->inherits("QSqlQueryModel")) {
00082         replaceSet(this->notNullSet, notNullCols);
00083     }
00084 }
```

Voici le graphe d'appel pour cette fonction :



4.36.3.22 void SH_ExtendedProxyModel ::setNullColumns (QList< int > nullCols)

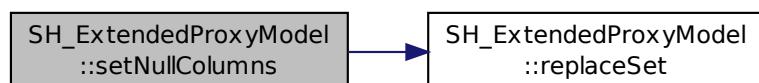
Définition à la ligne 68 du fichier [SH_ExtendedProxyModel.cpp](#).

Références [nullSet](#), et [replaceSet\(\)](#).

```

00068
00069     if (sourceModel()->inherits("QSqlQueryModel")) {
00070         replaceSet(this->nullSet, nullCols);
00071     }
00072 }
```

Voici le graphe d'appel pour cette fonction :



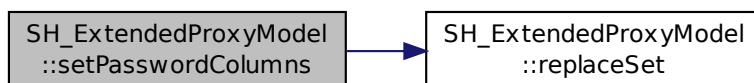
4.36.3.23 void SH_ExtendedProxyModel ::setPasswordColumns (QList< int > passwordCols)

Définition à la ligne 59 du fichier [SH_ExtendedProxyModel.cpp](#).

Références [passwordSet](#), et [replaceSet\(\)](#).

```
00059
00060     replaceSet(this->passwordSet, passwordCols);
00061 }
```

Voici le graphe d'appel pour cette fonction :



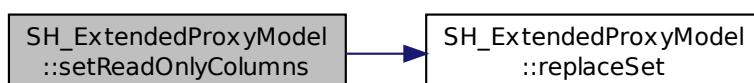
4.36.3.24 void SH_ExtendedProxyModel ::setReadOnlyColumns (QList< int > readonlyCols)

Définition à la ligne 50 du fichier [SH_ExtendedProxyModel.cpp](#).

Références [readonlySet](#), et [replaceSet\(\)](#).

```
00050
00051     replaceSet(this->readonlySet, readonlyCols);
00052 }
```

Voici le graphe d'appel pour cette fonction :



4.36.3.25 void SH_ExtendedProxyModel ::setSortKeyColumn (int column)

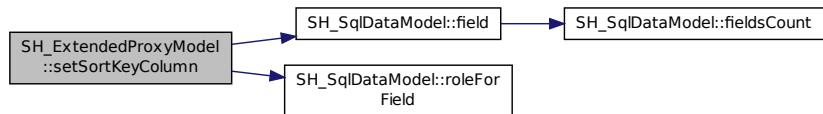
Définition à la ligne 246 du fichier [SH_ExtendedProxyModel.cpp](#).

Références [SH_SqlDataModel ::field\(\)](#), [model](#), [SH_SqlDataModel ::roleForField\(\)](#), [sortChanged\(\)](#), [sortIndex](#), et [SH_SqlDataFields ::sortOrder](#).

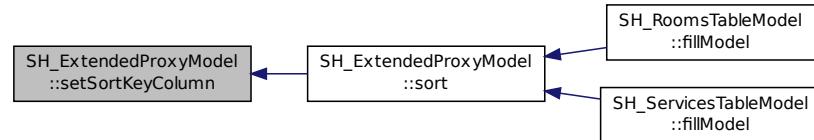
Référencé par [sort\(\)](#).

```
00247 {
00248     this->sortIndex = column;
00249     QSortFilterProxyModel::setSortRole(this->model->roleForField(column));
00250     QSortFilterProxyModel::sort(0, this->model->field(column)->
00251         sortOrder());
00251     emit sortChanged();
00252 }
```

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appels de cette fonction :



4.36.3.26 void SH_ExtendedProxyModel ::sort (int column, Qt ::SortOrder newOrder = Qt ::AscendingOrder)

Définition à la ligne 235 du fichier [SH_ExtendedProxyModel.cpp](#).

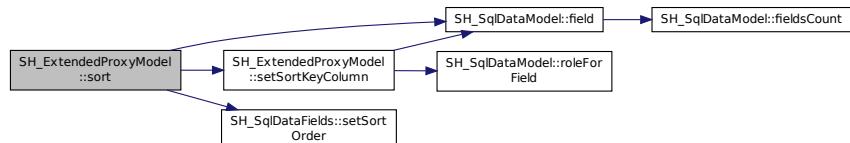
Références [SH_SqlDataModel ::field\(\)](#), [model](#), [setSortKeyColumn\(\)](#), et [SH_SqlDataFields ::setSortOrder\(\)](#).

Référencé par [SH_RoomsTableModel ::fillModel\(\)](#), et [SH_ServicesTableModel ::fillModel\(\)](#).

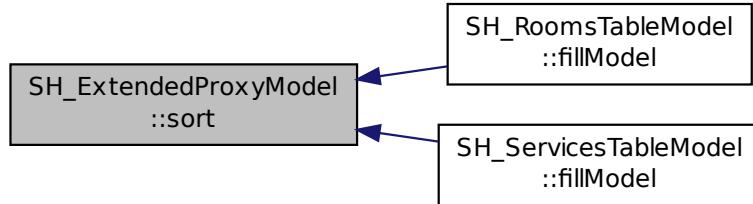
```

00236 {
00237     this->model->field(column)->setSortOrder(newOrder);
00238     SH_ExtendedProxyModel::setSortKeyColumn(column);
00239 }
  
```

Voici le graphe d'appel pour cette fonction :



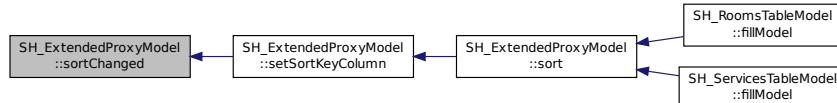
Voici le graphe des appelants de cette fonction :



4.36.3.27 void SH_ExtendedProxyModel ::sortChanged() [signal]

Référencé par [setSortKeyColumn\(\)](#).

Voici le graphe des appelants de cette fonction :



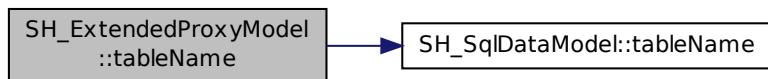
4.36.3.28 const QString SH_ExtendedProxyModel ::tableName() const [inline]

Définition à la ligne 45 du fichier [SH_ExtendedSqlProxyModel.h](#).

Références [model](#), et [SH_SqlDataManager ::tableName\(\)](#).

```
00045 { return this->model->tableName(); }
```

Voici le graphe d'appel pour cette fonction :



4.36.4 Documentation des données membres

4.36.4.1 QList<int> SH_ExtendedProxyModel ::booleanSet [private]

booleanSet

Définition à la ligne 255 du fichier [SH_ExtendedSqlProxyModel.h](#).

Référencé par [data\(\)](#), [flags\(\)](#), [setBooleanColumns\(\)](#), et [setData\(\)](#).

4.36.4.2 QList<int> SH_ExtendedProxyModel : :filters [private]

filters

Définition à la ligne 275 du fichier [SH_ExtendedSqlProxyModel.h](#).

Référencé par [addFilterKeyColumn\(\)](#), [containsFilterKeyColumn\(\)](#), [data\(\)](#), [invalidateFilter\(\)](#), et [removeFilterKeyColumn\(\)](#).

4.36.4.3 SH_SqlDataModel* SH_ExtendedProxyModel : :model [protected]

Définition à la ligne 241 du fichier [SH_ExtendedSqlProxyModel.h](#).

Référencé par [data\(\)](#), [fetch\(\)](#), [field\(\)](#), [fields\(\)](#), [fieldsCount\(\)](#), [SH_BillingsTableModel : :fillModel\(\)](#), [SH_RoomsTableModel : :fillModel\(\)](#), [SH_BookingsTableModel : :fillModel\(\)](#), [isEmpty\(\)](#), [lastError\(\)](#), [roleNames\(\)](#), [setSortKeyColumn\(\)](#), [SH_BillingsTableModel : :SH_BillingsTableModel\(\)](#), [SH_BillsTableModel : :SH_BillsTableModel\(\)](#), [SH_BookingsTableModel : :SH_BookingsTableModel\(\)](#), [SH_ClientsTableModel : :SH_ClientsTableModel\(\)](#), [SH_ExtendedProxyModel\(\)](#), [SH_GroupsTableModel : :SH_GroupsTableModel\(\)](#), [SH_RoomsTableModel : :SH_RoomsTableModel\(\)](#), [SH_ServicesTableModel : :SH_ServicesTableModel\(\)](#), [sort\(\)](#), et [tableName\(\)](#).

4.36.4.4 QList<int> SH_ExtendedProxyModel : :notNullSet [private]

notNullSet

Définition à la ligne 267 du fichier [SH_ExtendedSqlProxyModel.h](#).

Référencé par [filterAcceptsRow\(\)](#), et [setNotNullColumns\(\)](#).

4.36.4.5 QList<int> SH_ExtendedProxyModel : :nullSet [private]

nullSet

Définition à la ligne 271 du fichier [SH_ExtendedSqlProxyModel.h](#).

Référencé par [filterAcceptsRow\(\)](#), et [setNullColumns\(\)](#).

4.36.4.6 QList<int> SH_ExtendedProxyModel : :passwordSet [private]

passwordSet

Définition à la ligne 259 du fichier [SH_ExtendedSqlProxyModel.h](#).

Référencé par [data\(\)](#), et [setPasswordColumns\(\)](#).

4.36.4.7 QList<int> SH_ExtendedProxyModel : :readonlySet [private]

readonlySet

Définition à la ligne 263 du fichier [SH_ExtendedSqlProxyModel.h](#).

Référencé par [flags\(\)](#), et [setReadOnlyColumns\(\)](#).

4.36.4.8 int SH_ExtendedProxyModel : :sortIndex [private]

sortIndex

Définition à la ligne 279 du fichier [SH_ExtendedSqlProxyModel.h](#).

Référencé par [currentSortKeyColumn\(\)](#), [setSortKeyColumn\(\)](#), et [SH_ExtendedProxyModel\(\)](#).

4.36.5 Documentation des propriétés

4.36.5.1 `bool SH_ExtendedProxyModel : :empty [read]`

Définition à la ligne 21 du fichier [SH_ExtendedSqlProxyModel.h](#).

4.36.5.2 `QString SH_ExtendedProxyModel : :fieldsList [read]`

Définition à la ligne 18 du fichier [SH_ExtendedSqlProxyModel.h](#).

4.36.5.3 `QString SH_ExtendedProxyModel : :lastError [read]`

Définition à la ligne 19 du fichier [SH_ExtendedSqlProxyModel.h](#).

4.36.5.4 `int SH_ExtendedProxyModel : :sortKeyColumn [read], [write]`

Définition à la ligne 20 du fichier [SH_ExtendedSqlProxyModel.h](#).

4.36.5.5 `QString SH_ExtendedProxyModel : :table [read]`

Définition à la ligne 17 du fichier [SH_ExtendedSqlProxyModel.h](#).

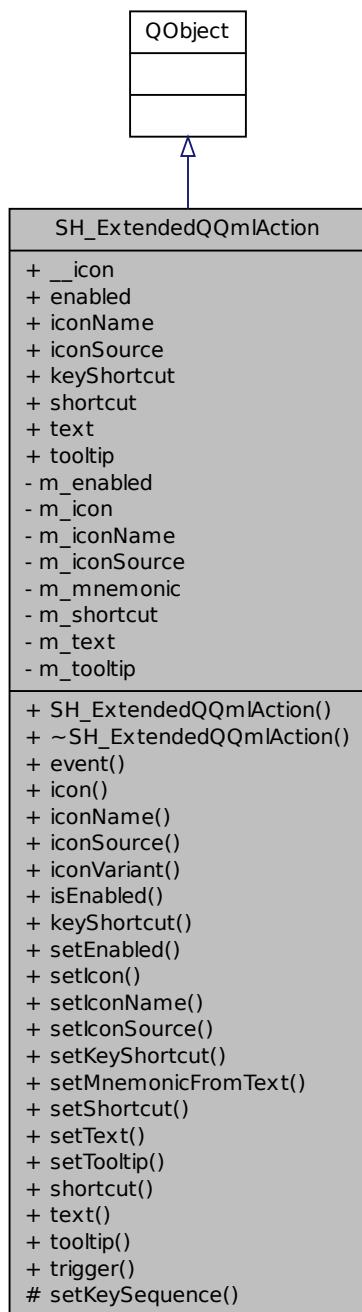
La documentation de cette classe a été générée à partir des fichiers suivants :

- [models/SH_ExtendedSqlProxyModel.h](#)
- [models/SH_ExtendedProxyModel.cpp](#)

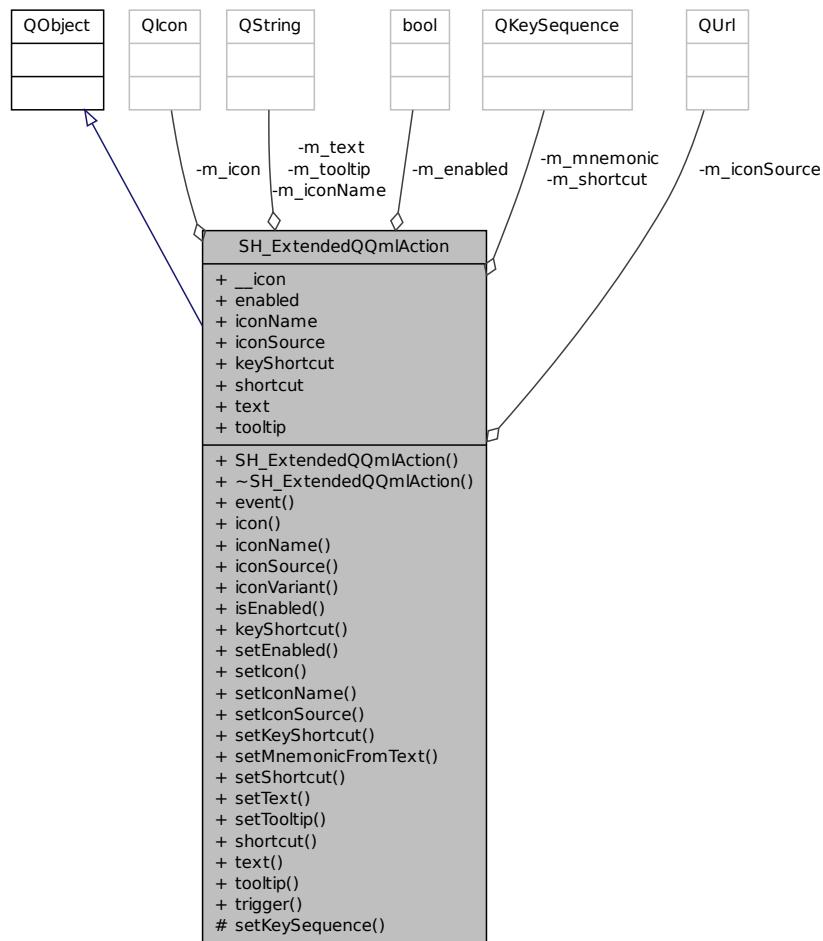
4.37 Référence de la classe SH_ExtendedQQmlAction

```
#include <SH_ExtendedQQmlAction.h>
```

Graphe d'héritage de SH_ExtendedQQmlAction :



Graphe de collaboration de SH_ExtendedQQmlAction :



Connecteurs publics

- void `trigger ()`

Signaux

- void `enabledChanged ()`
- void `iconChanged ()`
- void `iconNameChanged ()`
- void `iconSourceChanged ()`
- void `keyShortcutChanged (Qt::Key keyShortcut)`
- void `shortcutChanged (QString shortcut)`
- void `textChanged ()`
- void `toggled (bool checked)`
- void `tooltipChanged (QString arg)`
- void `triggered ()`

Fonctions membres publiques

- `SH_ExtendedQQmlAction (QObject *parent=0)`
- `~SH_ExtendedQQmlAction ()`
- bool `event (QEvent *e)`
- `QIcon icon () const`

- QString `iconName` () const
- QUrl `iconSource` () const
- QVariant `iconVariant` () const
- bool `isEnabled` () const
- QKeySequence `keyShortcut` () const
- void `setEnabled` (bool e)
- void `setIcon` (QIcon icon)
- void `setIconName` (const QString &iconName)
- void `setIconSource` (const QUrl &iconSource)
- void `setKeyShortcut` (const Qt : :Key &shortcut)
- void `setMnemonicFromText` (const QString &mnemonic)
- void `setShortcut` (const QString &shortcut)
- void `setText` (const QString &text)
- void `setTooltip` (const QString &tooltip)
- QString `shortcut` () const
- QString `text` () const
- QString `tooltip` () const

Fonctions membres protégées

- void `setKeySequence` (const QKeySequence &sequence)

Propriétés

- QVariant `_icon`
- bool `_enabled`
- QString `_iconName`
- QUrl `_iconSource`
- Qt : :Key `_keyShortcut`
- QString `_shortcut`
- QString `_text`
- QString `_tooltip`

Attributs privés

- bool `m_enabled`
 `m_enabled`
- QIcon `m_icon`
 `m_icon`
- QString `m_iconName`
 `m_iconName`
- QUrl `m_iconSource`
 `m_iconSource`
- QKeySequence `m_mnemonic`
 `m_mnemonic`
- QKeySequence `m_shortcut`
 `m_shortcut`
- QString `m_text`
 `m_text`
- QString `m_tooltip`
 `m_tooltip`

4.37.1 Description détaillée

Définition à la ligne 16 du fichier `SH_ExtendedQQmlAction.h`.

4.37.2 Documentation des constructeurs et destructeur

4.37.2.1 `SH_ExtendedQQmlAction` : `SH_ExtendedQQmlAction (QObject * parent = 0) [explicit]`

Définition à la ligne 14 du fichier `SH_ExtendedQQmlAction.cpp`.

```
00015     : QObject (parent)
```

```
00016     , m_enabled(true)
00017 {
00018 }
```

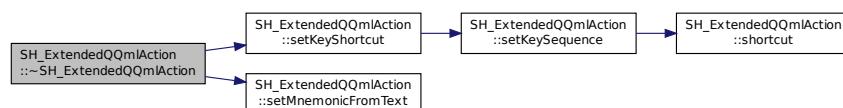
4.37.2.2 SH_ExtendedQQmlAction :~SH_ExtendedQQmlAction ()

Définition à la ligne 25 du fichier [SH_ExtendedQQmlAction.cpp](#).

Références [setKeyShortcut\(\)](#), et [setMnemonicFromText\(\)](#).

```
00026 {
00027     setKeyShortcut(Qt::Key_unknown);
00028     setMnemonicFromText(QString());
00029 }
```

Voici le graphe d'appel pour cette fonction :

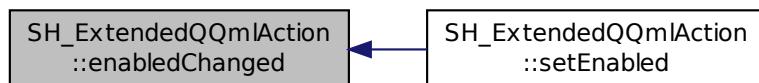


4.37.3 Documentation des fonctions membres

4.37.3.1 void SH_ExtendedQQmlAction ::enabledChanged () [signal]

Référencé par [setEnabled\(\)](#).

Voici le graphe des appels de cette fonction :



4.37.3.2 bool SH_ExtendedQQmlAction ::event (QEvent * e)

Définition à la ligne 228 du fichier [SH_ExtendedQQmlAction.cpp](#).

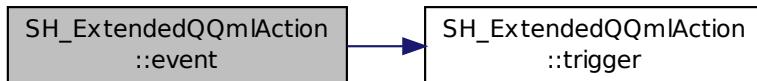
Références [m_enabled](#), [m_mnemonic](#), [m_shortcut](#), et [trigger\(\)](#).

```
00229 {
00230     if (!m_enabled)
00231         return false;
00232
00233     if (e->type() != QEvent::Shortcut)
00234         return false;
00235
00236     QShortcutEvent *se = static_cast<QShortcutEvent *>(e);
00237
00238     Q_ASSERT_X(se->key() == m_shortcut || se->key() == m_mnemonic,
00239                 "QQQuickAction::event",
00240                 "Received shortcut event from incorrect shortcut");
```

```

00241     if (se->isAmbiguous()) {
00242         qWarning("QQQuickAction::event: Ambiguous shortcut overload: %s", se->key().toString(
00243             QKeySequence::NativeText).toLatin1().constData());
00244         return false;
00245     }
00246     trigger();
00247
00248     return true;
00249 }
```

Voici le graphe d'appel pour cette fonction :



4.37.3.3 QIcon SH_ExtendedQQmlAction::icon() const [inline]

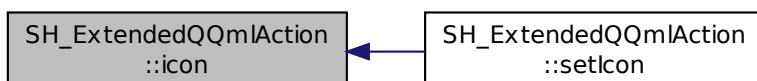
Définition à la ligne 166 du fichier [SH_ExtendedQQmlAction.h](#).

Références [m_icon](#).

Référencé par [setIcon\(\)](#).

```
00166 { return m_icon; }
```

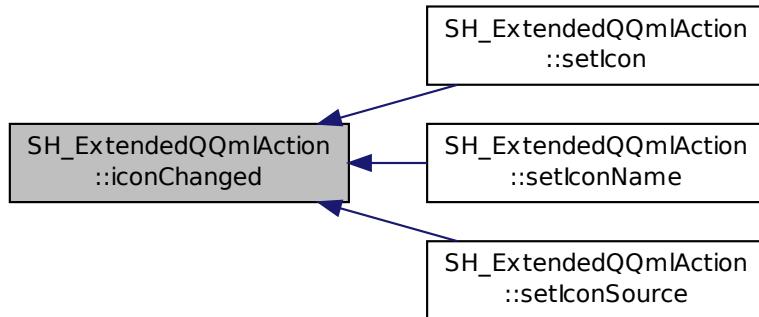
Voici le graphe des appels de cette fonction :



4.37.3.4 void SH_ExtendedQQmlAction::iconChanged() [signal]

Référencé par [setIcon\(\)](#), [setIconName\(\)](#), et [setIconSource\(\)](#).

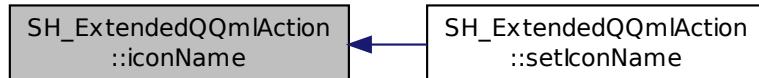
Voici le graphe des appelants de cette fonction :



4.37.3.5 QString SH_ExtendedQQmlAction ::iconName() const

Référencé par [setIconName\(\)](#).

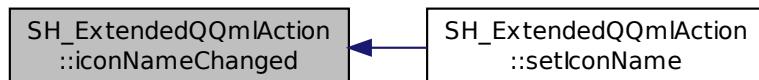
Voici le graphe des appelants de cette fonction :



4.37.3.6 void SH_ExtendedQQmlAction ::iconNameChanged() [signal]

Référencé par [setIconName\(\)](#).

Voici le graphe des appelants de cette fonction :



4.37.3.7 QUrl SH_ExtendedQQmlAction ::iconSource() const [inline]

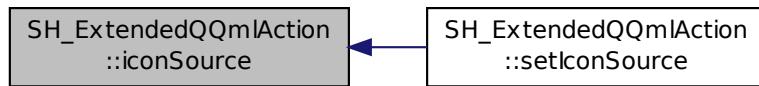
Définition à la ligne [121](#) du fichier [SH_ExtendedQQmlAction.h](#).

Références [m_iconSource](#).

Référencé par [setIconSource\(\)](#).

```
00121 { return m_iconSource; }
```

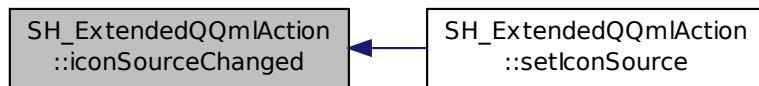
Voici le graphe des appels de cette fonction :



4.37.3.8 void SH_ExtendedQQmlAction ::iconSourceChanged() [signal]

Référencé par [setIconSource\(\)](#).

Voici le graphe des appels de cette fonction :



4.37.3.9 QVariant SH_ExtendedQQmlAction ::iconVariant() const [inline]

Définition à la ligne 173 du fichier [SH_ExtendedQQmlAction.h](#).

Références [m_icon](#).

```
00173 { return QVariant(m_icon); }
```

4.37.3.10 bool SH_ExtendedQQmlAction ::isEnabled() const [inline]

Définition à la ligne 151 du fichier [SH_ExtendedQQmlAction.h](#).

Références [m_enabled](#).

```
00151 { return m_enabled; }
```

4.37.3.11 QKeySequence SH_ExtendedQQmlAction ::keyShortcut() const [inline]

Définition à la ligne 83 du fichier [SH_ExtendedQQmlAction.h](#).

Références [m_shortcut](#).

```
00083 { return m_shortcut; }
```

4.37.3.12 void SH_ExtendedQQmlAction ::keyShortcutChanged (Qt ::Key keyShortcut) [signal]

4.37.3.13 void SH_ExtendedQQmlAction ::setEnabled (bool e)

Définition à la ligne 212 du fichier [SH_ExtendedQQmlAction.cpp](#).

Références [enabledChanged\(\)](#), et [m_enabled](#).

```
00213 {
00214     if (e == m_enabled)
00215         return;
00216     m_enabled = e;
00217     emit enabledChanged();
00218 }
```

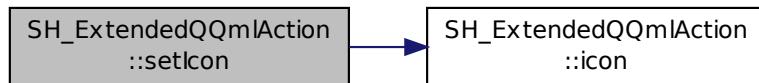
4.37.3.14 void SH_ExtendedQQmlAction ::setIcon (QIcon icon) [inline]

Définition à la ligne 180 du fichier [SH_ExtendedQQmlAction.h](#).

Références [icon\(\)](#), [iconChanged\(\)](#), et [m_icon](#).

```
00180 { m_icon = icon; emit iconChanged(); }
```

Voici le graphe d'appel pour cette fonction :



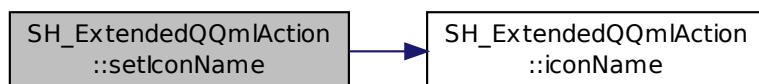
4.37.3.15 void SH_ExtendedQQmlAction ::setIconName (const QString & iconName)

Définition à la ligne 184 du fichier [SH_ExtendedQQmlAction.cpp](#).

Références [iconChanged\(\)](#), [iconName\(\)](#), [iconNameChanged\(\)](#), [m_icon](#), [m_iconName](#), et [m_iconSource](#).

```
00185 {
00186     if (iconName == m_iconName)
00187         return;
00188     m_iconName = iconName;
00189     m_icon = QIcon::fromTheme(m_iconName, QIcon(QQmlFile::urlToLocalFileOrQrc(
00190         m_iconSource)));
00190     emit iconNameChanged();
00191     emit iconChanged();
00192 }
```

Voici le graphe d'appel pour cette fonction :



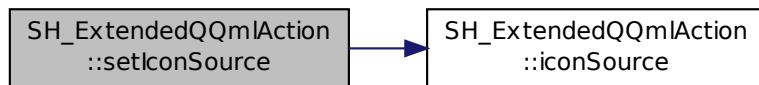
4.37.3.16 void SH_ExtendedQQmlAction ::setIconSource (const QUrl & iconSource)

Définition à la ligne 154 du fichier [SH_ExtendedQQmlAction.cpp](#).

Références [iconChanged\(\)](#), [iconSource\(\)](#), [iconSourceChanged\(\)](#), [m_icon](#), [m_iconName](#), et [m_iconSource](#).

```
00155 {
00156     if (iconSource == m_iconSource)
00157         return;
00158
00159     m_iconSource = iconSource;
00160     if (m_iconName.isEmpty() || m_icon.isNull()) {
00161         QString urlString = QQmlFile::urlToLocalFileOrQrc(iconSource);
00162         m_icon = QIcon(urlString);
00163     }
00164     emit iconChanged();
00165 }
00166 emit iconSourceChanged();
00167 }
```

Voici le graphe d'appel pour cette fonction :



4.37.3.17 void SH_ExtendedQQmlAction ::setKeySequence (const QKeySequence & sequence) [protected]

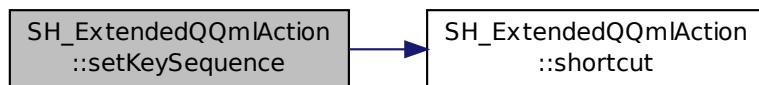
Définition à la ligne 78 du fichier [SH_ExtendedQQmlAction.cpp](#).

Références [m_shortcut](#), [shortcut\(\)](#), et [shortcutChanged\(\)](#).

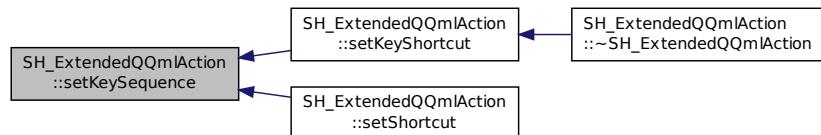
Référencé par [setKeyShortcut\(\)](#), et [setShortcut\(\)](#).

```
00078 {
00079     if (sequence == m_shortcut)
00080         return;
00081
00082     /*if (!m_shortcut.isEmpty())
00083         QGuiApplicationPrivate::instance() -> shortcutMap.removeShortcut(0, this, m_shortcut);
00084 */
00085     m_shortcut = sequence;
00086
00087     if (!m_shortcut.isEmpty()) {
00088         Qt::ShortcutContext context = Qt::WindowShortcut;
00089         /*QGuiApplicationPrivate::instance() -> shortcutMap.addShortcut(this, m_shortcut, context,
00090         qShortcutContextMatcher);*/
00091         emit shortcutChanged(shortcut());
00092 }
```

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :



4.37.3.18 void SH_ExtendedQQmlAction ::setKeyShortcut (const Qt::Key & shortcut)

Définition à la ligne 99 du fichier [SH_ExtendedQQmlAction.cpp](#).

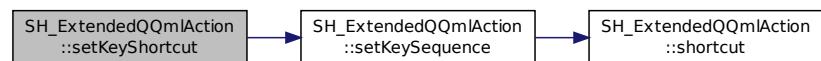
Références [setKeySequence\(\)](#).

Référencé par [~SH_ExtendedQQmlAction\(\)](#).

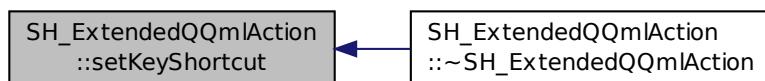
```

00100 {
00101     setKeySequence(QKeySequence(shortcut));
00102 }
  
```

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :



4.37.3.19 void SH_ExtendedQQmlAction ::setMnemonicFromText (const QString & mnemonic)

Définition à la ligne 132 du fichier [SH_ExtendedQQmlAction.cpp](#).

Références [m_mnemonic](#).

Référencé par [setText\(\)](#), et [~SH_ExtendedQQmlAction\(\)](#).

```

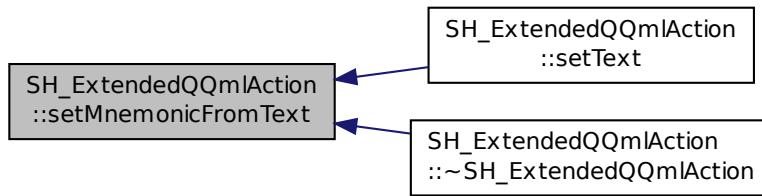
00133 {
00134     QKeySequence sequence = QKeySequence::mnemonic(text);
00135     if (m_mnemonic == sequence)
00136         return;
00137     /*if (!m_mnemonic.isEmpty())
  
```

```

00139     QGuiApplicationPrivate::instance()->shortcutMap.removeShortcut(0, this, m_mnemonic);
00140     */
00141     m_mnemonic = sequence;
00142
00143     if (!m_mnemonic.isEmpty()) {
00144         Qt::ShortcutContext context = Qt::WindowShortcut;
00145         /*QGuiApplicationPrivate::instance()->shortcutMap.addShortcut(this, m_mnemonic, context,
00146         qShortcutContextMatcher);*/
00147     }

```

Voici le graphe des appelleurs de cette fonction :



4.37.3.20 void SH_ExtendedQQmlAction ::setShortcut (const QString & shortcut)

Définition à la ligne 119 du fichier [SH_ExtendedQQmlAction.cpp](#).

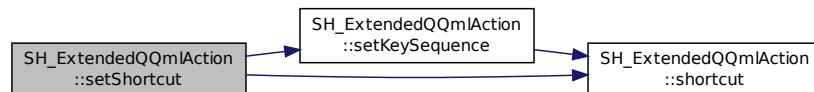
Références [setKeySequence\(\)](#), et [shortcut\(\)](#).

```

00120 {
00121     if(shortcut() == arg)
00122         return;
00123     setKeySequence(QKeySequence::fromString(arg));
00125 }

```

Voici le graphe d'appel pour cette fonction :



4.37.3.21 void SH_ExtendedQQmlAction ::setText (const QString & text)

Définition à la ligne 36 du fichier [SH_ExtendedQQmlAction.cpp](#).

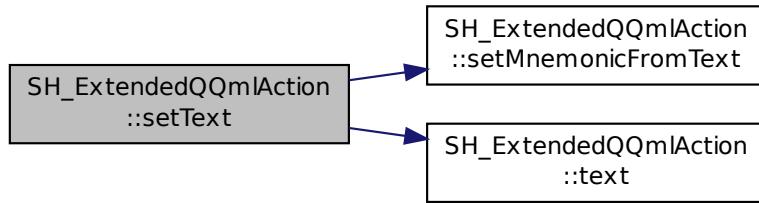
Références [m_text](#), [setMnemonicFromText\(\)](#), [text\(\)](#), et [textChanged\(\)](#).

```

00037 {
00038     if (text == m_text)
00039         return;
00040     m_text = text;
00041     setMnemonicFromText(m_text);
00042     emit textChanged();
00043 }

```

Voici le graphe d'appel pour cette fonction :



4.37.3.22 void SH_ExtendedQQmlAction ::setTooltip (const QString & tooltip)

Définition à la ligne 199 du fichier [SH_ExtendedQQmlAction.cpp](#).

Références [m_tooltip](#), et [tooltipChanged\(\)](#).

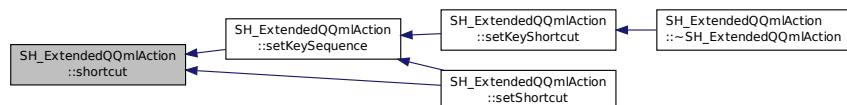
```

00200 {
00201     if (m_tooltip != arg) {
00202         m_tooltip = arg;
00203         emit tooltipChanged(arg);
00204     }
00205 }
```

4.37.3.23 QString SH_ExtendedQQmlAction ::shortcut () const

Référencé par [setKeySequence\(\)](#), et [setShortcut\(\)](#).

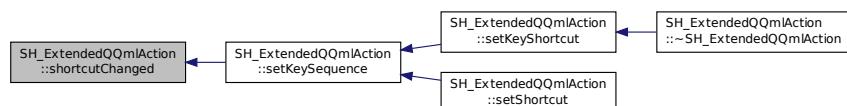
Voici le graphe des appelants de cette fonction :



4.37.3.24 void SH_ExtendedQQmlAction ::shortcutChanged (QString shortcut) [signal]

Référencé par [setKeySequence\(\)](#).

Voici le graphe des appelants de cette fonction :



4.37.3.25 QString SH_ExtendedQQmlAction ::text () const [inline]

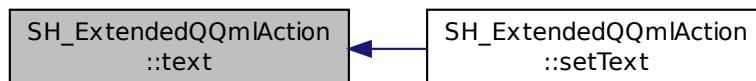
Définition à la ligne 53 du fichier [SH_ExtendedQQmlAction.h](#).

Références [m_text](#).

Référencé par [setText\(\)](#).

```
00053 { return m_text; }
```

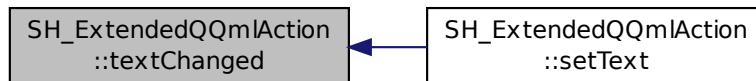
Voici le graphe des appelants de cette fonction :



4.37.3.26 void SH_ExtendedQQmlAction ::TextChanged () [signal]

Référencé par [setText\(\)](#).

Voici le graphe des appelants de cette fonction :



4.37.3.27 void SH_ExtendedQQmlAction ::toggled (bool checked) [signal]

4.37.3.28 QString SH_ExtendedQQmlAction ::tooltip () const [inline]

Définition à la ligne 136 du fichier [SH_ExtendedQQmlAction.h](#).

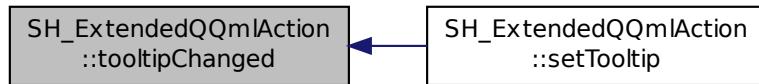
Références [m_tooltip](#).

```
00136 { return m_tooltip; }
```

4.37.3.29 void SH_ExtendedQQmlAction ::tooltipChanged (QString arg) [signal]

Référencé par [setTooltip\(\)](#).

Voici le graphe des appelants de cette fonction :



4.37.3.30 void SH_ExtendedQQmlAction ::trigger() [slot]

Définition à la ligne 256 du fichier [SH_ExtendedQQmlAction.cpp](#).

Références [m_enabled](#), et [triggered\(\)](#).

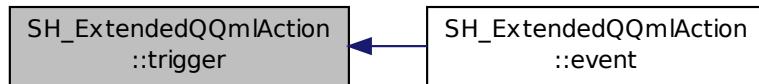
Référencé par [event\(\)](#).

```

00257 {
00258     if (!m_enabled)
00259         return;
00260     emit triggered();
00261 }

```

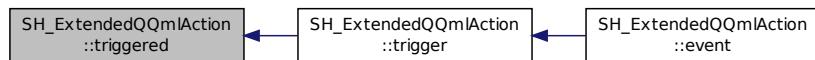
Voici le graphe des appelants de cette fonction :



4.37.3.31 void SH_ExtendedQQmlAction ::triggered() [signal]

Référencé par [trigger\(\)](#).

Voici le graphe des appelants de cette fonction :



4.37.4 Documentation des données membres

4.37.4.1 bool SH_ExtendedQQmlAction ::m_enabled [private]

[m_enabled](#)

Définition à la ligne 297 du fichier [SH_ExtendedQQmlAction.h](#).

Référencé par [event\(\)](#), [isEnabled\(\)](#), [setEnabled\(\)](#), et [trigger\(\)](#).

4.37.4.2 QIcon SH_ExtendedQQmlAction ::m_icon [private]

m_icon

Définition à la ligne 293 du fichier [SH_ExtendedQQmlAction.h](#).

Référencé par [icon\(\)](#), [iconVariant\(\)](#), [setIcon\(\)](#), [setIconName\(\)](#), et [setIconSource\(\)](#).

4.37.4.3 QString SH_ExtendedQQmlAction ::m_iconName [private]

m_iconName

Définition à la ligne 289 du fichier [SH_ExtendedQQmlAction.h](#).

Référencé par [setIconName\(\)](#), et [setIconSource\(\)](#).

4.37.4.4 QUrl SH_ExtendedQQmlAction ::m_iconSource [private]

m_iconSource

Définition à la ligne 285 du fichier [SH_ExtendedQQmlAction.h](#).

Référencé par [iconSource\(\)](#), [setIconName\(\)](#), et [setIconSource\(\)](#).

4.37.4.5 QKeySequence SH_ExtendedQQmlAction ::m_mnemonic [private]

m_mnemonic

Définition à la ligne 305 du fichier [SH_ExtendedQQmlAction.h](#).

Référencé par [event\(\)](#), et [setMnemonicFromText\(\)](#).

4.37.4.6 QKeySequence SH_ExtendedQQmlAction ::m_shortcut [private]

m_shortcut

Définition à la ligne 301 du fichier [SH_ExtendedQQmlAction.h](#).

Référencé par [event\(\)](#), [keyShortcut\(\)](#), et [setKeySequence\(\)](#).

4.37.4.7 QString SH_ExtendedQQmlAction ::m_text [private]

m_text

Définition à la ligne 281 du fichier [SH_ExtendedQQmlAction.h](#).

Référencé par [setText\(\)](#), et [text\(\)](#).

4.37.4.8 QString SH_ExtendedQQmlAction ::m_tooltip [private]

m_tooltip

Définition à la ligne 309 du fichier [SH_ExtendedQQmlAction.h](#).

Référencé par [setTooltip\(\)](#), et [tooltip\(\)](#).

4.37.5 Documentation des propriétés

4.37.5.1 QVariant SH_ExtendedQQmlAction ::_icon [read]

Définition à la ligne 23 du fichier [SH_ExtendedQQmlAction.h](#).

4.37.5.2 bool SH_ExtendedQQmlAction ::enabled [read], [write]

Définition à la ligne 25 du fichier [SH_ExtendedQQmlAction.h](#).

4.37.5.3 QString SH_ExtendedQQmlAction ::iconName [read], [write]

Définition à la ligne 22 du fichier [SH_ExtendedQQmlAction.h](#).

4.37.5.4 QUrl SH_ExtendedQQmlAction ::iconSource [read], [write]

Définition à la ligne 21 du fichier [SH_ExtendedQQmlAction.h](#).

4.37.5.5 Qt ::Key SH_ExtendedQQmlAction ::keyShortcut [write]

Définition à la ligne 29 du fichier [SH_ExtendedQQmlAction.h](#).

4.37.5.6 QString SH_ExtendedQQmlAction ::shortcut [read], [write]

Définition à la ligne 28 du fichier [SH_ExtendedQQmlAction.h](#).

4.37.5.7 QString SH_ExtendedQQmlAction ::text [read], [write]

Définition à la ligne 20 du fichier [SH_ExtendedQQmlAction.h](#).

4.37.5.8 QString SH_ExtendedQQmlAction ::tooltip [read], [write]

Définition à la ligne 24 du fichier [SH_ExtendedQQmlAction.h](#).

La documentation de cette classe a été générée à partir des fichiers suivants :

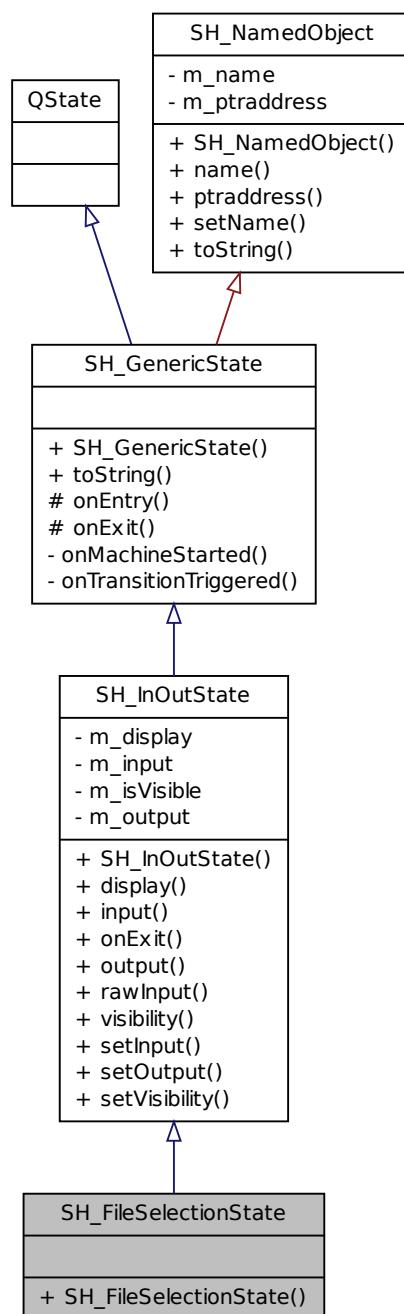
- views/[SH_ExtendedQQmlAction.h](#)
- views/[SH_ExtendedQQmlAction.cpp](#)

4.38 Référence de la classe SH_FileSelectionState

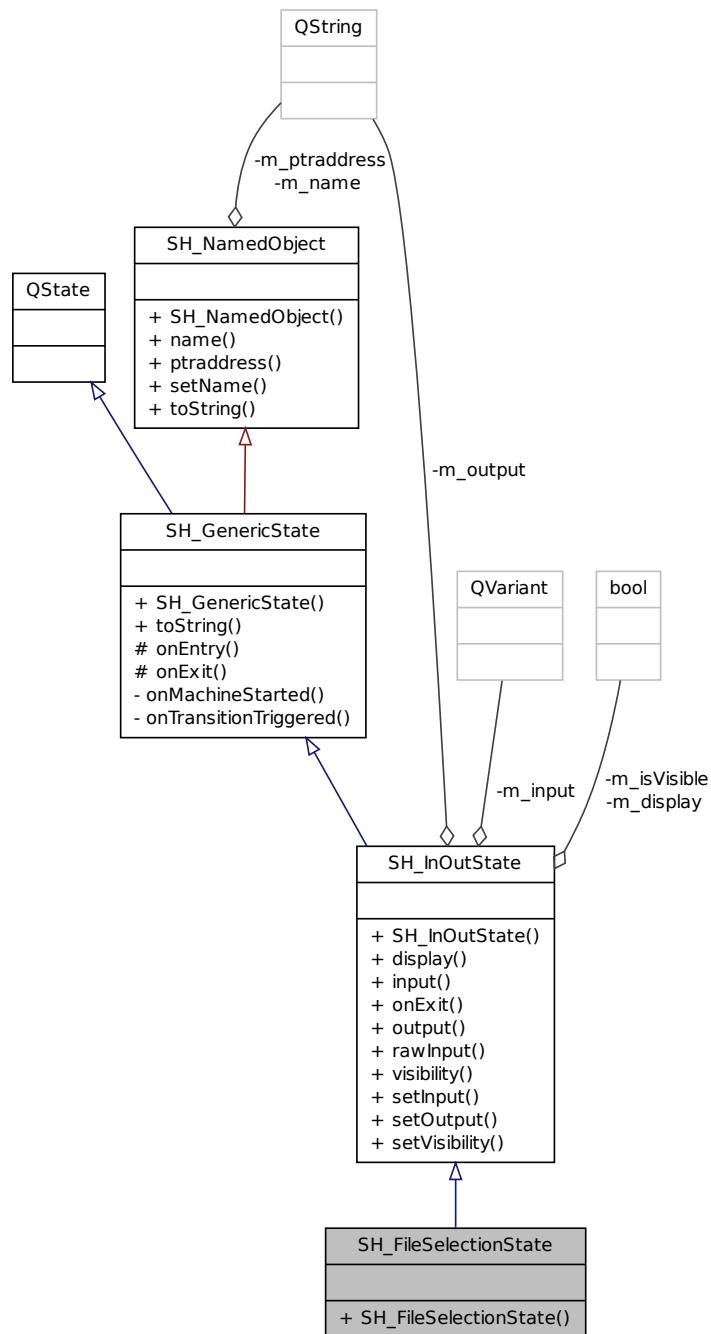
The [SH_FileSelectionState](#) class.

```
#include <SH_FileSelectionState.h>
```

Graphe d'héritage de SH_FileSelectionState :



Graphe de collaboration de SH_FileSelectionState :



Connecteurs publics

- `virtual void setInput (const QVariant &input)`
- `virtual void setOutput (const QString &output)`
- `virtual void setVisibility (bool isVisible)`

Signaux

- void `next ()`
- void `resendInput (QVariant input)`
- void `sendOutput (QVariant output)`

Fonctions membres publiques

- `SH_FileSelectionState (QString output, QString name, QState *parent=0)`
- void `display (bool canDisplay)`
- virtual QVariant `input () const`
- void `onExit (QEvent *event)`
- virtual QString `output () const`
- virtual QVariant `rawInput () const`
- QString `toString ()`
- bool `visibility ()`

Fonctions membres protégées

- void `onEntry (QEvent *event)`

4.38.1 Description détaillée

The `SH_FileSelectionState` class.

Définition à la ligne 7 du fichier `SH_FileSelectionState.h`.

4.38.2 Documentation des constructeurs et destructeur

4.38.2.1 `SH_FileSelectionState : :SH_FileSelectionState (QString output, QString name, QState * parent = 0)`

Définition à la ligne 8 du fichier `SH_FileSelectionState.cpp`.

```
00008
00009     SH_InOutState(output, name, parent)
00010 {
00011 }
00012 }
```

:

4.38.3 Documentation des fonctions membres

4.38.3.1 `void SH_InOutState : :display (bool canDisplay) [inherited]`

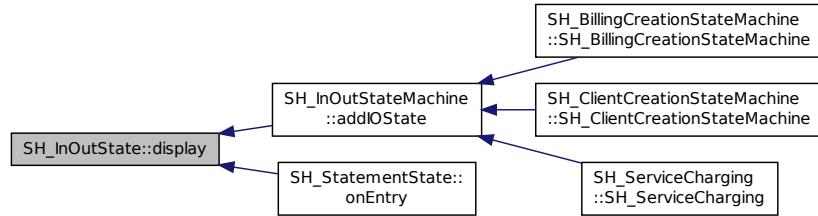
Définition à la ligne 95 du fichier `SH_IOState.cpp`.

Références `SH_InOutState : :m_display`, `SH_InOutState : :m_isVisible`, `SH_InOutState : :m_output`, et `SH_InOutState : :sendOutput()`.

Référencé par `SH_InOutStateMachine : :addIOState()`, et `SH_StatementState : :onEntry()`.

```
00096 {
00097     m_display=canDisplay;
00098     if(m_display && !m_output.isEmpty() && m_isVisible) {
00099         qDebug() << "resalut !" << QVariant(m_output);
00100         emit sendOutput(QVariant(m_output));
00101     }
00102 }
```

Voici le graphe des appels de cette fonction :



4.38.3.2 QVariant SH_InOutState : :input() const [virtual], [inherited]

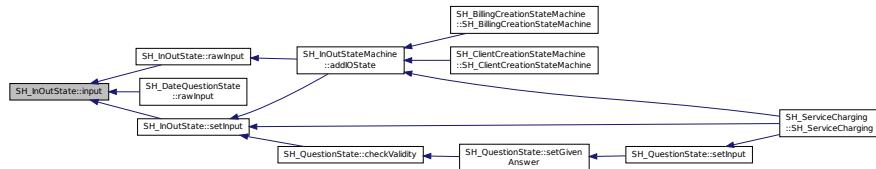
Définition à la ligne 20 du fichier [SH_IOState.cpp](#).

Références SH_InOutState ::m_input.

Référencé par `SH_InOutState::rawInput()`, `SH_DateQuestionState::rawInput()`, et `SH_InOutState::setInput()`.

```
00021 {  
00022     return m_input;  
00023 }
```

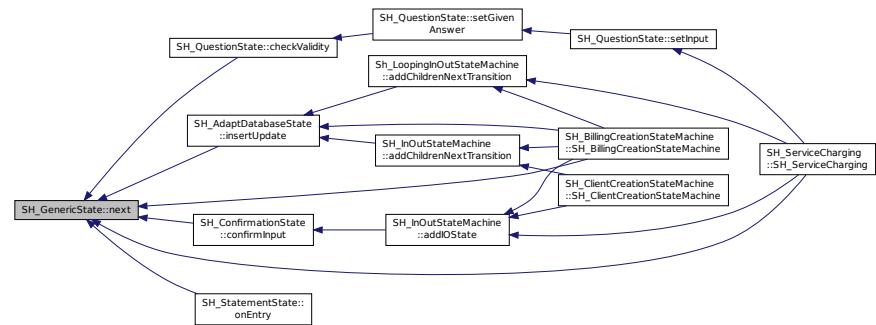
Voici le graphe des appels de cette fonction :



4.38.3.3 void SH_GenericState ::next() [signal], [inherited]

Référencé par [SH_QuestionState](#) : `:checkValidity()`, [SH_ConfirmationState](#) : `:confirmInput()`, [SH_AdaptDatabase-State](#) : `:insertUpdate()`, [SH_StatementState](#) : `:onEntry()`, [SH_BillingCreationStateMachine](#) : `:SH_BillingCreation-StateMachine()`, et [SH_ServiceCharging](#) : `:SH_ServiceCharging()`.

Voici le graphe des appels de cette fonction :



4.38.3.4 void SH_GenericState ::onEntry (QEvent * event) [protected], [inherited]

Définition à la ligne 62 du fichier [SH_GenericDebugableState.cpp](#).

Références [SH_NamedObject ::name\(\)](#).

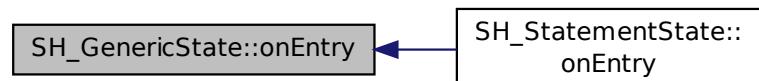
Référencé par [SH_StatementState ::onEntry\(\)](#).

```
00063 {
00064     Q_UNUSED(event);
00065     qDebug() << "Machine: " << machine()->objectName() << " entered " << name();
00066 }
```

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appels de cette fonction :



4.38.3.5 void SH_InOutState ::onExit (QEvent * event) [inherited]

Définition à la ligne 110 du fichier [SH_IOSState.cpp](#).

Références [SH_InOutState ::m_input](#), [SH_InOutState ::m_isVisible](#), [SH_GenericState ::onExit\(\)](#), et [SH_InOutState ::resendInput\(\)](#).

```
00111 {
00112     SH_GenericState::onExit(event);
00113     if(m_isVisible) {
00114         emit resendInput(m_input);
00115     }
00116 }
```

Voici le graphe d'appel pour cette fonction :



4.38.3.6 QString SH_InOutState ::output() const [virtual], [inherited]

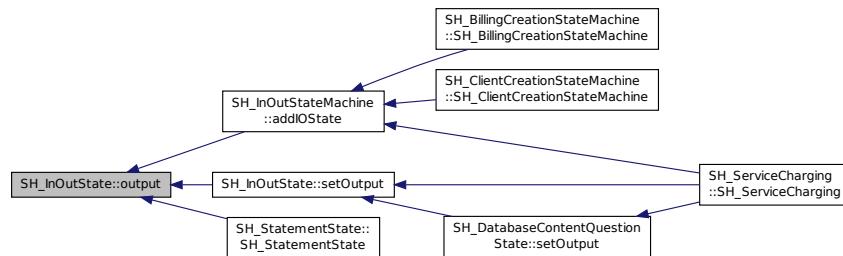
Définition à la ligne 56 du fichier [SH_IOState.cpp](#).

Références [SH_InOutState ::m_output](#).

Référencé par [SH_InOutStateMachine ::addIOState\(\)](#), [SH_InOutState ::setOutput\(\)](#), et [SH_StatementState ::SH_StatementState\(\)](#).

```
00057 {
00058     return m_output;
00059 }
```

Voici le graphe des appels de cette fonction :



4.38.3.7 QVariant SH_InOutState ::rawInput() const [virtual], [inherited]

Réimplémentée dans [SH_DateQuestionState](#), et [SH_DatabaseContentQuestionState](#).

Définition à la ligne 30 du fichier [SH_IOState.cpp](#).

Références [SH_InOutState ::input\(\)](#).

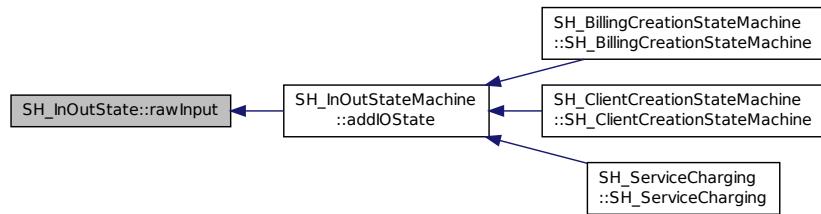
Référencé par [SH_InOutStateMachine ::addIOState\(\)](#).

```
00031 {
00032     return input();
00033 }
```

Voici le graphe d'appel pour cette fonction :



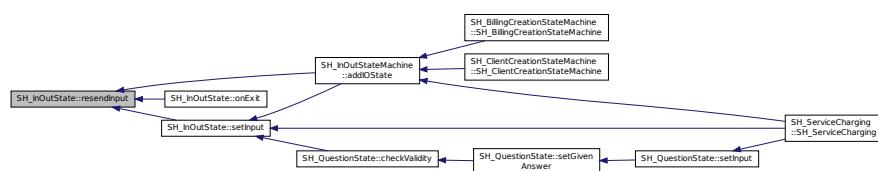
Voici le graphe des appels de cette fonction :



4.38.3.8 void SH_InOutState : :resendInput (QVariant *input*) [signal], [inherited]

Référencé par `SH_InOutStateMachine` ::`addIOState()`, `SH_InOutState` ::`onExit()`, et `SH_InOutState` ::`setInput()`.

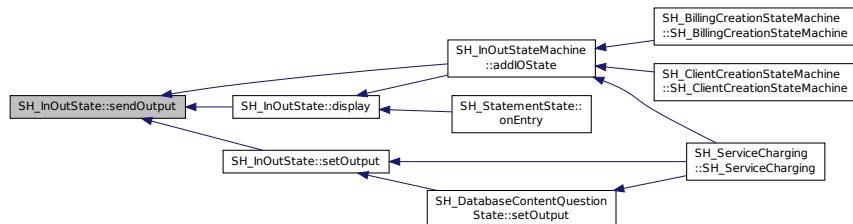
Voici le graphe des appels de cette fonction :



4.38.3.9 void SH_InOutState : :sendOutput (QVariant output) [signal], [inherited]

Référencé par `SH_InOutStateMachine` : `:addIOState()`, `SH_InOutState` : `:display()`, et `SH_InOutState` : `:setOutput()`.

Voici le graphe des appels de cette fonction :



4.38.3.10 void SH_InOutState ::setInput (const QVariant & input) [virtual], [slot], [inherited]

Réimplémentée dans SH QuestionState, et SH StatementState.

Définition à la ligne 41 du fichier SH_IOState.cpp.

Références SH_InOutState ::input(), SH_InOutState ::m_input, SH_InOutState ::m_isVisible, et SH_InOutState ::resendInput().

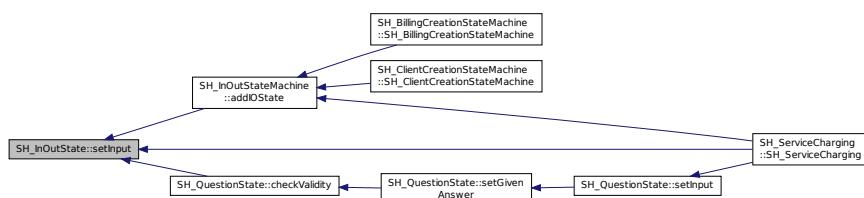
Référencé par [SH_InOutStateMachine](#) : `:addIOState()`, [SH_QuestionState](#) : `:checkValidity()`, et [SH_ServiceCharging](#) : `:SH_ServiceCharging()`.

```
00042 {
00043     qDebug() << "new input " << input.toString();
00044     m_input = input;
00045     if(m_isVisible) {
00046         emit resendInput(m_input);
00047     }
00048 }
```

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :



4.38.3.11 void SH_InOutState ::setOutput(const QString & output) [virtual], [slot], [inherited]

Réimplémentée dans [SH_DatabaseContentQuestionState](#).

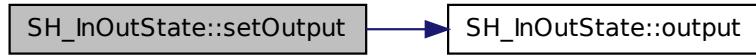
Définition à la ligne 68 du fichier [SH_IOState.cpp](#).

Références [SH_InOutState](#) : `:m_isVisible`, [SH_InOutState](#) : `:m_output`, [SH_InOutState](#) : `:output()`, et [SH_InOutState](#) : `:sendOutput()`.

Référencé par [SH_DatabaseContentQuestionState](#) : `:setOutput()`, et [SH_ServiceCharging](#) : `:SH_ServiceCharging()`.

```
00069 {
00070     m_output = output;
00071     if(m_isVisible) {
00072         emit sendOutput(QVariant(m_output));
00073     }
00074 }
```

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :



4.38.3.12 void SH_InOutState ::setVisibility (bool isVisible) [virtual], [slot], [inherited]

Définition à la ligne 81 du fichier [SH_IOState.cpp](#).

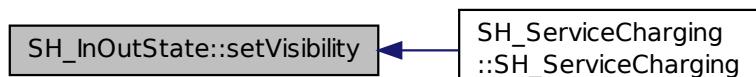
Références [SH_InOutState ::m_isVisible](#).

Référencé par [SH_ServiceCharging ::SH_ServiceCharging\(\)](#).

```

00082 {
00083     m_isVisible = isVisible;
00084 }
```

Voici le graphe des appelants de cette fonction :



4.38.3.13 QString SH_GenericState ::toString () [virtual], [inherited]

Réimplémentée à partir de [SH_NamedObject](#).

Définition à la ligne 21 du fichier [SH_GenericDebugableState.cpp](#).

Références [SH_NamedObject ::toString\(\)](#), et [SH_InOutStateMachine ::toString\(\)](#).

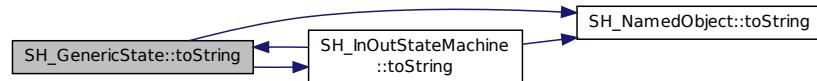
Référencé par [SH_InOutStateMachine ::addChildsNextTransition\(\)](#), [SH_DateQuestionState ::rawInput\(\)](#), et [SH_InOutStateMachine ::toString\(\)](#).

```

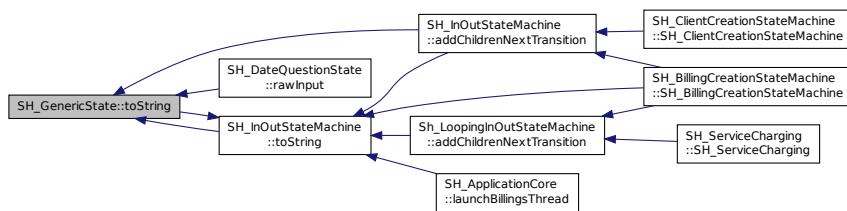
00022 {
00023     QStateMachine* machine = this->machine();
00024     SH_InOutStateMachine* mach = qobject_cast<
00025         SH_InOutStateMachine *>(machine);
00026     if(mach) {
00027         return SH_NamedObject::toString() + " [in "+mach->
00028             toString()+" ] ";
00029     } else {
00030         return SH_NamedObject::toString();
00031     }
00030 }

```

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appels de cette fonction :



4.38.3.14 bool SH_InOutState::visibility() [inherited]

Définition à la ligne 91 du fichier [SH_IOState.cpp](#).

Références [SH_InOutState::m_isVisible](#).

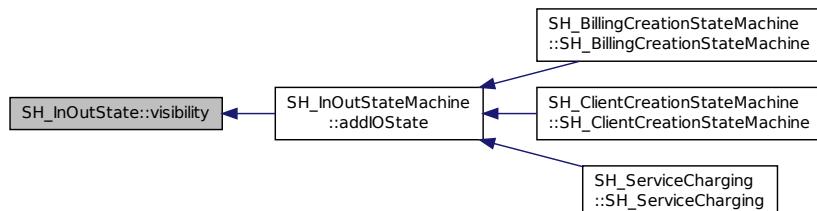
Référencé par [SH_InOutStateMachine::addIOState\(\)](#).

```

00091 {
00092     return m_isVisible;
00093 }

```

Voici le graphe des appels de cette fonction :



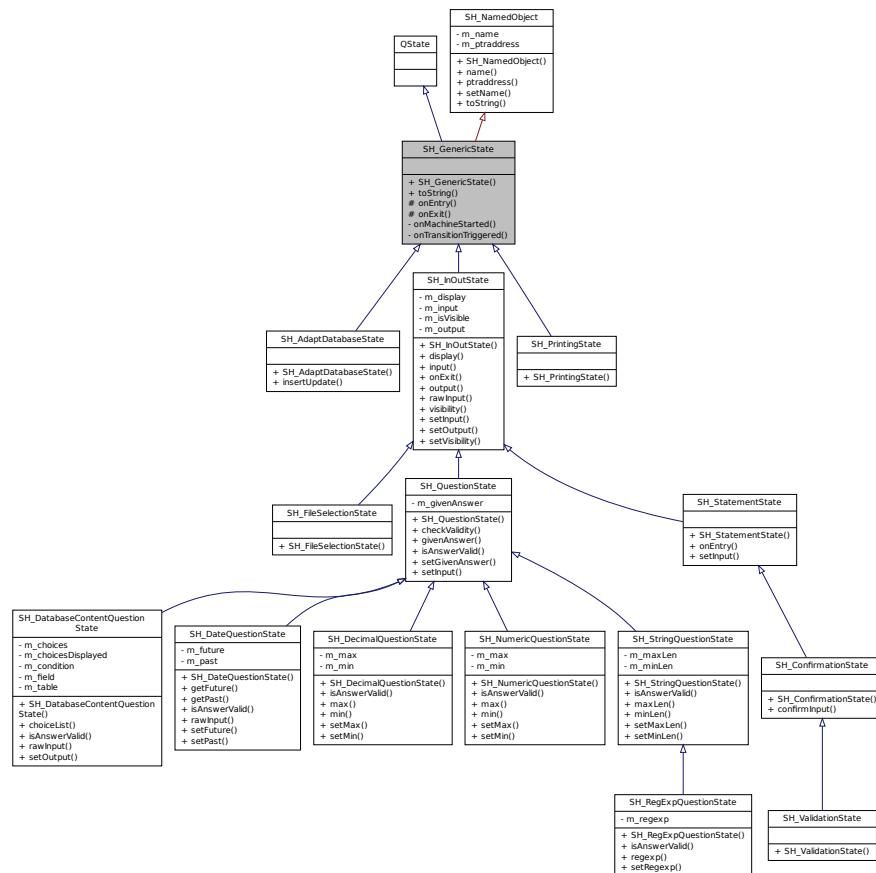
La documentation de cette classe a été générée à partir des fichiers suivants :

- logic/[SH_FileSelectionState.h](#)
- logic/[SH_FileSelectionState.cpp](#)

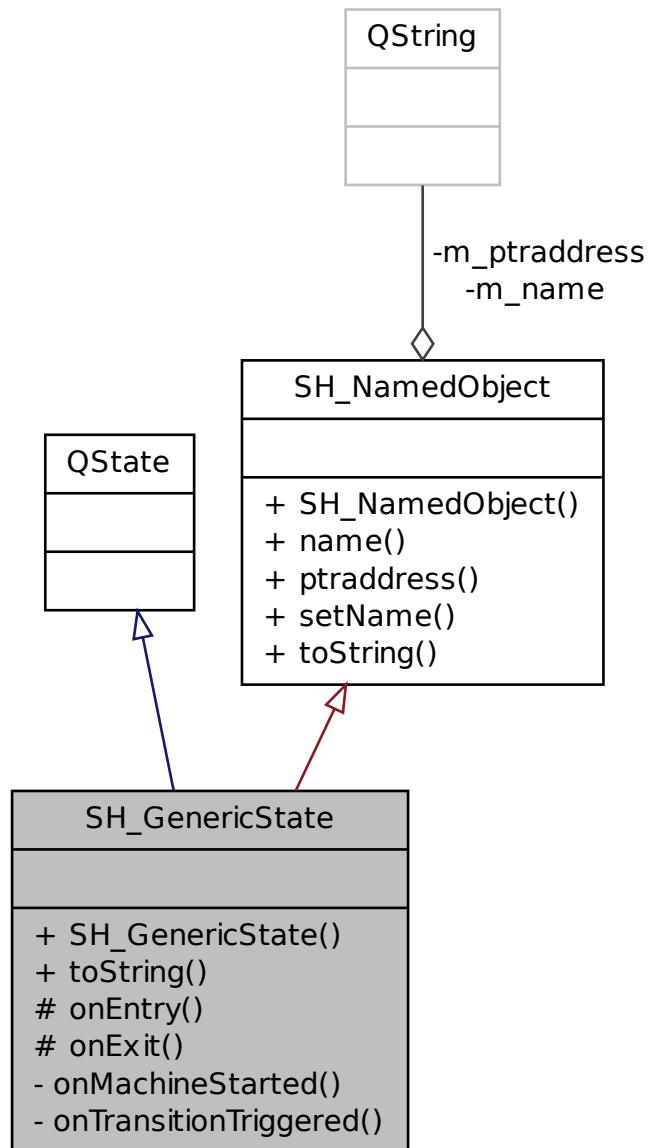
4.39 Référence de la classe SH_GenericState

```
#include <SH_GenericDebugableState.h>
```

Graphe d'héritage de SH_GenericState :



Graphe de collaboration de SH_GenericState :



Signaux

- void `next ()`

Fonctions membres publiques

- `SH_GenericState (QString name="", QState *parent=0)`
- `QString toString ()`

Fonctions membres protégées

- void [onEntry](#) (QEvent *event)
- void [onExit](#) (QEvent *event)

Connecteurs privés

- void [onMachineStarted](#) ()
- void [onTransitionTriggered](#) ()

Fonctions membres privées

- virtual QString [name](#) () const
- QString [ptraddress](#) () const
- virtual void [setName](#) (const QString &[name](#))

4.39.1 Description détaillée

Définition à la ligne 12 du fichier [SH_GenericDebugableState.h](#).

4.39.2 Documentation des constructeurs et destructeur

4.39.2.1 SH_GenericState : :SH_GenericState (QString *name* = " ", QState * *parent* = 0)

Paramètres

<i>name</i>	
<i>parent</i>	

Définition à la ligne 10 du fichier [SH_GenericDebugableState.cpp](#).

```
00010      :
00011      QState(parent), SH_NamedObject(name)
00012  {
00013 }
```

4.39.3 Documentation des fonctions membres

4.39.3.1 QString SH_NamedObject : :name () const [virtual], [inherited]

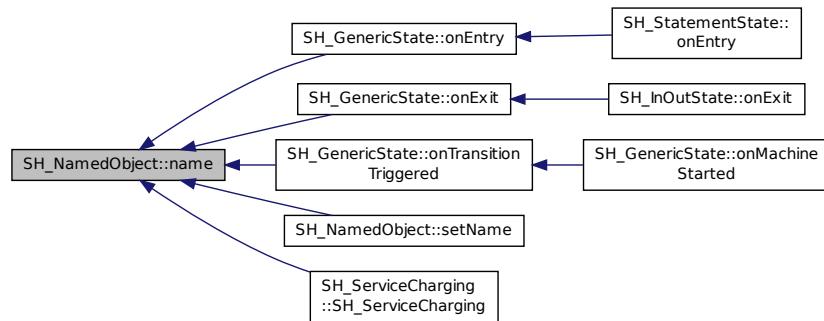
Définition à la ligne 32 du fichier [SH_NamedObject.cpp](#).

Références [SH_NamedObject : :m_name](#).

Référencé par [onEntry\(\)](#), [onExit\(\)](#), [onTransitionTriggered\(\)](#), [SH_NamedObject : :setName\(\)](#), et [SH_Service-Charging : :SH_ServiceCharging\(\)](#).

```
00033 {
00034     return m\_name;
00035 }
```

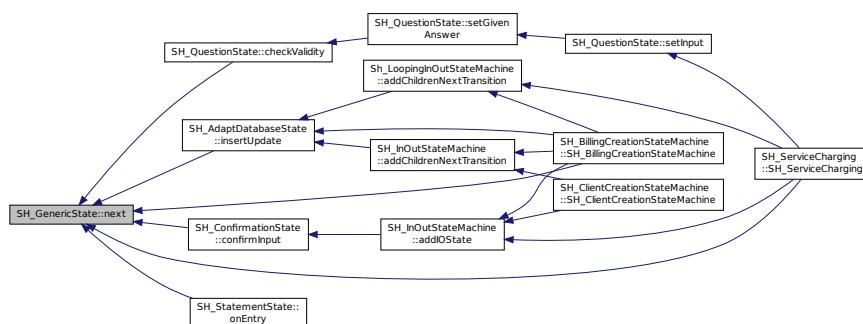
Voici le graphe des appelants de cette fonction :



4.39.3.2 void SH_GenericState ::next() [signal]

Référencé par [SH_QuestionState ::checkValidity\(\)](#), [SH_ConfirmationState ::confirmInput\(\)](#), [SH_AdaptDatabaseState ::insertUpdate\(\)](#), [SH_StatementState ::onEntry\(\)](#), [SH_BillingCreationStateMachine ::SH_BillingCreationStateMachine\(\)](#), et [SH_ServiceCharging ::SH_ServiceCharging\(\)](#).

Voici le graphe des appelants de cette fonction :



4.39.3.3 void SH_GenericState ::onEntry (QEvent * event) [protected]

Définition à la ligne 62 du fichier [SH_GenericDebugableState.cpp](#).

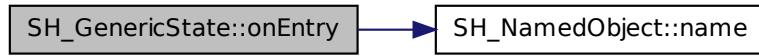
Références [SH_NamedObject ::name\(\)](#).

Référencé par [SH_StatementState ::onEntry\(\)](#).

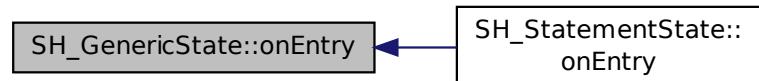
```

00063 {
00064     Q_UNUSED(event);
00065     qDebug() << "Machine: " << machine()->objectName() << " entered " << name();
00066 }
  
```

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :



4.39.3.4 void SH_GenericState ::onExit (QEvent * event) [protected]

Définition à la ligne 73 du fichier [SH_GenericDebugableState.cpp](#).

Références [SH_NamedObject ::name\(\)](#).

Référencé par [SH_InOutState ::onExit\(\)](#).

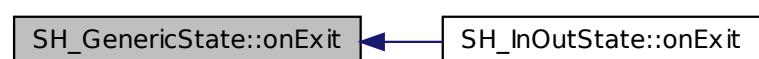
```

00074 {
00075     Q_UNUSED(event);
00076     qDebug() << "Machine: " << machine()->objectName() << " exited " << name();
00077 }
  
```

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :



4.39.3.5 void SH_GenericState ::onMachineStarted() [private], [slot]

Définition à la ligne 83 du fichier [SH_GenericDebugableState.cpp](#).

Références [onTransitionTriggered\(\)](#).

```
00084 {
00085     foreach (QAbstractTransition* tr, transitions())
00086         connect(tr, SIGNAL(triggered()), this, SLOT(onTransitionTriggered()));
00087 }
```

Voici le graphe d'appel pour cette fonction :



4.39.3.6 void SH_GenericState ::onTransitionTriggered() [private], [slot]

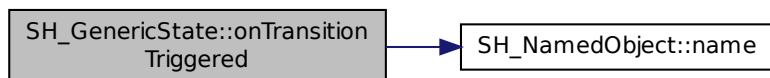
Définition à la ligne 36 du fichier [SH_GenericDebugableState.cpp](#).

Références [SH_NamedObject ::name\(\)](#).

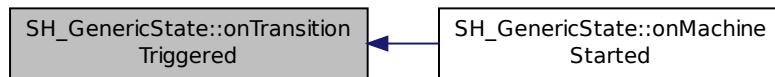
Référencé par [onMachineStarted\(\)](#).

```
00037 {
00038     QAbstractTransition* tr = qobject_cast<QAbstractTransition*>(sender());
00039     if (tr == 0) return;
00040
00041     SH_GenericState* sourceState = qobject_cast<SH_GenericState*>(tr->
00042         sourceState());
00043     SH_GenericState* targetState = qobject_cast<SH_GenericState*>(tr->
00044         targetState());
00045
00046     QString log;
00047     QTextStream logStream(&log);
00048     logStream << machine()>objectName() << " transition from ";
00049     if (sourceState) logStream << sourceState->name();
00050     else logStream << tr->sourceState();
00051     logStream << " to ";
00052     if (targetState) logStream << targetState->name();
00053     else logStream << tr->targetState();
00054     logStream.flush();
00055     qDebug() << "Machine: " << log;
00056 }
```

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appels de cette fonction :



4.39.3.7 QString SH_NamedObject ::ptraddress () const [inherited]

Définition à la ligne 54 du fichier [SH_NamedObject.cpp](#).

Références [SH_NamedObject ::m_ptraddress](#).

```

00055 {
00056     return m_ptraddress;
00057 }

```

4.39.3.8 void SH_NamedObject ::setName (const QString & name) [virtual], [inherited]

Définition à la ligne 43 du fichier [SH_NamedObject.cpp](#).

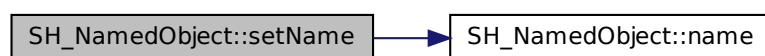
Références [SH_NamedObject ::m_name](#), et [SH_NamedObject ::name\(\)](#).

```

00044 {
00045     m_name = name;
00046 }

```

Voici le graphe d'appel pour cette fonction :



4.39.3.9 QString SH_GenericState ::toString () [virtual]

Réimplémentée à partir de [SH_NamedObject](#).

Définition à la ligne 21 du fichier [SH_GenericDebugableState.cpp](#).

Références [SH_NamedObject ::toString\(\)](#), et [SH_InOutStateMachine ::toString\(\)](#).

Référencé par [SH_InOutStateMachine ::addChildNextTransition\(\)](#), [SH_DateQuestionState ::rawInput\(\)](#), et [SH_InOutStateMachine ::toString\(\)](#).

```

00022 {
00023     QStateMachine* machine = this->machine();
00024     SH_InOutStateMachine* mach = qobject_cast<
00025         SH_InOutStateMachine *>(machine);
00026     if(mach) {
00027         return SH_NamedObject::toString() + " [in "+mach-

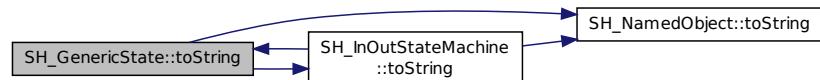
```

```

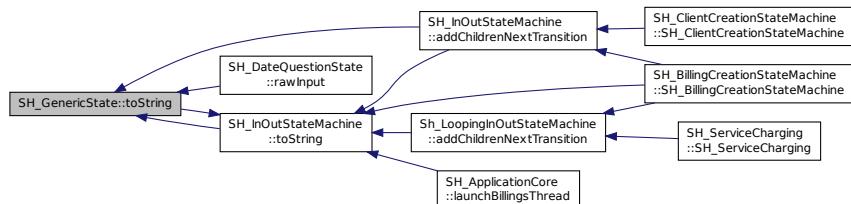
    toString()+" ] ";
00027     } else {
00028         return SH_NamedObject::toString();
00029     }
00030 }

```

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appels de cette fonction :



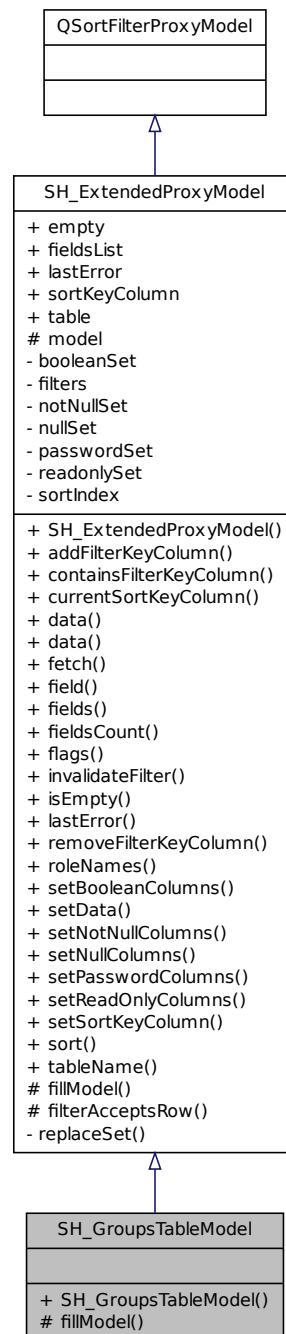
La documentation de cette classe a été générée à partir des fichiers suivants :

- logic/[SH_GenericDebugableState.h](#)
- logic/[SH_GenericDebugableState.cpp](#)

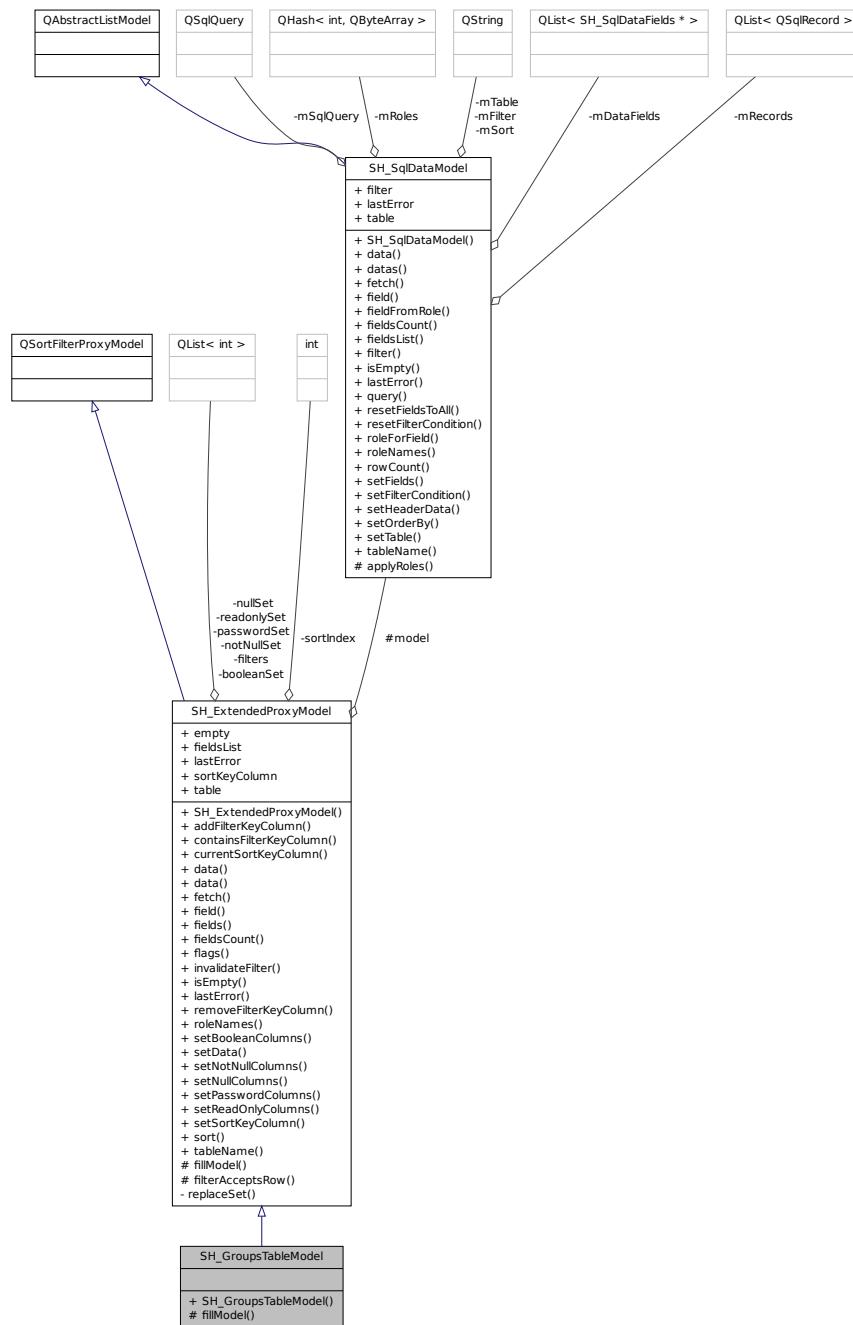
4.40 Référence de la classe SH_GroupsTableModel

```
#include <SH_GroupsTableModel.h>
```

Graphe d'héritage de SH_GroupsTableModel :



Graphe de collaboration de SH_GroupsTableModel :



Signaux

- void `sortChanged ()`

Fonctions membres publiques

- `SH_GroupsTableModel (QObject *parent=0)`
- `Q_INVOKABLE void addFilterKeyColumn (int column)`
- `Q_INVOKABLE bool containsFilterKeyColumn (int column)`
- `const int currentSortKeyColumn () const`

- Q_INVOKABLE QVariant **data** (int row, int column) const
- Q_INVOKABLE QVariant **data** (const QModelIndex &index, int role=Qt::DisplayRole) const
- Q_INVOKABLE bool **fetch** (QString **tableName**="", QString **filter**="", QString **sort**="", QStringList **fields**=QStringList())
- Q_INVOKABLE SH_SqlDataFields * **field** (int i) const
- const QString **fields** () const
- Q_INVOKABLE int **fieldsCount** () const
- Q_INVOKABLE Qt::ItemFlags **flags** (const QModelIndex &index) const
- void **invalidateFilter** ()
- const bool **isEmpty** () const
- const QString **lastError** () const
- Q_INVOKABLE void **removeFilterKeyColumn** (int column)
- virtual Q_INVOKABLE QHash<int, QByteArray > **roleNames** () const
- void **setBooleanColumns** (QList<int> **boolCols**)
- Q_INVOKABLE bool **setData** (const QModelIndex &index, const QVariant &value, int role=Qt::EditRole)
- void **setNotNullColumns** (QList<int> **notNullCols**)
- void **setNullColumns** (QList<int> **nullCols**)
- void **setPasswordColumns** (QList<int> **passwordCols**)
- void **setReadOnlyColumns** (QList<int> **readonlyCols**)
- void **setSortKeyColumn** (int column)
- Q_INVOKABLE void **sort** (int column, Qt::SortOrder newOrder=Qt::AscendingOrder)
- const QString **tableName** () const

Fonctions membres protégées

- void **fillModel** ()
- bool **filterAcceptsRow** (int source_row, const QModelIndex &source_parent) const

Attributs protégés

- SH_SqlDataModel * **model**

Propriétés

- bool **empty**
- QString **fieldsList**
- QString **lastError**
- int **sortKeyColumn**
- QString **table**

4.40.1 Description détaillée

Définition à la ligne 14 du fichier [SH_GroupsTableModel.h](#).

4.40.2 Documentation des constructeurs et destructeur

4.40.2.1 SH_GroupsTableModel : :SH_GroupsTableModel (QObject * parent = 0)

Paramètres

<i>parent</i>	
---------------	--

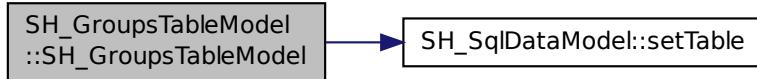
Définition à la ligne 9 du fichier [SH_GroupsTableModel.cpp](#).

Références [SH_ExtendedProxyModel](#) : :model, et [SH_SqlDataModel](#) : :setTable().

```

00009
00010     SH_ExtendedProxyModel (parent)
00011 {
00012     SH_ExtendedProxyModel::model->setTable ("GROUPS");
00013 }
```

Voici le graphe d'appel pour cette fonction :



4.40.3 Documentation des fonctions membres

4.40.3.1 void SH_ExtendedProxyModel : :addFilterKeyColumn (int column) [inherited]

Définition à la ligne 259 du fichier [SH_ExtendedProxyModel.cpp](#).

Références [SH_ExtendedProxyModel : :filters](#).

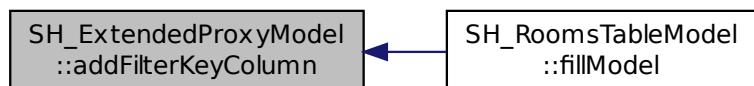
Référencé par [SH_RoomsTableModel : :fillModel\(\)](#).

```

00260 {
00261     this->filters.append\(column\);
00262 }

```

Voici le graphe des appels de cette fonction :



4.40.3.2 bool SH_ExtendedProxyModel : :containsFilterKeyColumn (int column) [inherited]

Définition à la ligne 225 du fichier [SH_ExtendedProxyModel.cpp](#).

Références [SH_ExtendedProxyModel : :filters](#).

```

00226 {
00227     return this->filters.contains\(column\);
00228 }

```

4.40.3.3 const int SH_ExtendedProxyModel : :currentSortKeyColumn () const [inline], [inherited]

Définition à la ligne 38 du fichier [SH_ExtendedSqlProxyModel.h](#).

Références [SH_ExtendedProxyModel : :sortIndex](#).

```
00038 {return this->sortIndex;}
```

4.40.3.4 QVariant SH_ExtendedProxyModel::data(int row, int column) const [inherited]

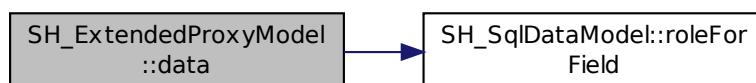
Définition à la ligne 269 du fichier [SH_ExtendedProxyModel.cpp](#).

Références [SH_ExtendedProxyModel::model](#), et [SH_SqlDataModel::roleForField\(\)](#).

Référencé par [SH_ExtendedProxyModel::setData\(\)](#).

```
00270 {
00271     QModelIndex modelIndex = this->index(row, 0);
00272     return this->data(modelIndex, this->model->roleForField(column));
00273 }
```

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appels de cette fonction :



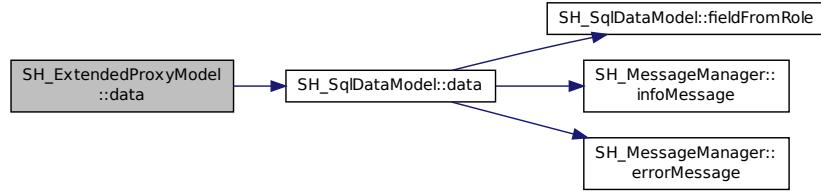
4.40.3.5 QVariant SH_ExtendedProxyModel::data(const QModelIndex & index, int role = Qt::DisplayRole) const [inherited]

Définition à la ligne 127 du fichier [SH_ExtendedProxyModel.cpp](#).

Références [SH_ExtendedProxyModel::booleanSet](#), [SH_SqlDataModel::data\(\)](#), [SH_ExtendedProxyModel::filters](#), [SH_ExtendedProxyModel::model](#), et [SH_ExtendedProxyModel::passwordSet](#).

```
00128 {
00129     if (index.isValid())
00130     {
00131         if (this->booleanSet.contains(role))
00132         {
00133             return index.data(Qt::EditRole).toBool() ? QVariant(Qt::Checked) : QVariant(Qt::Unchecked);
00134         }
00135         else if (this->passwordSet.contains(role))
00136         {
00137             return QVariant("****");
00138         }
00139         else if (!this->filters.contains(role))
00140         {
00141             QModelIndex source_index = QSortFilterProxyModel::mapToSource(index);
00142             if (source_index.isValid())
00143                 return this->model->data(source_index, role);
00144             }
00145         }
00146     }
00147     return QVariant();
00148 }
```

Voici le graphe d'appel pour cette fonction :



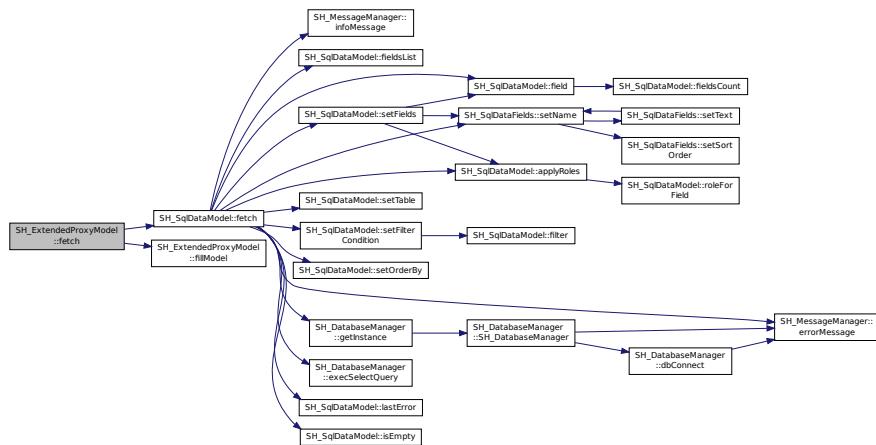
4.40.3.6 bool SH_ExtendedProxyModel::fetch (QString tableName = "", QString filter = "", QString sort = "", QStringList fields = QStringList()) [inherited]

Définition à la ligne 280 du fichier [SH_ExtendedProxyModel.cpp](#).

Références `SH_SqlDataModel` : `:fetch()`, `SH_ExtendedProxyModel` : `:fillModel()`, et `SH_ExtendedProxyModel` : `:model`.

```
00281 {  
00282     bool fetched = this->model->fetch(tableName, filter, sort,  
00283         fields);  
00284     if (fetched)  
00285     {  
00286         this->fillModel();  
00287     }  
00288     this->setSourceModel(this->model);  
00289     return fetched;  
00290 }
```

Voici le graphe d'appel pour cette fonction :



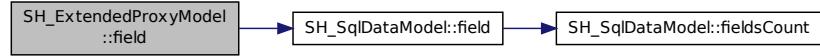
4.40.3.7 **Q_INVOKABLE SH_SqlDataFields* SH_ExtendedProxyModel ::field (int i) const [inline], [inherited]**

Définition à la ligne 82 du fichier SH_ExtendedSqlProxyModel.h.

Références `SH_SqlDataModel` : `:field()`, et `SH_ExtendedProxyModel` : `:model`.

```
00082 { return this->model->field(i); }
```

Voici le graphe d'appel pour cette fonction :



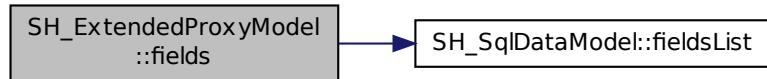
4.40.3.8 const QString SH_ExtendedProxyModel ::fields() const [inline], [inherited]

Définition à la ligne 52 du fichier [SH_ExtendedSqlProxyModel.h](#).

Références [SH_SqlDataModel ::fieldsList\(\)](#), et [SH_ExtendedProxyModel ::model](#).

```
00052 { if(this->model->fieldsList().isEmpty()){ return "*"; } else { return this->model->fieldsList().join(", "); } }
```

Voici le graphe d'appel pour cette fonction :



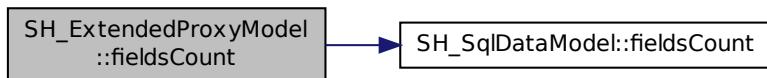
4.40.3.9 Q_INVOKABLE int SH_ExtendedProxyModel ::fieldsCount() const [inline], [inherited]

Définition à la ligne 89 du fichier [SH_ExtendedSqlProxyModel.h](#).

Références [SH_SqlDataModel ::fieldsCount\(\)](#), et [SH_ExtendedProxyModel ::model](#).

```
00089 { return this->model->fieldsCount(); }
```

Voici le graphe d'appel pour cette fonction :



4.40.3.10 void SH_GroupsTableModel : :fillModel () [protected], [virtual]

Implémente [SH_ExtendedProxyModel](#).

Définition à la ligne 20 du fichier [SH_GroupsTableModel.cpp](#).

```
00021 {
00022 }
```

4.40.3.11 bool SH_ExtendedProxyModel : :filterAcceptsRow (int source_row, const QModelIndex & source_parent) const [protected], [inherited]

Définition à la ligne 92 du fichier [SH_ExtendedProxyModel.cpp](#).

Références [SH_ExtendedProxyModel : :notNullSet](#), et [SH_ExtendedProxyModel : :nullSet](#).

```
00093 {
00094     Q_UNUSED(source_parent);
00095
00096     if (!this->notNullSet.isEmpty())
00097     {
00098         QSqlQueryModel *m = static_cast<QSqlQueryModel *>(sourceModel());
00099         foreach(int column, this->notNullSet)
00100         {
00101             if (m->record(source_row).isNull(column))
00102             {
00103                 return false;
00104             }
00105         }
00106     }
00107
00108     if (!this->nullSet.isEmpty())
00109     {
00110         QSqlQueryModel *m = static_cast<QSqlQueryModel *>(sourceModel());
00111         foreach(int column, this->nullSet)
00112         {
00113             if (!m->record(source_row).isNull(column))
00114             {
00115                 return false;
00116             }
00117         }
00118     }
00119     return true;
00120 }
```

4.40.3.12 Qt : :itemFlags SH_ExtendedProxyModel : :flags (const QModelIndex & index) const [inherited]

Définition à la ligne 179 du fichier [SH_ExtendedProxyModel.cpp](#).

Références [SH_ExtendedProxyModel : :booleanSet](#), et [SH_ExtendedProxyModel : :readonlySet](#).

```
00180 {
00181     if (!index.isValid())
00182     {
00183         return Qt::ItemIsEnabled;
00184     }
00185     if (!this->booleanSet.isEmpty())
00186     {
00187         return Qt::ItemIsUserCheckable | Qt::ItemIsSelectable | Qt::ItemIsEnabled;
00188     }
00189     else if (!this->readonlySet.isEmpty())
00190     {
00191         return Qt::ItemIsSelectable;
00192     }
00193     else
00194     {
00195         return QSortFilterProxyModel::flags(index);
00196     }
00197 }
00198 }
```

4.40.3.13 void SH_ExtendedProxyModel : :invalidateFilter () [inherited]

Définition à la ligne 205 du fichier [SH_ExtendedProxyModel.cpp](#).

Références [SH_ExtendedProxyModel : :filters](#).

```
00206 {
00207     this->filters.clear();
00208 }
```

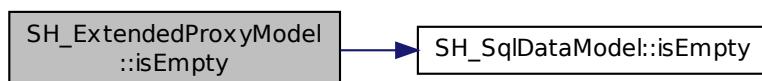
4.40.3.14 const bool SH_ExtendedProxyModel : :isEmpty () const [inline], [inherited]

Définition à la ligne 66 du fichier [SH_ExtendedSqlProxyModel.h](#).

Références [SH_SqlDataModel : :isEmpty\(\)](#), et [SH_ExtendedProxyModel : :model](#).

```
00066 { return this->model->isEmpty(); }
```

Voici le graphe d'appel pour cette fonction :



4.40.3.15 const QString SH_ExtendedProxyModel : :lastError () const [inline], [inherited]

Définition à la ligne 59 du fichier [SH_ExtendedSqlProxyModel.h](#).

Références [SH_SqlDataModel : :lastError](#), et [SH_ExtendedProxyModel : :model](#).

```
00059 { return this->model->lastError(); }
```

4.40.3.16 void SH_ExtendedProxyModel : :removeFilterKeyColumn (int column) [inherited]

Définition à la ligne 215 du fichier [SH_ExtendedProxyModel.cpp](#).

Références [SH_ExtendedProxyModel : :filters](#).

```
00216 {
00217     this->filters.removeAt(this->filters.indexOf(column));
00218 }
```

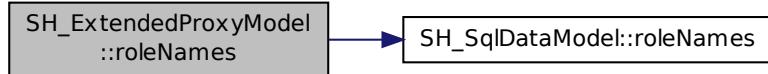
4.40.3.17 virtual Q_INVOKABLE QHash<int, QByteArray> SH_ExtendedProxyModel : :roleNames () const [inline], [virtual], [inherited]

Définition à la ligne 165 du fichier [SH_ExtendedSqlProxyModel.h](#).

Références [SH_ExtendedProxyModel : :model](#), et [SH_SqlDataModel : :roleNames\(\)](#).

```
00165 { return this->model->roleNames(); }
```

Voici le graphe d'appel pour cette fonction :



4.40.3.18 void SH_ExtendedProxyModel ::setBooleanColumns (QList< int > boolCols) [inherited]

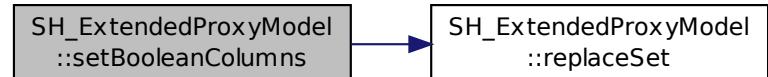
Définition à la ligne 41 du fichier [SH_ExtendedProxyModel.cpp](#).

Références [SH_ExtendedProxyModel ::booleanSet](#), et [SH_ExtendedProxyModel ::replaceSet\(\)](#).

```

00041
00042     replaceSet(this->booleanSet, boolCols);
00043 }
```

Voici le graphe d'appel pour cette fonction :



4.40.3.19 bool SH_ExtendedProxyModel ::setData (const QModelIndex & index, const QVariant & value, int role = Qt :: EditRole) [inherited]

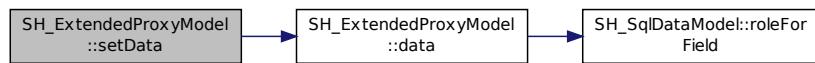
Définition à la ligne 156 du fichier [SH_ExtendedProxyModel.cpp](#).

Références [SH_ExtendedProxyModel ::booleanSet](#), et [SH_ExtendedProxyModel ::data\(\)](#).

```

00157 {
00158     if (!index.isValid())
00159         return false;
00160
00161     if (this->booleanSet.contains(role))
00162     {
00163         QVariant data = (value.toInt() == Qt :: Checked) ? QVariant(1) : QVariant(0);
00164         return QSortFilterProxyModel :: setData(index, data, role);
00165     }
00166     else
00167     {
00168         return QSortFilterProxyModel :: setData(index, value, role);
00169     }
00170
00171 }
```

Voici le graphe d'appel pour cette fonction :



4.40.3.20 void SH_ExtendedProxyModel ::setNotNullColumns (QList< int > notNullCols) [inherited]

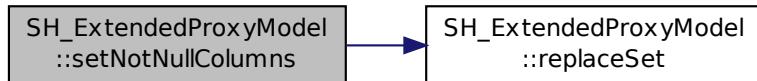
Définition à la ligne 80 du fichier [SH_ExtendedProxyModel.cpp](#).

Références [SH_ExtendedProxyModel ::notNullSet](#), et [SH_ExtendedProxyModel ::replaceSet\(\)](#).

```

00080
00081     if (sourceModel()->inherits("QSqlQueryModel")) {
00082         replaceSet(this->notNullSet, notNullCols);
00083     }
00084 }
```

Voici le graphe d'appel pour cette fonction :



4.40.3.21 void SH_ExtendedProxyModel ::setNullColumns (QList< int > nullCols) [inherited]

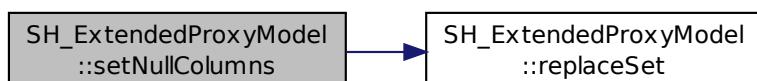
Définition à la ligne 68 du fichier [SH_ExtendedProxyModel.cpp](#).

Références [SH_ExtendedProxyModel ::nullSet](#), et [SH_ExtendedProxyModel ::replaceSet\(\)](#).

```

00068
00069     if (sourceModel()->inherits("QSqlQueryModel")) {
00070         replaceSet(this->nullSet, nullCols);
00071     }
00072 }
```

Voici le graphe d'appel pour cette fonction :



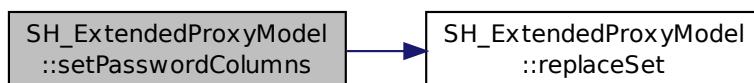
4.40.3.22 void SH_ExtendedProxyModel : setPasswordColumns (QList< int > *passwordCols*) [inherited]

Définition à la ligne 59 du fichier [SH_ExtendedProxyModel.cpp](#).

Références [SH_ExtendedProxyModel : :passwordSet](#), et [SH_ExtendedProxyModel : :replaceSet\(\)](#).

```
00059
00060     replaceSet(this->passwordSet, passwordCols);
00061 }
```

Voici le graphe d'appel pour cette fonction :



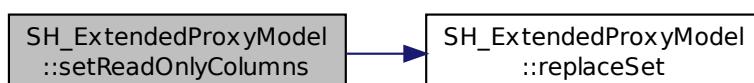
4.40.3.23 void SH_ExtendedProxyModel : :setReadOnlyColumns (QList< int > *readonlyCols*) [inherited]

Définition à la ligne 50 du fichier [SH_ExtendedProxyModel.cpp](#).

Références [SH_ExtendedProxyModel : :readonlySet](#), et [SH_ExtendedProxyModel : :replaceSet\(\)](#).

```
00050
00051     replaceSet(this->readonlySet, readonlyCols);
00052 }
```

Voici le graphe d'appel pour cette fonction :



4.40.3.24 void SH_ExtendedProxyModel : :setSortKeyColumn (int *column*) [inherited]

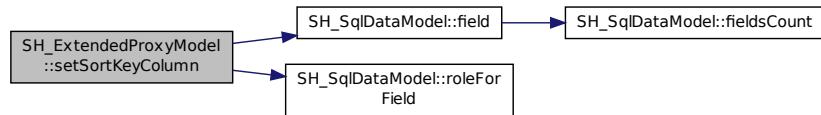
Définition à la ligne 246 du fichier [SH_ExtendedProxyModel.cpp](#).

Références [SH_SqlDataModel : :field\(\)](#), [SH_ExtendedProxyModel : :model](#), [SH_SqlDataModel : :roleForField\(\)](#), [SH_ExtendedProxyModel : :sortChanged\(\)](#), [SH_ExtendedProxyModel : :sortIndex](#), et [SH_SqlDataFields : :sortOrder](#).

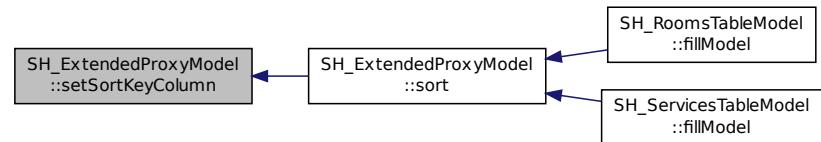
Référencé par [SH_ExtendedProxyModel : :sort\(\)](#).

```
00247 {
00248     this->sortIndex = column;
00249     QSortFilterProxyModel::setSortRole(this->model->roleForField(column));
00250     QSortFilterProxyModel::sort(0, this->model->field(column)->
00251         sortOrder());
00251     emit sortChanged();
00252 }
```

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appels de cette fonction :



4.40.3.25 void SH_ExtendedProxyModel ::sort (int column, Qt ::SortOrder newOrder = Qt ::AscendingOrder) [inherited]

Définition à la ligne 235 du fichier [SH_ExtendedProxyModel.cpp](#).

Références [SH_SqlDataModel ::field\(\)](#), [SH_ExtendedProxyModel ::model\(\)](#), [SH_ExtendedProxyModel ::setSortKeyColumn\(\)](#), et [SH_SqlDataFields ::setSortOrder\(\)](#).

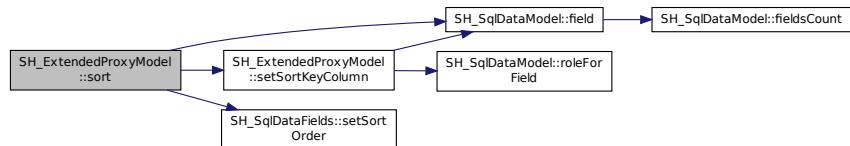
Référencé par [SH_RoomsTableModel ::fillModel\(\)](#), et [SH_ServicesTableModel ::fillModel\(\)](#).

```

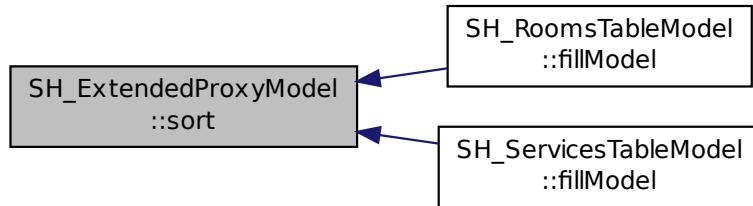
00236 {
00237     this->model->field(column)->setSortOrder(newOrder);
00238     SH_ExtendedProxyModel::setSortKeyColumn(column);
00239 }

```

Voici le graphe d'appel pour cette fonction :



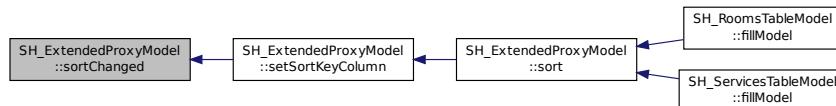
Voici le graphe des appelants de cette fonction :



4.40.3.26 void SH_ExtendedProxyModel::sortChanged() [signal], [inherited]

Référencé par [SH_ExtendedProxyModel::setSortKeyColumn\(\)](#).

Voici le graphe des appelants de cette fonction :



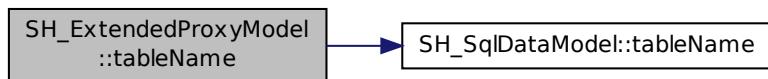
4.40.3.27 const QString SH_ExtendedProxyModel::tableName() const [inline], [inherited]

Définition à la ligne 45 du fichier [SH_ExtendedSqlProxyModel.h](#).

Références [SH_ExtendedProxyModel::model](#), et [SH_SqlDataModel::tableName\(\)](#).

```
00045 { return this->model->tableName(); }
```

Voici le graphe d'appel pour cette fonction :



4.40.4 Documentation des données membres

4.40.4.1 SH_SqlDataModel* SH_ExtendedProxyModel::model [protected], [inherited]

Définition à la ligne 241 du fichier [SH_ExtendedSqlProxyModel.h](#).

Référencé par [SH_ExtendedProxyModel](#) : `:data()`, [SH_ExtendedProxyModel](#) : `:fetch()`, [SH_ExtendedProxyModel](#) : `:field()`, [SH_ExtendedProxyModel](#) : `:fields()`, [SH_ExtendedProxyModel](#) : `:fieldsCount()`, [SH_BillingsTableModel](#) : `:fillModel()`, [SH_RoomsTableModel](#) : `:fillModel()`, [SH_BookingsTableModel](#) : `:fillModel()`, [SH_ExtendedProxyModel](#) : `:isEmpty()`, [SH_ExtendedProxyModel](#) : `:lastError()`, [SH_ExtendedProxyModel](#) : `:roleNames()`, [SH_ExtendedProxyModel](#) : `:setSortKeyColumn()`, [SH_BillingsTableModel](#) : `:SH_BillingsTableModel()`, [SH_BillsTableModel](#) : `:SH_BillsTableModel()`, [SH_BookingsTableModel](#) : `:SH_BookingsTableModel()`, [SH_ClientsTableModel](#) : `:SH_ClientsTableModel()`, [SH_ExtendedProxyModel](#) : `:SH_ExtendedProxyModel()`, [SH_GroupsTableModel](#)(), [SH_RoomsTableModel](#) : `:SH_RoomsTableModel()`, [SH_ServicesTableModel](#) : `:SH_ServicesTableModel()`, [SH_ExtendedProxyModel](#) : `:sort()`, et [SH_ExtendedProxyModel](#) : `:tableName()`.

4.40.5 Documentation des propriétés

4.40.5.1 `bool SH_ExtendedProxyModel::empty [read], [inherited]`

Définition à la ligne 21 du fichier [SH_ExtendedSqlProxyModel.h](#).

4.40.5.2 `QString SH_ExtendedProxyModel::fieldsList [read], [inherited]`

Définition à la ligne 18 du fichier [SH_ExtendedSqlProxyModel.h](#).

4.40.5.3 `QString SH_ExtendedProxyModel::lastError [read], [inherited]`

Définition à la ligne 19 du fichier [SH_ExtendedSqlProxyModel.h](#).

4.40.5.4 `int SH_ExtendedProxyModel::sortKeyColumn [read], [write], [inherited]`

Définition à la ligne 20 du fichier [SH_ExtendedSqlProxyModel.h](#).

4.40.5.5 `QString SH_ExtendedProxyModel::table [read], [inherited]`

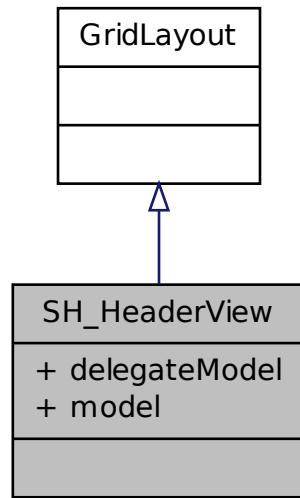
Définition à la ligne 17 du fichier [SH_ExtendedSqlProxyModel.h](#).

La documentation de cette classe a été générée à partir des fichiers suivants :

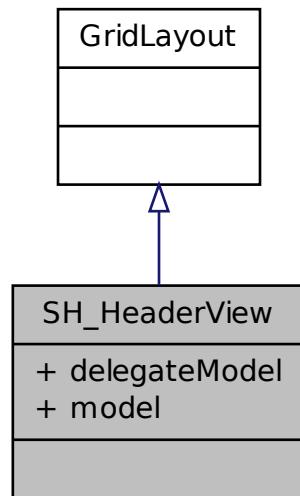
- [models/SH_GroupsTableModel.h](#)
- [models/SH_GroupsTableModel.cpp](#)

4.41 Référence de la classe SH_HeaderView

Graphe d'héritage de SH_HeaderView :



Graphe de collaboration de SH_HeaderView :



Signaux

- void `checked` (int index, bool ch)

- void `up` (int index)

Propriétés

- var `delegateModel`
- alias `model`

4.41.1 Description détaillée

Définition à la ligne 4 du fichier [SH_HeaderView.qml](#).

4.41.2 Documentation des fonctions membres

4.41.2.1 void `SH_HeaderView ::checked` (int *index*, bool *ch*) [signal]

4.41.2.2 void `SH_HeaderView ::up` (int *index*) [signal]

4.41.3 Documentation des propriétés

4.41.3.1 var `SH_HeaderView ::delegateModel`

Définition à la ligne 7 du fichier [SH_HeaderView.qml](#).

4.41.3.2 alias `SH_HeaderView ::model`

Définition à la ligne 9 du fichier [SH_HeaderView.qml](#).

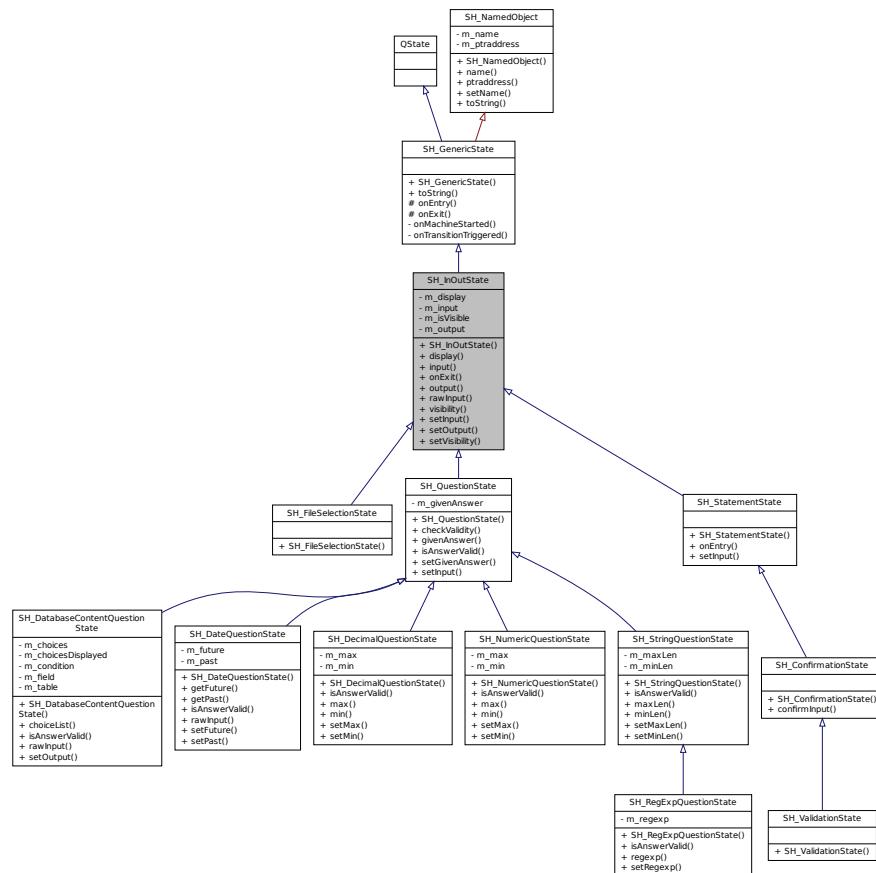
La documentation de cette classe a été générée à partir du fichier suivant :

- views/qml/[SH_HeaderView.qml](#)

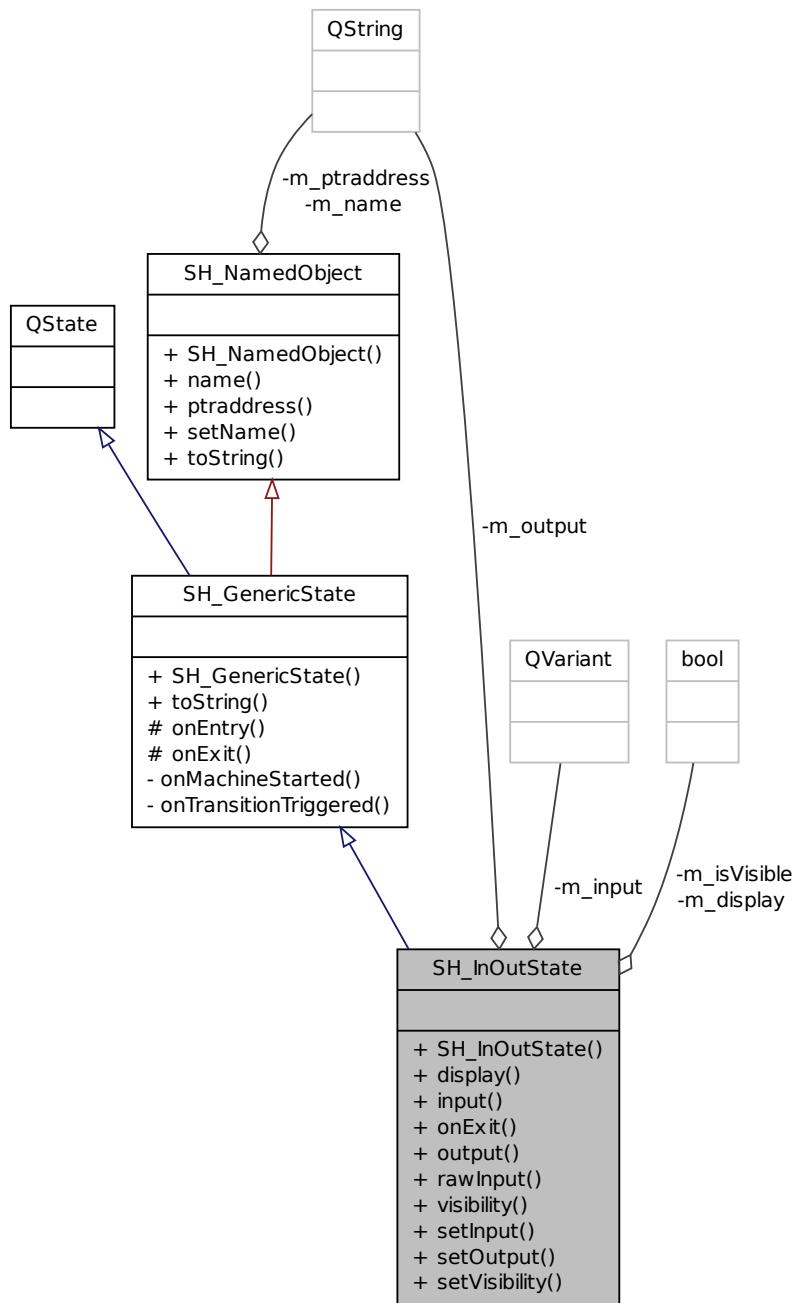
4.42 Référence de la classe SH_InOutState

```
#include <SH_IOSState.h>
```

Graphe d'héritage de SH_InOutState :



Graphe de collaboration de SH_InOutState :



Connecteurs publics

- `virtual void setInput (const QVariant &input)`
- `virtual void setOutput (const QString &output)`
- `virtual void setVisibility (bool isVisible)`

Signaux

- `void next ()`

- void [resendInput \(QVariant input\)](#)
- void [sendOutput \(QVariant output\)](#)

Fonctions membres publiques

- [SH_InOutState \(QString output, QString name, QState *parent=0\)](#)
- void [display \(bool canDisplay\)](#)
- virtual QVariant [input \(\) const](#)
- void [onExit \(QEvent *event\)](#)
- virtual QString [output \(\) const](#)
- virtual QVariant [rawInput \(\) const](#)
- QString [toString \(\)](#)
- bool [visibility \(\)](#)

Fonctions membres protégées

- void [onEntry \(QEvent *event\)](#)

Attributs privés

- bool [m_display](#)
 [m_display](#)
- QVariant [m_input](#)
 [m_input](#)
- bool [m_isVisible](#)
 [m_isVisible](#)
- QString [m_output](#)
 [m_output](#)

4.42.1 Description détaillée

Définition à la ligne 11 du fichier [SH_IOState.h](#).

4.42.2 Documentation des constructeurs et destructeur

4.42.2.1 SH_InOutState : :SH_InOutState (QString output, QString name, QState * parent = 0)

Définition à la ligne 9 du fichier [SH_IOState.cpp](#).

```
00009
00010      SH_GenericState(name, parent), m_output(output),
00011      m_isVisible(true)
00012 }
```

:

4.42.3 Documentation des fonctions membres

4.42.3.1 void SH_InOutState : :display (bool canDisplay)

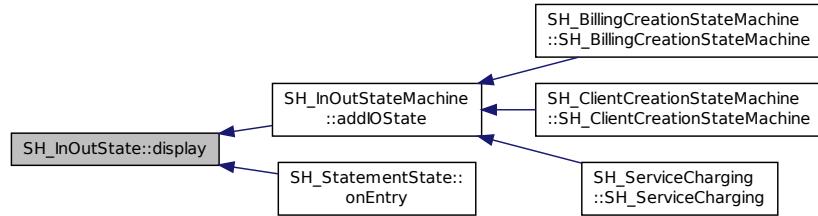
Définition à la ligne 95 du fichier [SH_IOState.cpp](#).

Références [m_display](#), [m_isVisible](#), [m_output](#), et [sendOutput\(\)](#).

Référencé par [SH_InOutStateMachine : :addIOState\(\)](#), et [SH_StatementState : :onEntry\(\)](#).

```
00096 {
00097     m_display=canDisplay;
00098     if(m_display && !m_output.isEmpty() && m_isVisible) {
00099         qDebug() << "resalut !" << QVariant(m_output);
00100         emit sendOutput(QVariant(m_output));
00101     }
00102 }
```

Voici le graphe des appelants de cette fonction :



4.42.3.2 QVariant SH_InOutState ::input() const [virtual]

Définition à la ligne 20 du fichier [SH_IOState.cpp](#).

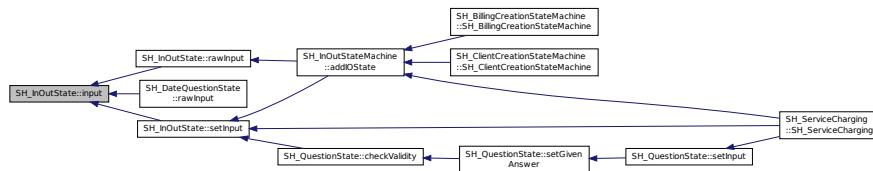
Références [m_input](#).

Référencé par [rawInput\(\)](#), [SH_DateQuestionState ::rawInput\(\)](#), et [setInput\(\)](#).

```

00021 {
00022     return m_input;
00023 }
  
```

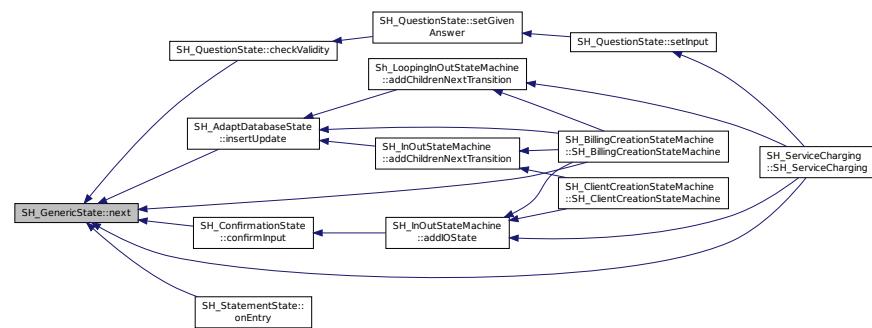
Voici le graphe des appelants de cette fonction :



4.42.3.3 void SH_GenericState ::next() [signal], [inherited]

Référencé par [SH_QuestionState ::checkValidity\(\)](#), [SH_ConfirmationState ::confirmInput\(\)](#), [SH_AdaptDatabaseState ::insertUpdate\(\)](#), [SH_StatementState ::onEntry\(\)](#), [SH_BillingCreationStateMachine ::SH_BillingCreationStateMachine\(\)](#), et [SH_ServiceCharging ::SH_ServiceCharging\(\)](#).

Voici le graphe des appelants de cette fonction :



4.42.3.4 void SH_GenericState ::onEntry (QEvent * event) [protected], [inherited]

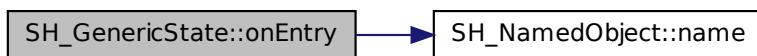
Définition à la ligne 62 du fichier [SH_GenericDebugableState.cpp](#).

Références [SH_NamedObject ::name\(\)](#).

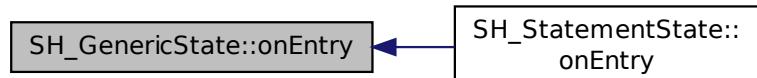
Référencé par [SH_StatementState ::onEntry\(\)](#).

```
00063 {
00064     Q_UNUSED(event);
00065     qDebug() << "Machine: " << machine() ->objectName() << " entered " << name();
00066 }
```

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appels de cette fonction :



4.42.3.5 void SH_InOutState ::onExit (QEvent * event)

Définition à la ligne 110 du fichier [SH_IOState.cpp](#).

Références [m_input](#), [m_isVisible](#), [SH_GenericState ::onExit\(\)](#), et [resendInput\(\)](#).

```
00111 {
00112     SH_GenericState::onExit(event);
00113     if(m_isVisible) {
00114         emit resendInput(m_input);
00115     }
00116 }
```

Voici le graphe d'appel pour cette fonction :



4.42.3.6 QString SH_InOutState ::output() const [virtual]

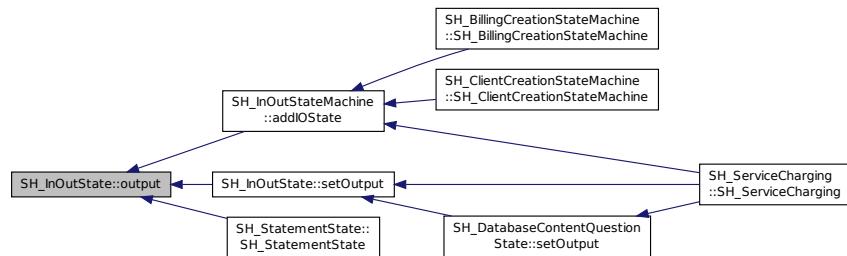
Définition à la ligne 56 du fichier [SH_IOState.cpp](#).

Références [m_output](#).

Référencé par [SH_InOutStateMachine ::addIOState\(\)](#), [setOutput\(\)](#), et [SH_StatementState ::SH_StatementState\(\)](#).

```
00057 {
00058     return m_output;
00059 }
```

Voici le graphe des appels de cette fonction :



4.42.3.7 QVariant SH_InOutState ::rawInput() const [virtual]

Réimplémentée dans [SH_DateQuestionState](#), et [SH_DatabaseContentQuestionState](#).

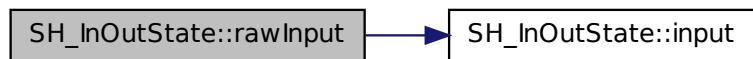
Définition à la ligne 30 du fichier [SH_IOState.cpp](#).

Références [input\(\)](#).

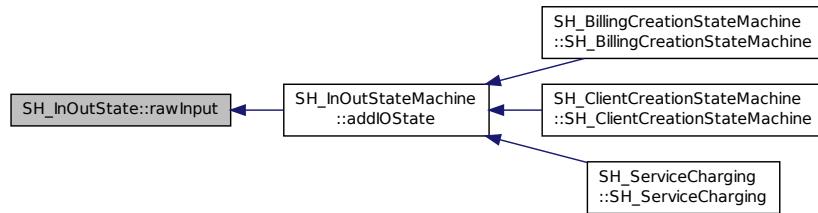
Référencé par [SH_InOutStateMachine ::addIOState\(\)](#).

```
00031 {
00032     return input();
00033 }
```

Voici le graphe d'appel pour cette fonction :



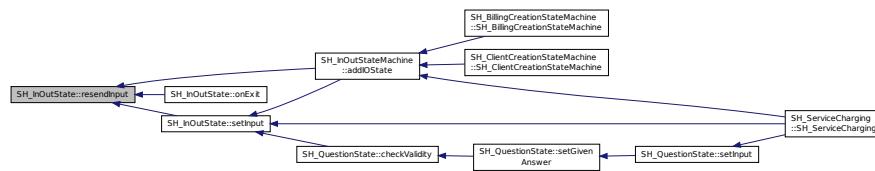
Voici le graphe des appels de cette fonction :



4.42.3.8 void SH_InOutState ::resendInput (QVariant *input*) [signal]

Référencé par [SH_InOutStateMachine ::addIOState\(\)](#), [onExit\(\)](#), et [setInput\(\)](#).

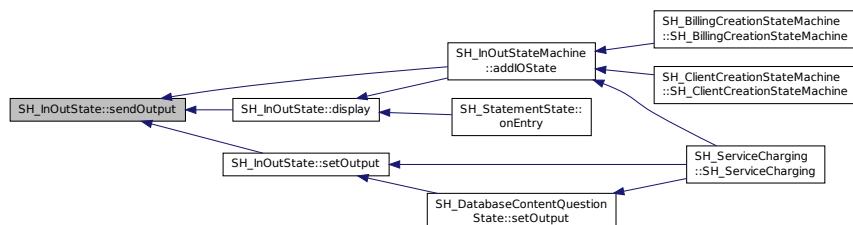
Voici le graphe des appels de cette fonction :



4.42.3.9 void SH_InOutState ::sendOutput (QVariant *output*) [signal]

Référencé par [SH_InOutStateMachine ::addIOState\(\)](#), [display\(\)](#), et [setOutput\(\)](#).

Voici le graphe des appels de cette fonction :



4.42.3.10 void SH_InOutState ::setInput (const QVariant & *input*) [virtual], [slot]

Réimplémentée dans [SH_QuestionState](#), et [SH_StatementState](#).

Définition à la ligne 41 du fichier [SH_IState.cpp](#).

Références [input\(\)](#), [m_input](#), [m_isVisible](#), et [resendInput\(\)](#).

Référencé par [SH_InOutStateMachine ::addIOState\(\)](#), [SH_QuestionState ::checkValidity\(\)](#), et [SH_ServiceCharging ::SH_ServiceCharging\(\)](#).

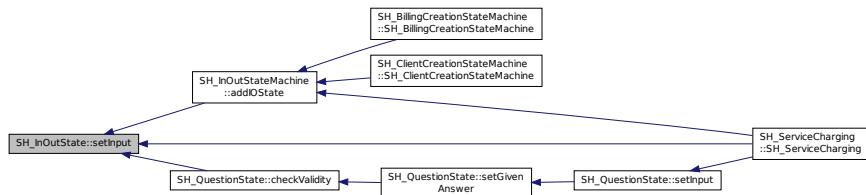
```

00042 {
00043     qDebug() << "new input " << input.toString();
00044     m_input = input;
00045     if(m_isVisible) {
00046         emit resendInput(m_input);
00047     }
00048 }
```

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appels de cette fonction :



4.42.3.11 void SH_InOutState ::setOutput(const QString & output) [virtual], [slot]

Réimplémentée dans [SH_DatabaseContentQuestionState](#).

Définition à la ligne 68 du fichier [SH_IOState.cpp](#).

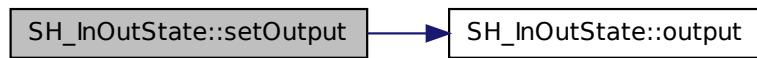
Références [m_isVisible](#), [m_output](#), [output\(\)](#), et [sendOutput\(\)](#).

Référencé par [SH_DatabaseContentQuestionState ::setOutput\(\)](#), et [SH_ServiceCharging ::SH_ServiceCharging\(\)](#).

```

00069 {
00070     m_output = output;
00071     if(m_isVisible) {
00072         emit sendOutput(QVariant(m_output));
00073     }
00074 }
```

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :



4.42.3.12 void SH_InOutState ::setVisibility (bool isVisible) [virtual], [slot]

Définition à la ligne 81 du fichier [SH_IOState.cpp](#).

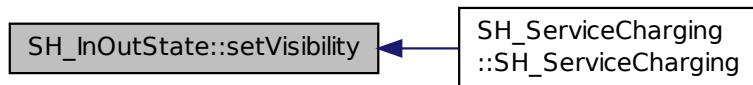
Références [m_isVisible](#).

Référencé par [SH_ServiceCharging ::SH_ServiceCharging\(\)](#).

```

00082 {
00083     m_isVisible = isVisible;
00084 }
```

Voici le graphe des appelants de cette fonction :



4.42.3.13 QString SH_GenericState ::toString () [virtual], [inherited]

Réimplémentée à partir de [SH_NamedObject](#).

Définition à la ligne 21 du fichier [SH_GenericDebugableState.cpp](#).

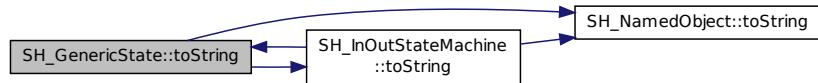
Références [SH_NamedObject ::toString\(\)](#), et [SH_InOutStateMachine ::toString\(\)](#).

Référencé par [SH_InOutStateMachine ::addChildsNextTransition\(\)](#), [SH_DateQuestionState ::rawInput\(\)](#), et [SH_InOutStateMachine ::toString\(\)](#).

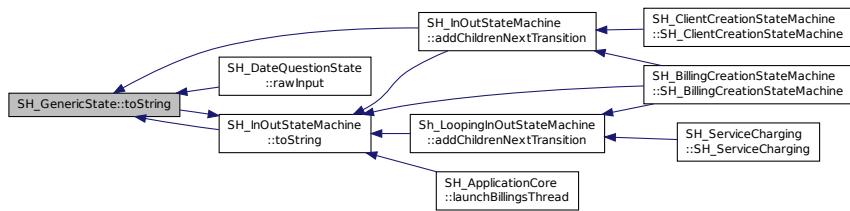
```

00022 {
00023     QStateMachine* machine = this->machine();
00024     SH_InOutStateMachine* mach = qobject_cast<
00025         SH_InOutStateMachine *>(machine);
00026     if(mach) {
00027         return SH_NamedObject::toString() + " [in "+mach->
00028             toString()+"] ";
00029     } else {
00030         return SH_NamedObject::toString();
00031     }
00032 }
```

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :



4.42.3.14 bool SH_InOutState::visibility()

Définition à la ligne 91 du fichier [SH_IOState.cpp](#).

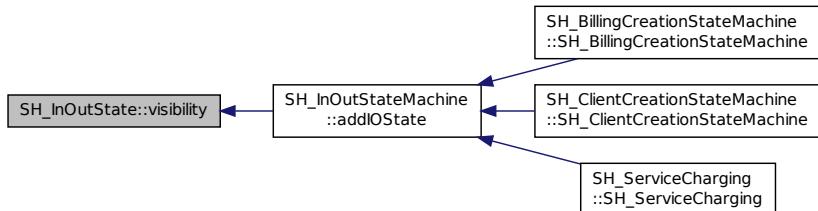
Références [m_isVisible](#).

Référencé par [SH_InOutStateMachine::addIOState\(\)](#).

```

00091
00092     return m_isVisible;
00093 }
  
```

Voici le graphe des appelants de cette fonction :



4.42.4 Documentation des données membres

4.42.4.1 bool SH_InOutState::m_display [private]

`m_display`

Définition à la ligne 128 du fichier [SH_IOState.h](#).

Référencé par [display\(\)](#).

4.42.4.2 QVariant SH_InOutState ::m_input [private]

m_input

Définition à la ligne 116 du fichier [SH_IOState.h](#).

Référencé par [input\(\)](#), [onExit\(\)](#), et [setInput\(\)](#).

4.42.4.3 bool SH_InOutState ::m_isVisible [private]

m_isVisible

Définition à la ligne 124 du fichier [SH_IOState.h](#).

Référencé par [display\(\)](#), [onExit\(\)](#), [setInput\(\)](#), [setOutput\(\)](#), [setVisibility\(\)](#), et [visibility\(\)](#).

4.42.4.4 QString SH_InOutState ::m_output [private]

m_output

Définition à la ligne 120 du fichier [SH_IOState.h](#).

Référencé par [display\(\)](#), [output\(\)](#), et [setOutput\(\)](#).

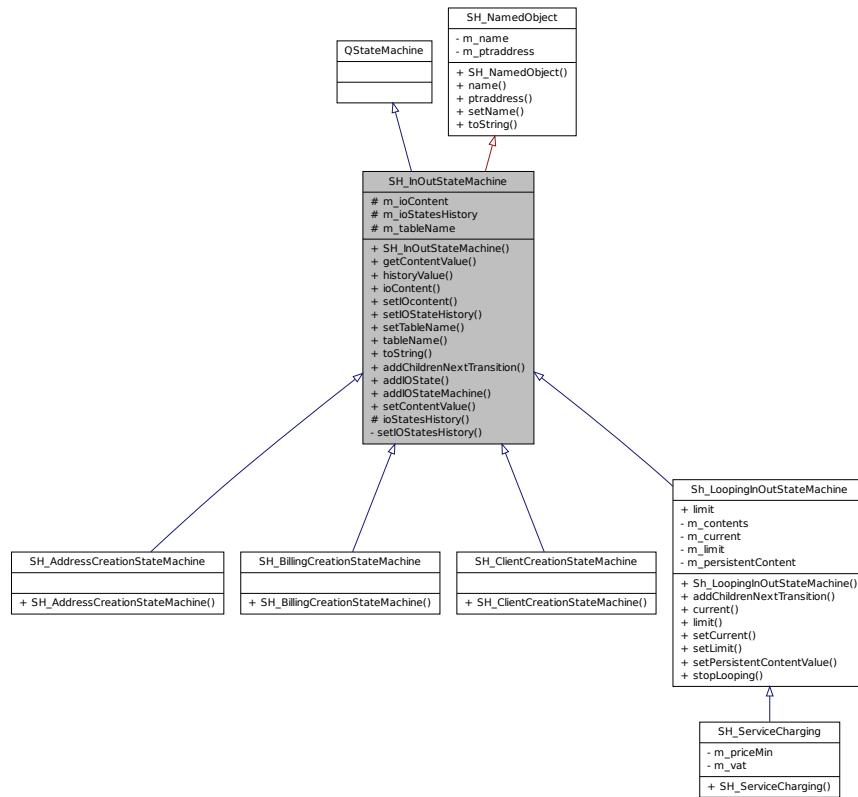
La documentation de cette classe a été générée à partir des fichiers suivants :

- logic/[SH_IOState.h](#)
- logic/[SH_IOState.cpp](#)

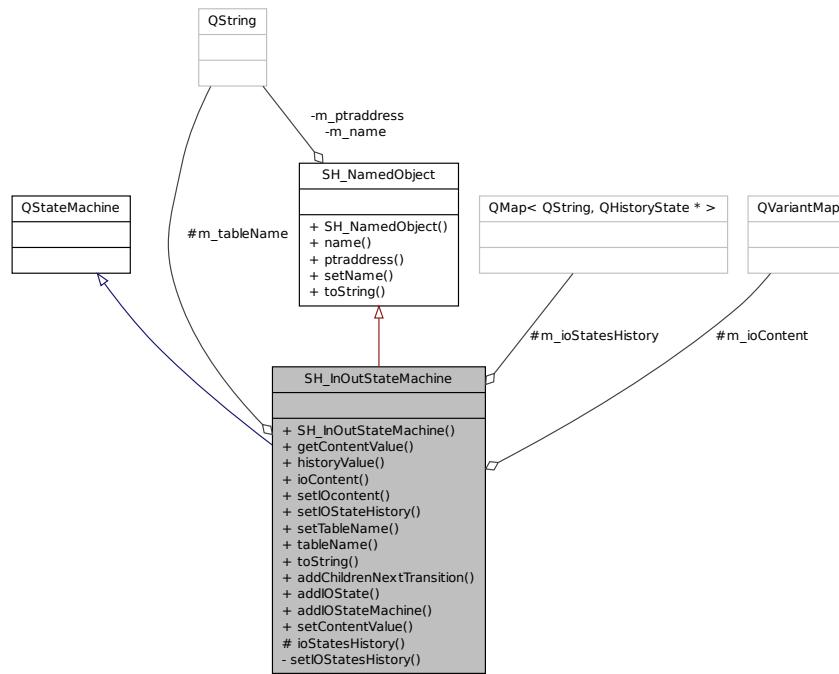
4.43 Référence de la classe SH_InOutStateMachine

```
#include <SH_IStateMachine.h>
```

Graphe d'héritage de SH_InOutStateMachine :



Graphe de collaboration de SH_InOutStateMachine :



Connecteurs publics

- void `addChildsNextTransition` (QAbstractState *previousState, QAbstractState *nextState)
- void `addIOState` (SH_InOutState *state, QString field)
- void `addIOStateMachine` (SH_InOutStateMachine *fsm)
- void `setContentValue` (QVariant content, QString field)

Signaux

- void `cancelReplacement` ()
- void `clearAll` ()
- void `confirmInput` ()
- void `displayCalendar` ()
- void `displayFileDialog` ()
- void `next` ()
- void `receiveInput` (QString input)
- void `replaceInput` (QString field)
- void `resendText` (QString text, bool editable=false)
- void `sendText` (QString text, bool editable=false)
- void `validateInput` ()

Fonctions membres publiques

- SH_InOutStateMachine (QString tableName, QString name="", QObject *parent=0)
- QVariant `getContentValue` (QString field)
- QHistoryState * `historyValue` (QString field)
- QVariantMap `ioContent` () const
- void `setIOcontent` (const QVariantMap &ioContent)
- void `setIOStateHistory` (QHistoryState *state, QString field)
- void `setTableName` (const QString &tableName)
- QString `tableName` () const
- QString `toString` ()

Fonctions membres protégées

- QMap< QString, QHistoryState * > `ioStatesHistory` () const

Attributs protégés

- QVariantMap `m_ioContent`
 `m_ioContent`
- QMap< QString, QHistoryState * > `m_ioStatesHistory`
 `m_ioStatesHistory`
- QString `m_tableName`
 `m_tableName`

Fonctions membres privées

- virtual QString `name` () const
- QString `ptraddress` () const
- void `setIOStatesHistory` (const QMap< QString, QHistoryState * > &ioStatesHistory)
- virtual void `setName` (const QString &name)

4.43.1 Description détaillée

Définition à la ligne 15 du fichier `SH_IOStateMachine.h`.

4.43.2 Documentation des constructeurs et destructeur

4.43.2.1 SH_InOutStateMachine : :SH_InOutStateMachine (QString tableName, QString name = " ", QObject * parent = 0)

Définition à la ligne 14 du fichier [SH_IOStateMachine.cpp](#).

```
00014
00015     QStateMachine(parent), SH_NamedObject(name),
00016     m_tableName(tableName)
00017 {
00018     qDebug() << "nouvelle IOStateMachine";
00019 }
```

4.43.3 Documentation des fonctions membres

4.43.3.1 void SH_InOutStateMachine : :addChildrenNextTransition (QAbstractState * previousState, QAbstractState * nextState) [slot]

Définition à la ligne 250 du fichier [SH_IOStateMachine.cpp](#).

Références `clearAll()`, `historyValue()`, `SH_AdaptDatabaseState : :insertUpdate()`, `m_ioContent`, `m_tableName`, `next()`, `replaceInput()`, `sendText()`, `setContentValue()`, `SH_GenericState : :toString()`, et `toString()`.

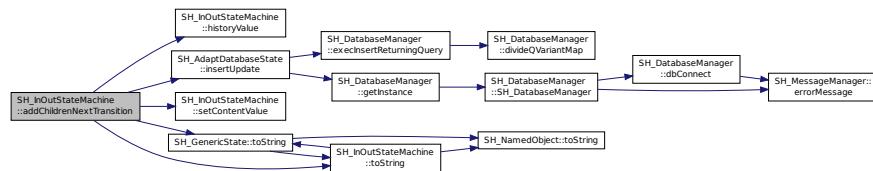
Référencé par `SH_BillingCreationStateMachine : :SH_BillingCreationStateMachine()`, et `SH_ClientCreationStateMachine : :SH_ClientCreationStateMachine()`.

```
00251 {
00252     SH_InOutStateMachine* fsmPreviousState = qobject_cast<
00253         SH_InOutStateMachine*>(previousState);
00254     SH_GenericState* genPreviousState = qobject_cast<
00255         SH_GenericState*>(previousState);
00256     QFinalState* final = qobject_cast<QFinalState*>(nextState);
00257     if(final) {
00258         SH_AdaptDatabaseState* saveState = new
00259             SH_AdaptDatabaseState("enregistrement de la machine "+
00260             toString());
00261         if(genPreviousState) {
00262             genPreviousState->addTransition(genPreviousState, SIGNAL(next()), saveState);
00263         }
00264         if(fsmPreviousState) {
00265             fsmPreviousState->addTransition(fsmPreviousState, SIGNAL(next()), saveState);
00266         }
00267         if(genPreviousState || fsmPreviousState) {
00268             connect(previousState, &QAbstractState::exited, [=]() {
00269                 connect(saveState, &QAbstractState::entered, [=]() {
00270                     emit this->sendText("Merci !");
00271                     setContentValue(saveState->insertUpdate(
00272                         m_tableName, m_ioContent), "ID");
00273                     emit this->clearAll();
00274                 });
00275             });
00276             saveState->addTransition(saveState, SIGNAL(next()), final);
00277         }
00278     } else {
00279         if(genPreviousState) {
00280             qDebug() << "next transition between " << genPreviousState->toString() << " and " <<
00281             nextState;
00282             genPreviousState->addTransition(genPreviousState, SIGNAL(next()), nextState);
00283         }
00284         if(fsmPreviousState) {
00285             qDebug() << "next transition between " << fsmPreviousState->toString() << " and " <<
00286             nextState;
00287             fsmPreviousState->addTransition(fsmPreviousState, SIGNAL(next()), nextState);
00288         }
00289     }
00290     if(genPreviousState) {
00291         /*à faire au moment de l'entrée dans l'état previousState*/
00292         connect(genPreviousState, &QAbstractState::entered, [=]() {
00293             connect(this, &SH_InOutStateMachine::replaceInput, [=](
00294                 QString field) {
00295                 /*après avoir demandé à revenir sur un état précédent, on attend la fin de l'état actuel
00296                 puis on retourne à l'historique de l'état désiré; celui-ci fini, on passe à l'état qui aurait du suivre celui
00297                 pendant lequel on a demandé à revenir sur un état précédent*/
00298                 QHistoryState* hState = historyValue(field);
00299                 if(hState) { /*si l'historique existe (on a déjà quitté l'état voulu)*/
00300                     hState->parentState()->addTransition(hState->parentState(), SIGNAL(
00301                         next()), nextState);
00302                     genPreviousState->addTransition(genPreviousState, SIGNAL(
00303                         next()), hState);
00304                 }
00305             });
00306         });
00307     }
00308 }
```

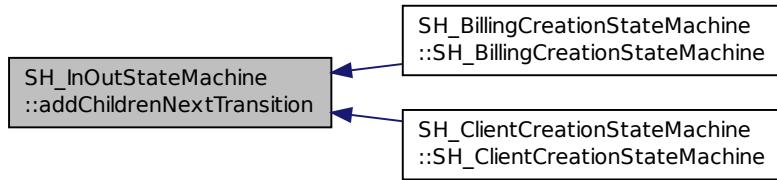
```

00293         });
00294     });
00295 }
00296 }
```

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :



4.43.3.2 void SH_InOutStateMachine ::addIOState (SH_InOutState * state, QString field) [slot]

Définition à la ligne 110 du fichier [SH_IOStateMachine.cpp](#).

Références [SH_ConfirmationState ::confirmInput\(\)](#), [SH_InOutState ::display\(\)](#), [displayCalendar\(\)](#), [displayFileDialog\(\)](#), [SH_InOutState ::output\(\)](#), [SH_InOutState ::rawInput\(\)](#), [receiveInput\(\)](#), [SH_InOutState ::resendInput\(\)](#), [resendText\(\)](#), [SH_InOutState ::sendOutput\(\)](#), [sendText\(\)](#), [setContentValue\(\)](#), [SH_InOutState ::setInput\(\)](#), [setIOStateHistory\(\)](#), [validateInput\(\)](#), et [SH_InOutState ::visibility\(\)](#).

Référencé par [SH_BillingCreationStateMachine ::SH_BillingCreationStateMachine\(\)](#), [SH_ClientCreationStateMachine ::SH_ClientCreationStateMachine\(\)](#), et [SH_ServiceCharging ::SH_ServiceCharging\(\)](#).

```

00111 {
00112     /*à faire au moment de l'entrée dans l'état state*/
00113     connect(state, &QState::entered, [=]() {
00114         qDebug() << "entered !";
00115         state->display(true);
00116         connect(this, &SH_InOutStateMachine::receiveInput, state, &
00117             SH_InOutState::setInput); /* la réception d'une valeur entraîne son enregistrement
00118             comme entrée de l'utilisateur auprès de l'état*/
00119         connect(this, &SH_InOutStateMachine::receiveInput, [=](QString in
00120         ) { qDebug() << "hello world !"; state->setInput(in)}); /* la réception d'une valeur entraîne son
00121             enregistrement comme entrée de l'utilisateur auprès de l'état*/
00122         connect(state, &SH_InOutState::sendOutput, [=](QVariant out) {qDebug() <<
00123             "connected !"; emit this->sendText(out.toString(), false);});
00124         connect(state, &SH_InOutState::resendInput, [=](QVariant in) {emit this->
00125             resendText(in.toString(), true);});
00126         if(state->visibility()) {
00127             state->sendOutput(QVariant(state->output()));
00128         } else {
00129             qDebug() << "invisible";
00130         }
00131     });
00132     SH_ValidationState *validationState = qobject_cast<
00133         SH_ValidationState*>(state);

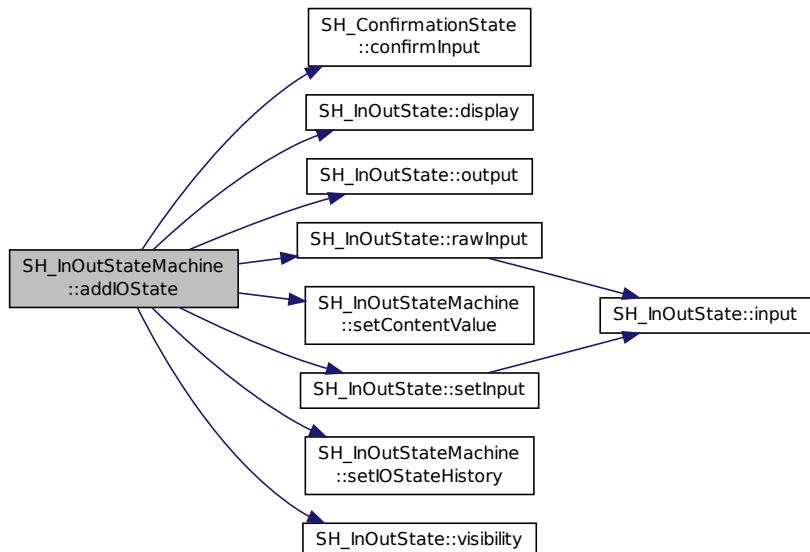
```

```

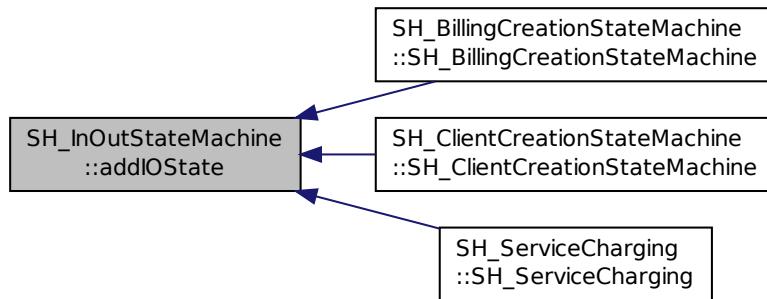
00127     if(validationState) {
00128         /*à faire au moment de l'entrée dans l'état state*/
00129         connect(validationState, &QState::entered, [=]() {
00130             connect(this, &SH_InOutStateMachine::validateInput,
00131                     validationState, &SH_ValidationState::confirmInput);
00132         });
00133         SH_ConfirmationState *confirmationState = qobject_cast<
00134             SH_ConfirmationState*>(state);
00135         if(confirmationState) {
00136             /*à faire au moment de l'entrée dans l'état state*/
00137             connect(confirmationState, &QState::entered, [=]() {
00138                 connect(this, &SH_InOutStateMachine::validateInput,
00139                         confirmationState, &SH_ConfirmationState::confirmInput);
00140             });
00141             SH_DateQuestionState *dateState = qobject_cast<
00142                 SH_DateQuestionState*>(state);
00143             if(dateState) {
00144                 /*à faire au moment de l'entrée dans l'état state*/
00145                 connect(dateState, &QState::entered, this, &
00146                         SH_InOutStateMachine::displayCalendar);
00147             }
00148             SH_FileSelectionState *fileState = qobject_cast<
00149                 SH_FileSelectionState*>(state);
00150             if(fileState) {
00151                 /*à faire au moment de l'entrée dans l'état state*/
00152                 connect(fileState, &QState::entered, this, &
00153                         SH_InOutStateMachine::displayFileDialog);
00154             }
00155             /*à faire au moment de la sortie de l'état state*/
00156             connect(state, &QState::exited, [=]() {
00157                 qDebug() << "exited !";
00158                 if(!field.isEmpty()) {
00159                     setContentValue(state->rawInput(), field);
00160                     /*gestion de l'historique des états pour pouvoir revenir à l'état state après l'avoir quitté*/
00161                     QHistoryState* hState = new QHistoryState(state);
00162                     setIOStateHistory(hState, field);
00163                 }
00164                 state->disconnect(this); /*plus aucune action sur l'état ne pourra être provoquée par la machine*/
00165             });
00166         }
00167     }

```

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appels de cette fonction :



4.43.3.3 void SH_InOutStateMachine ::addIOStateMachine (SH_InOutStateMachine * fsm) [slot]

Définition à la ligne 175 du fichier [SH_IOStateMachine.cpp](#).

Références [cancelReplacement\(\)](#), [confirmInput\(\)](#), [displayCalendar\(\)](#), [receiveInput\(\)](#), [replaceInput\(\)](#), [resendText\(\)](#), [sendText\(\)](#), et [validateInput\(\)](#).

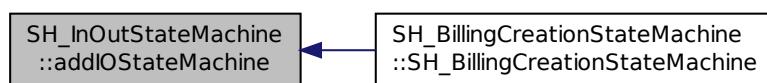
Référencé par [SH_BillingCreationStateMachine ::SH_BillingCreationStateMachine\(\)](#).

```

00176 {
00177     /*à faire au moment de l'entrée dans la machine d'état fsm*/
00178     connect(fsm, &QState::entered, [=]() {
00179         connect(this, &SH_InOutStateMachine::receiveInput, fsm, &
00180             SH_InOutStateMachine::receiveInput);
00180         connect(this, &SH_InOutStateMachine::sendText, fsm, &
00181             SH_InOutStateMachine::sendText);
00181         connect(this, &SH_InOutStateMachine::resendText, fsm, &
00182             SH_InOutStateMachine::resendText);
00182         connect(this, &SH_InOutStateMachine::confirmInput, fsm, &
00183             SH_InOutStateMachine::confirmInput);
00183         connect(this, &SH_InOutStateMachine::validateInput, fsm, &
00184             SH_InOutStateMachine::validateInput);
00184         connect(this, &SH_InOutStateMachine::replaceInput, fsm, &
00185             SH_InOutStateMachine::replaceInput);
00185         connect(this, &SH_InOutStateMachine::cancelReplacement, fsm,
00186             &SH_InOutStateMachine::cancelReplacement);
00186         connect(this, &SH_InOutStateMachine::displayCalendar, fsm, &
00187             SH_InOutStateMachine::displayCalendar);
00187     });
00188     /*à faire au moment de la sortie de la machine d'état fsm*/
00189     connect(fsm, &QState::exited, [=]() {
00190         fsm->disconnect(this); /*plus aucune action sur la machine d'état fille ne pourra être provoquée
00190         par la machine mère*/
00191     });
00192
00193 }

```

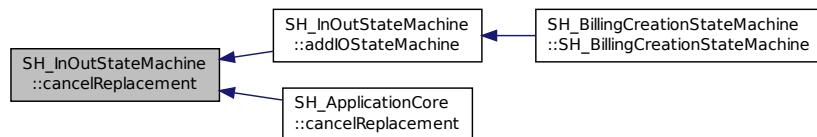
Voici le graphe des appels de cette fonction :



4.43.3.4 void SH_InOutStateMachine ::cancelReplacement() [signal]

Référencé par [addIOStateMachine\(\)](#), et [SH_ApplicationCore ::cancelReplacement\(\)](#).

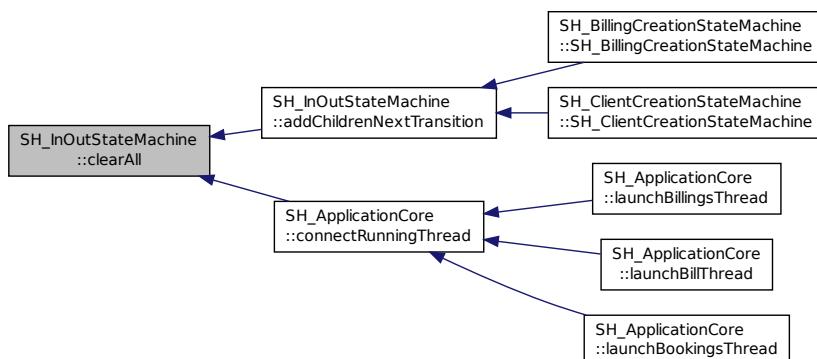
Voici le graphe des appelants de cette fonction :



4.43.3.5 void SH_InOutStateMachine ::clearAll() [signal]

Référencé par [addChildrenNextTransition\(\)](#), et [SH_ApplicationCore ::connectRunningThread\(\)](#).

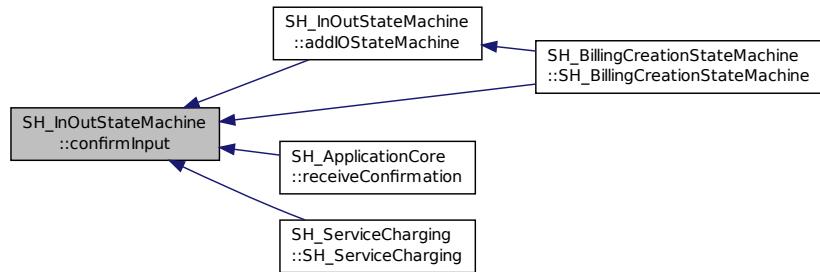
Voici le graphe des appelants de cette fonction :



4.43.3.6 void SH_InOutStateMachine ::confirmInput() [signal]

Référencé par [addIOStateMachine\(\)](#), [SH_ApplicationCore ::receiveConfirmation\(\)](#), [SH_BillingCreationStateMachine ::SH_BillingCreationStateMachine\(\)](#), et [SH_ServiceCharging ::SH_ServiceCharging\(\)](#).

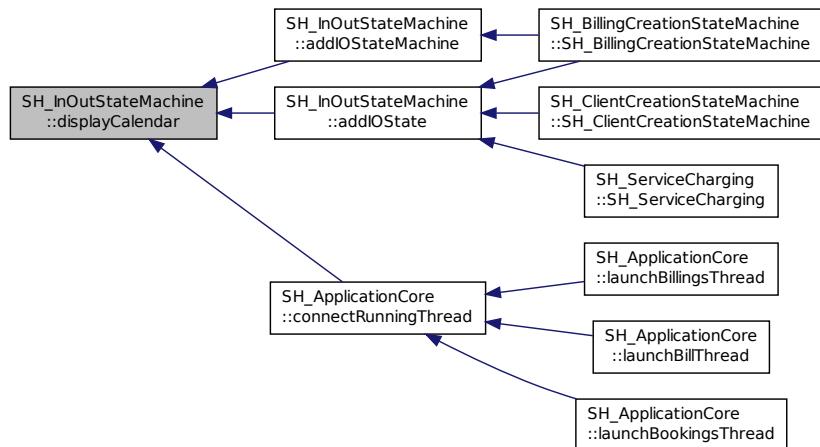
Voici le graphe des appelants de cette fonction :



4.43.3.7 void SH_InOutStateMachine ::displayCalendar() [signal]

Référencé par `addIOState()`, `addIOStateMachine()`, et `SH_ApplicationCore ::connectRunningThread()`.

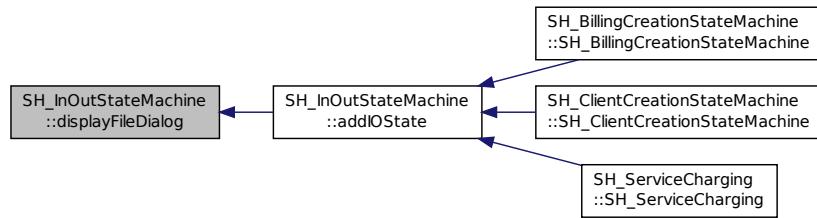
Voici le graphe des appelants de cette fonction :



4.43.3.8 void SH_InOutStateMachine ::displayFileDialog() [signal]

Référencé par `addIOState()`.

Voici le graphe des appelants de cette fonction :



4.43.3.9 QVariant SH_InOutStateMachine ::getContentValue (QString field)

Définition à la ligne 65 du fichier [SH_IOStateMachine.cpp](#).

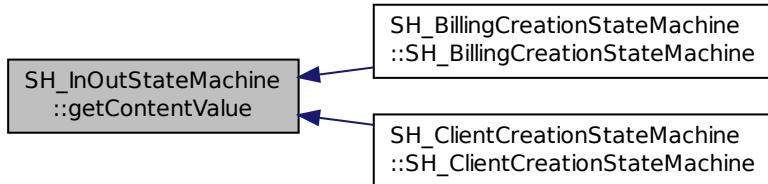
Références [m_ioContent](#).

Référencé par [SH_BillingCreationStateMachine ::SH_BillingCreationStateMachine\(\)](#), et [SH_ClientCreationStateMachine ::SH_ClientCreationStateMachine\(\)](#).

```

00066 {
00067     return m_ioContent.value(field);
00068 }
  
```

Voici le graphe des appelants de cette fonction :



4.43.3.10 QHistoryState * SH_InOutStateMachine ::historyValue (QString field)

Définition à la ligne 238 du fichier [SH_IOStateMachine.cpp](#).

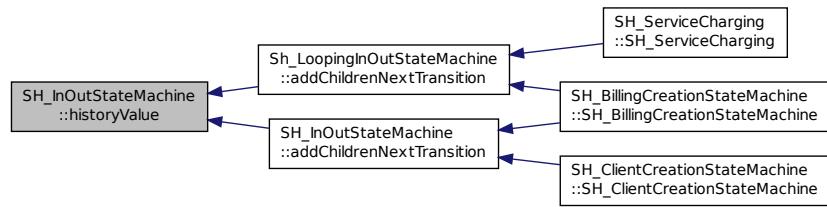
Références [m_ioStatesHistory](#).

Référencé par [Sh_LoopingInOutStateMachine ::addChildsNextTransition\(\)](#), et [addChildsNextTransition\(\)](#).

```

00239 {
00240     return m_ioStatesHistory.value(field);
00241 }
  
```

Voici le graphe des appelants de cette fonction :



4.43.3.11 QVariantMap SH_InOutStateMachine ::ioContent() const

Définition à la ligne 43 du fichier [SH_IOStateMachine.cpp](#).

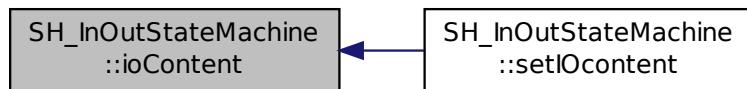
Références [m_ioContent](#).

Référencé par [setIOcontent\(\)](#).

```

00044 {
00045     return m_ioContent;
00046 }
```

Voici le graphe des appelants de cette fonction :



4.43.3.12 QMap<QString, QHistoryState * > SH_InOutStateMachine ::ioStatesHistory() const [protected]

Définition à la ligne 202 du fichier [SH_IOStateMachine.cpp](#).

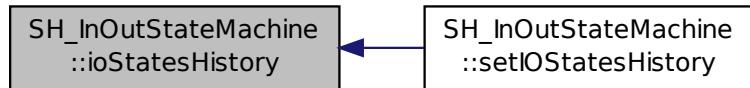
Références [m_ioStatesHistory](#).

Référencé par [setIOStatesHistory\(\)](#).

```

00203 {
00204     return m_ioStatesHistory;
00205 }
```

Voici le graphe des appelants de cette fonction :



4.43.3.13 QString SH_NamedObject::name() const [virtual], [inherited]

Définition à la ligne 32 du fichier [SH_NamedObject.cpp](#).

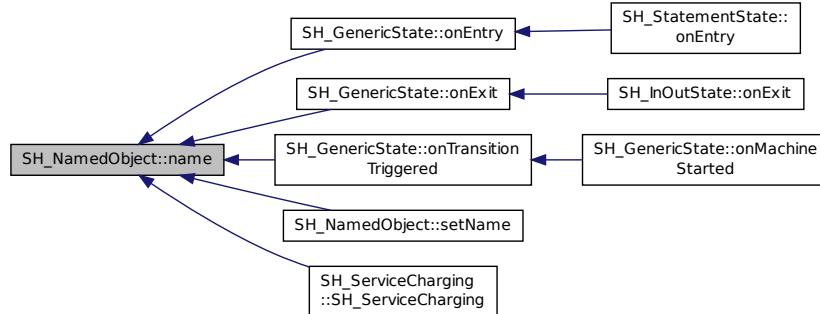
Références [SH_NamedObject::m_name](#).

Référencé par [SH_GenericState::onEntry\(\)](#), [SH_GenericState::onExit\(\)](#), [SH_GenericState::onTransitionTriggered\(\)](#), [SH_NamedObject::setName\(\)](#), et [SH_ServiceCharging::SH_ServiceCharging\(\)](#).

```

00033 {
00034     return m_name;
00035 }
  
```

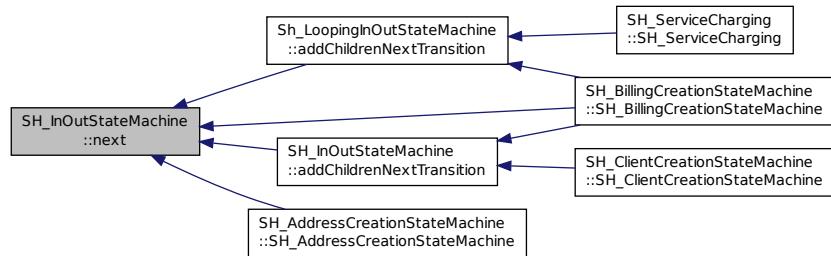
Voici le graphe des appelants de cette fonction :



4.43.3.14 void SH_InOutStateMachine::next() [signal]

Référencé par [Sh_LoopingInOutStateMachine::addChildrenNextTransition\(\)](#), [addChildrenNextTransition\(\)](#), [SH_AddressCreationStateMachine::SH_AddressCreationStateMachine\(\)](#), et [SH_BillingCreationStateMachine::SH_BillingCreationStateMachine\(\)](#).

Voici le graphe des appelants de cette fonction :



4.43.3.15 QString SH_NamedObject ::ptraddress () const [inherited]

Définition à la ligne 54 du fichier [SH_NamedObject.cpp](#).

Références [SH_NamedObject ::m_ptraddress](#).

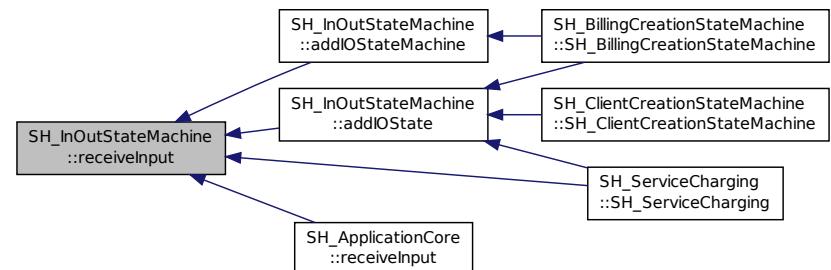
```

00055 {
00056     return m_ptraddress;
00057 }
  
```

4.43.3.16 void SH_InOutStateMachine ::receiveInput (QString input) [signal]

Référencé par [addIOState\(\)](#), [addIOStateMachine\(\)](#), [SH_ApplicationCore ::receiveInput\(\)](#), et [SH_ServiceCharging ::SH_ServiceCharging\(\)](#).

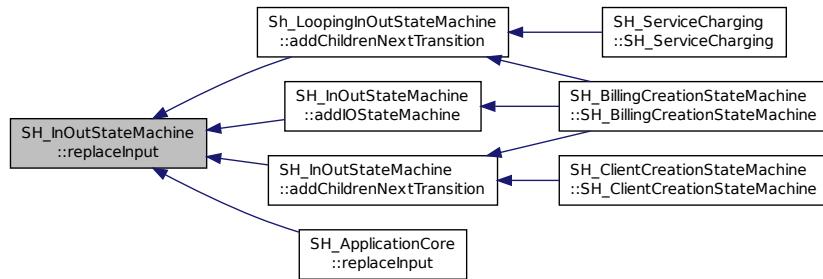
Voici le graphe des appelants de cette fonction :



4.43.3.17 void SH_InOutStateMachine ::replaceInput (QString field) [signal]

Référencé par [Sh_LoopingInOutStateMachine ::addChildrenNextTransition\(\)](#), [addChildrenNextTransition\(\)](#), [addIOStateMachine\(\)](#), et [SH_ApplicationCore ::replaceInput\(\)](#).

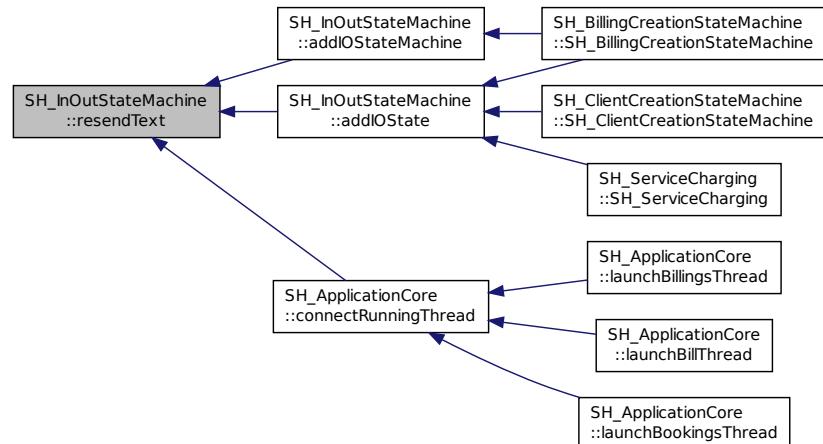
Voici le graphe des appelants de cette fonction :



4.43.3.18 void SH_InOutStateMachine ::resendText (QString text, bool editable = false) [signal]

Référencé par [addIOState\(\)](#), [addIOStateMachine\(\)](#), et [SH_ApplicationCore ::connectRunningThread\(\)](#).

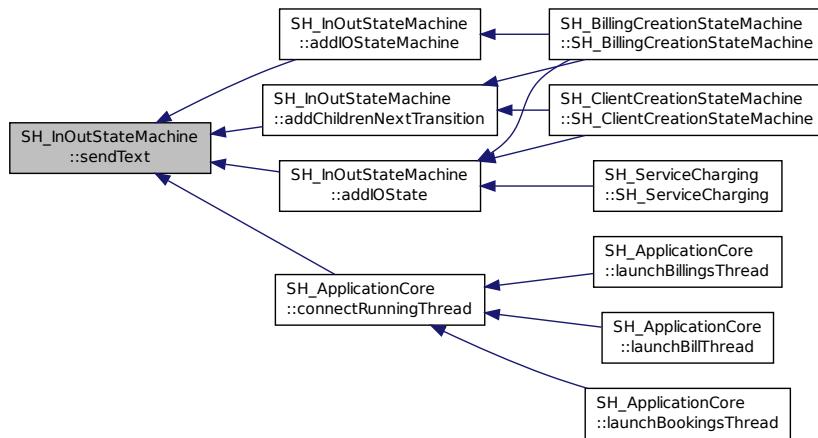
Voici le graphe des appelants de cette fonction :



4.43.3.19 void SH_InOutStateMachine ::sendText (QString text, bool editable = false) [signal]

Référencé par [addChildrenNextTransition\(\)](#), [addIOState\(\)](#), [addIOStateMachine\(\)](#), et [SH_ApplicationCore ::connectRunningThread\(\)](#).

Voici le graphe des appelants de cette fonction :



4.43.3.20 void SH_InOutStateMachine ::setContentValue (QVariant content, QString field) [slot]

Définition à la ligne 99 du fichier [SH_IOStateMachine.cpp](#).

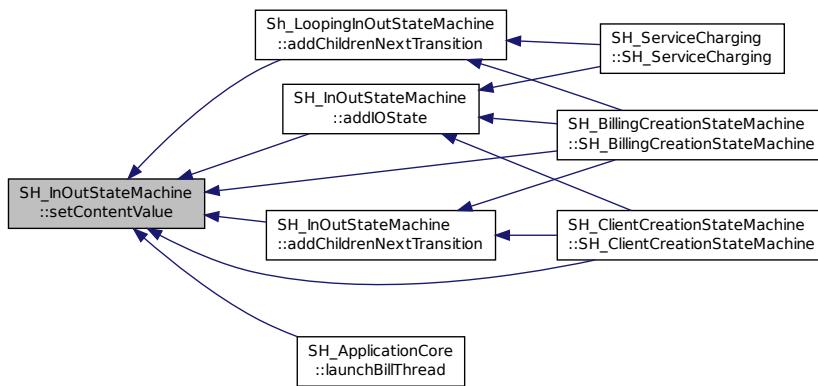
Références [m_ioContent](#).

Référencé par [Sh_LoopingInOutStateMachine ::addChildrenNextTransition\(\)](#), [addChildrenNextTransition\(\)](#), [addIOState\(\)](#), [SH_ApplicationCore ::launchBillThread\(\)](#), [SH_BillingCreationStateMachine ::SH_BillingCreationStateMachine\(\)](#), et [SH_ClientCreationStateMachine ::SH_ClientCreationStateMachine\(\)](#).

```

00100 {
00101     m_ioContent.insert(field, content);
00102 }
  
```

Voici le graphe des appelants de cette fonction :



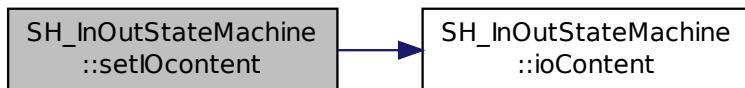
4.43.3.21 void SH_InOutStateMachine ::setIOcontent (const QVariantMap & ioContent)

Définition à la ligne 54 du fichier [SH_IOStateMachine.cpp](#).

Références [ioContent\(\)](#), et [m_ioContent](#).

```
00055 {
00056     m_ioContent = ioContent;
00057 }
```

Voici le graphe d'appel pour cette fonction :



4.43.3.22 void SH_InOutStateMachine ::setIOStateHistory (QHistoryState * state, QString field)

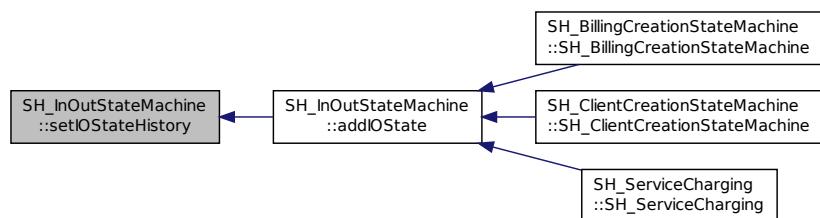
Définition à la ligne [226](#) du fichier [SH_IOStateMachine.cpp](#).

Références [m_ioStatesHistory](#).

Référencé par [addIOState\(\)](#).

```
00227 {
00228     m_ioStatesHistory.insert(field, state); /*remplacement si plusieurs fois*/
00229 }
```

Voici le graphe des appelants de cette fonction :



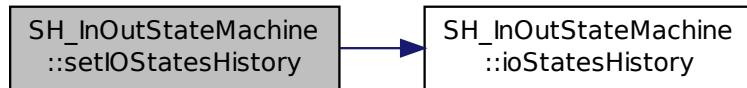
4.43.3.23 void SH_InOutStateMachine ::setIOStatesHistory (const QMap<QString, QHistoryState *> & ioStatesHistory) [private]

Définition à la ligne [214](#) du fichier [SH_IOStateMachine.cpp](#).

Références [ioStatesHistory\(\)](#), et [m_ioStatesHistory](#).

```
00215 {
00216     m_ioStatesHistory = ioStatesHistory;
00217 }
```

Voici le graphe d'appel pour cette fonction :



4.43.3.24 void SH_NamedObject ::setName (const QString & name) [virtual], [inherited]

Définition à la ligne 43 du fichier [SH_NamedObject.cpp](#).

Références [SH_NamedObject ::m_name](#), et [SH_NamedObject ::name\(\)](#).

```

00044 {
00045     m_name = name;
00046 }

```

Voici le graphe d'appel pour cette fonction :



4.43.3.25 void SH_InOutStateMachine ::setTableName (const QString & tableName)

Définition à la ligne 87 du fichier [SH_IOStateMachine.cpp](#).

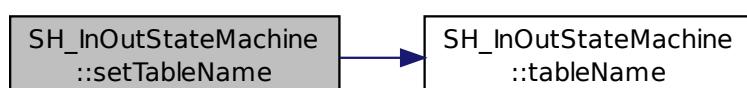
Références [m_tableName](#), et [tableName\(\)](#).

```

00088 {
00089     m_tableName = tableName;
00090 }

```

Voici le graphe d'appel pour cette fonction :



4.43.3.26 QString SH_InOutStateMachine ::tableName () const

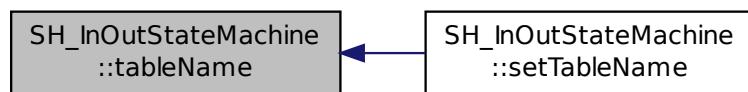
Définition à la ligne 76 du fichier [SH_IOStateMachine.cpp](#).

Références [m_tableName](#).

Référencé par [setTableName\(\)](#).

```
00077 {  
00078     return m_tableName;  
00079 }
```

Voici le graphe des appels de cette fonction :



4.43.3.27 QString SH_InOutStateMachine ::toString () [virtual]

Réimplémentée à partir de [SH_NamedObject](#).

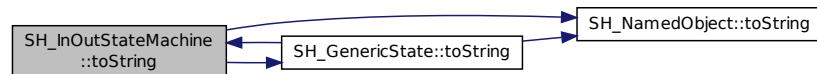
Définition à la ligne 26 du fichier [SH_IOStateMachine.cpp](#).

Références [SH_NamedObject ::toString\(\)](#), et [SH_GenericState ::toString\(\)](#).

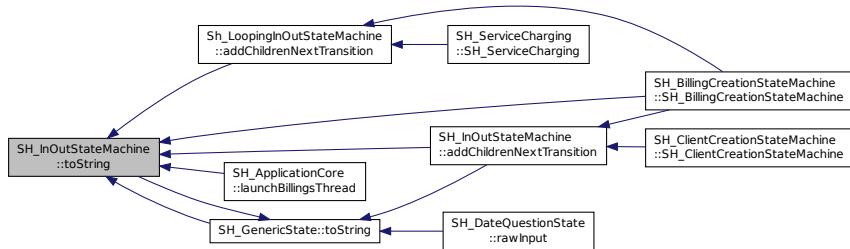
Référencé par [Sh_LoopingInOutStateMachine ::addChildsNextTransition\(\)](#), [addChildsNextTransition\(\)](#), [SH_ApplicationCore ::launchBillingsThread\(\)](#), [SH_BillingCreationStateMachine ::SH_BillingCreationStateMachine\(\)](#), et [SH_GenericState ::toString\(\)](#).

```
00027 {  
00028     QObject* parent = this->parent();  
00029     SH_GenericState* par = qobject_cast<SH_GenericState *>(parent);  
00030     if(par) {  
00031         return SH_NamedObject::toString() + " [descending from "+par->  
00032             toString()+"] ";  
00033     } else {  
00034         return SH_NamedObject::toString();  
00035     }  
00035 }
```

Voici le graphe d'appel pour cette fonction :



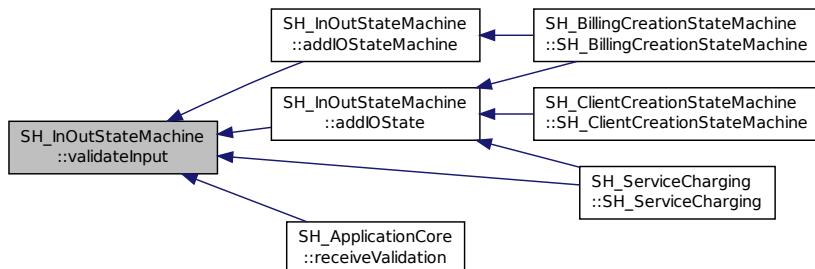
Voici le graphe des appelants de cette fonction :



4.43.3.28 void SH_InOutStateMachine ::validateInput() [signal]

Référencé par [addIOState\(\)](#), [addIOStateMachine\(\)](#), [SH_ApplicationCore ::receiveValidation\(\)](#), et [SH_ServiceCharging ::SH_ServiceCharging\(\)](#).

Voici le graphe des appelants de cette fonction :



4.43.4 Documentation des données membres

4.43.4.1 QVariantMap SH_InOutStateMachine ::m_ioContent [protected]

m_ioContent

Définition à la ligne 209 du fichier [SH_IOStateMachine.h](#).

Référencé par [Sh_LoopingInOutStateMachine ::addChildrenNextTransition\(\)](#), [addChildrenNextTransition\(\)](#), [getContentValue\(\)](#), [ioContent\(\)](#), [setContentValue\(\)](#), [setIOcontent\(\)](#), et [SH_BillingCreationStateMachine ::SH_BillingCreationStateMachine\(\)](#).

4.43.4.2 QMap<QString, QHistoryState*> SH_InOutStateMachine ::m_ioStatesHistory [protected]

m_ioStatesHistory

Définition à la ligne 217 du fichier [SH_IOStateMachine.h](#).

Référencé par [historyValue\(\)](#), [ioStatesHistory\(\)](#), [setIOStateHistory\(\)](#), et [setIOStatesHistory\(\)](#).

4.43.4.3 QString SH_InOutStateMachine ::m_tableName [protected]

m_tableName

Définition à la ligne 213 du fichier [SH_IOStateMachine.h](#).

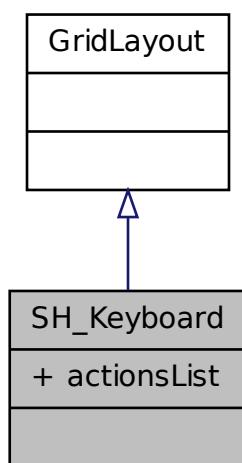
Référencé par [Sh_LoopingInOutStateMachine](#) ::addChildrenNextTransition(), addChildrenNextTransition(), setTableName(), [SH_BillingCreationStateMachine](#) ::SH_BillingCreationStateMachine(), et tableName().

La documentation de cette classe a été générée à partir des fichiers suivants :

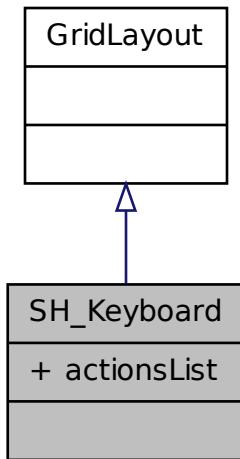
- [logic/SH_IOStateMachine.h](#)
- [logic/SH_IOStateMachine.cpp](#)

4.44 Référence de la classe SH_Keyboard

Graphe d'héritage de SH_Keyboard :



Graphe de collaboration de SH_Keyboard :



Propriétés

- var [actionsList](#)

4.44.1 Description détaillée

Définition à la ligne [4](#) du fichier [SH_Keyboard.qml](#).

4.44.2 Documentation des propriétés

4.44.2.1 var SH_Keyboard : :actionsList

Définition à la ligne [7](#) du fichier [SH_Keyboard.qml](#).

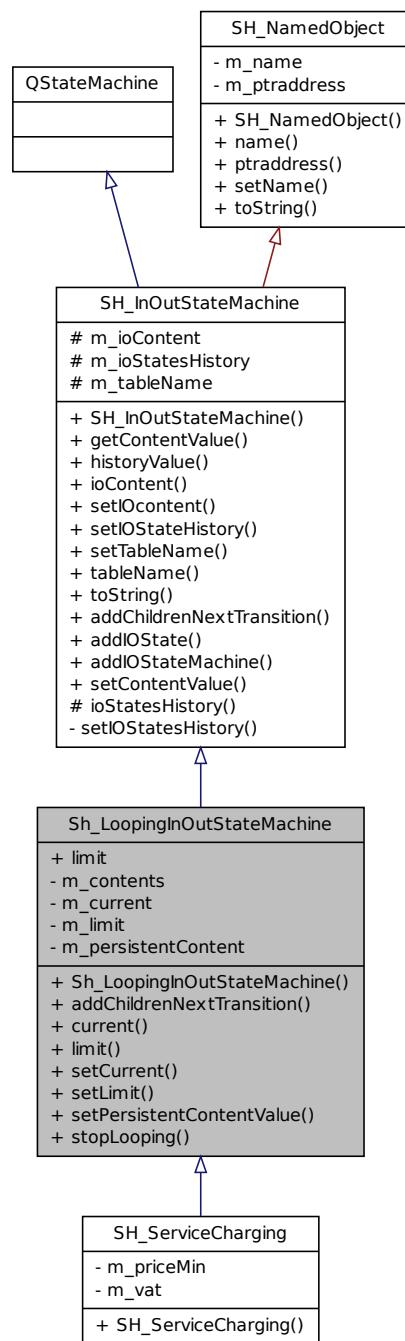
La documentation de cette classe a été générée à partir du fichier suivant :

- [views/qml/SH_Keyboard.qml](#)

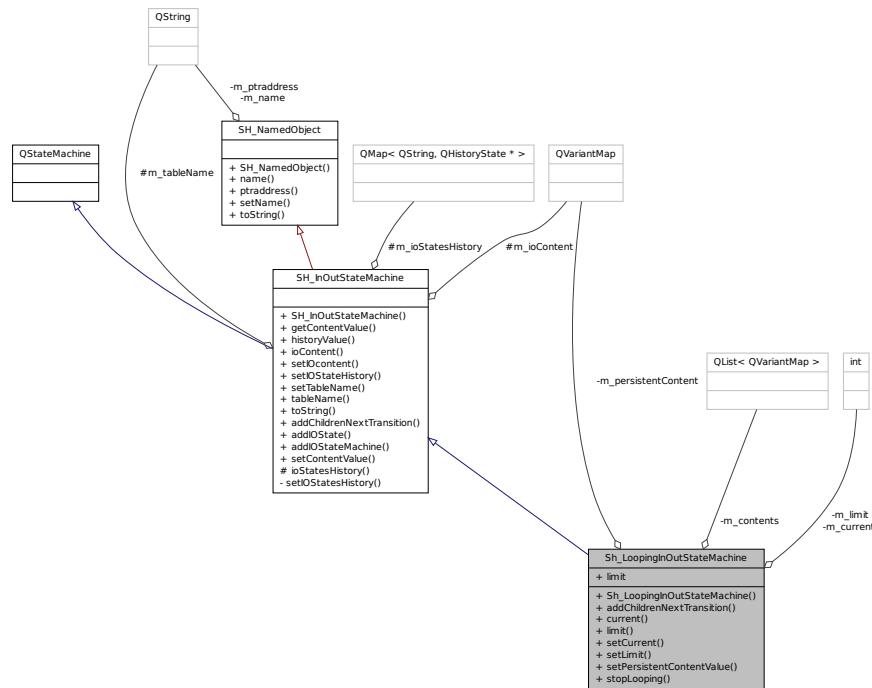
4.45 Référence de la classe Sh_LoopingInOutStateMachine

```
#include <SH_LoopingIOStateMachine.h>
```

Graphe d'héritage de Sh_LoopingInOutStateMachine :



Graphe de collaboration de Sh_LoopingInOutStateMachine :



Connecteurs publics

- void `addIOState (SH_InOutState *state, QString field)`
- void `addIOStateMachine (SH_InOutStateMachine *fsm)`
- void `setContentValue (QVariant content, QString field)`

Signaux

- void `cancelReplacement ()`
- void `clearAll ()`
- void `confirmInput ()`
- void `displayCalendar ()`
- void `displayFileDialog ()`
- void `limitChanged ()`
- void `next ()`
- void `receiveInput (QString input)`
- void `replaceInput (QString field)`
- void `resendText (QString text, bool editable=false)`
- void `sendText (QString text, bool editable=false)`
- void `validateInput ()`

Fonctions membres publiques

- `Sh_LoopingInOutStateMachine (QString tableName, QString name="looping", int limit=0, QObject *parent=0)`
- void `addChildenNextTransition (QAbstractState *previousState, QAbstractState *nextState)`
- int `current () const`
- QVariant `getContentValue (QString field)`
- QHistoryState * `historyValue (QString field)`
- QVariantMap `ioContent () const`
- int `limit () const`
- void `setCurrent (int current)`
- void `setIOcontent (const QVariantMap &ioContent)`
- void `setIOStateHistory (QHistoryState *state, QString field)`
- void `setLimit (int limit)`
- void `setPersistentContentValue (QVariant value, QString field)`
- void `setTableName (const QString &tableName)`

- void `stopLooping()`
- QString `tableName()` const
- QString `toString()`

Fonctions membres protégées

- QMap<QString, QHistoryState * > `ioStatesHistory()` const

Attributs protégés

- QVariantMap `m_ioContent`
`m_ioContent`
- QMap<QString, QHistoryState * > `m_ioStatesHistory`
`m_ioStatesHistory`
- QString `m_tableName`
`m_tableName`

Propriétés

- int `limit`

Attributs privés

- QList<QVariantMap> `m_contents`
`m_contents`
- int `m_current`
`m_current`
- int `m_limit`
`m_limit`
- QVariantMap `m_persistentContent`
`m_persistentContent`

4.45.1 Description détaillée

Définition à la ligne 10 du fichier `SH_LoopingIOStateMachine.h`.

4.45.2 Documentation des constructeurs et destructeur

4.45.2.1 `Sh_LoopingInOutStateMachine : :Sh_LoopingInOutStateMachine (QString tableName, QString name = "looping", int limit = 0, QObject * parent = 0)`

Définition à la ligne 10 du fichier `SH_LoopingIOStateMachine.cpp`.

```
00010
00011     :
00011     SH_InOutStateMachine(tableName, name, parent),
00011     m_limit(limit), m_current(-1)
00012 {
00013
00014 }
```

4.45.3 Documentation des fonctions membres

4.45.3.1 `void Sh_LoopingInOutStateMachine : :addChildsNextTransition (QAbstractState * previousState, QAbstractState * nextState)`

Définition à la ligne 86 du fichier `SH_LoopingIOStateMachine.cpp`.

Références `SH_InOutStateMachine` : `:historyValue()`, `SH_AdaptDatabaseState` : `:insertUpdate()`, `m_contents`, `m_current`, `SH_InOutStateMachine` : `:m_ioContent`, `m_limit`, `m_persistentContent`, `SH_InOutStateMachine` : `:m_tableName`, `SH_InOutStateMachine` : `:next()`, `SH_InOutStateMachine` : `:replaceInput()`, `SH_InOutStateMachine` : `:setContentValue()`, et `SH_InOutStateMachine` : `:toString()`.

Référencé par `SH_BillingCreationStateMachine` : `:SH_BillingCreationStateMachine()`, et `SH_ServiceCharging` : `:SH_ServiceCharging()`.

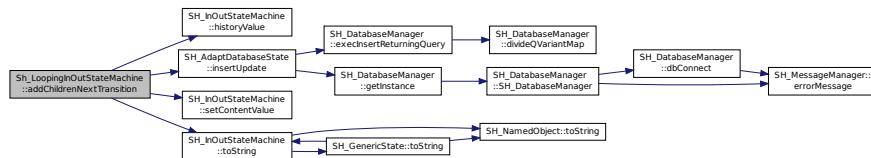
```

00087 {
00088     SH_GenericState* genPreviousState = qobject_cast<
00089         SH_GenericState*>(previousState);
00090     SH_InOutStateMachine* fsmPreviousState = qobject_cast<
00091         SH_InOutStateMachine*>(previousState);
00092     QFinalState* final = qobject_cast<QFinalState*>(nextState);
00093     if(final) {
00094         /*à faire au moment de l'entrée dans l'état previousState*/
00095         connect(previousState, &QAbstractState::entered, [=]() {
00096             m_current++;
00097             m_contents.append(m_ioContent);
00098             m_ioContent.clear();
00099             m_ioContent = m_persistentContent;
00100             if(m_limit == 0 || m_current < m_limit) {
00101                 if(genPreviousState) {
00102                     connect(genPreviousState, &QAbstractState::entered, [=]() {
00103                         genPreviousState->addTransition(genPreviousState, SIGNAL(
00104                             next()), initialState());
00105                     });
00106                 if(fsmPreviousState) {
00107                     connect(fsmPreviousState, &QAbstractState::entered, [=]() {
00108                         fsmPreviousState->addTransition(fsmPreviousState, SIGNAL(
00109                             next()), initialState());
00110                     });
00111                 } else {
00112                     SH_AdaptDatabaseState* nextSaveState = new
00113                     SH_AdaptDatabaseState("enregistrement 0 de la machine "+
00114                     toString());
00115                     if(genPreviousState) {
00116                         genPreviousState->addTransition(genPreviousState, SIGNAL(
00117                             next()), nextSaveState);
00118                     }
00119                     if(fsmPreviousState) {
00120                         fsmPreviousState->addTransition(fsmPreviousState, SIGNAL(
00121                             next()), nextSaveState);
00122                     }
00123                     if(genPreviousState || fsmPreviousState) {
00124                         for(int i = 1; i < m_limit; i++) {
00125                             SH_AdaptDatabaseState* saveState = nextSaveState;
00126                             nextSaveState = new SH_AdaptDatabaseState(QString(
00127                                 "enregistrement %1 de la machine %2").arg(QString::number(i)).arg(toString()));
00128                             saveState->addTransition(saveState, SIGNAL(next()), nextSaveState);
00129                             connect(saveState, &QAbstractState::exited, [=]() {
00130                                 connect(nextSaveState, &QAbstractState::entered, [=]() {
00131                                     setContentValue(nextSaveState->
00132                                         insertUpdate(m_tableName, m_contents[i]), "ID");
00133                                     });
00134                                 });
00135                             nextSaveState->addTransition(nextSaveState, SIGNAL(next()), final);
00136                         }
00137                     }
00138                 }
00139             }
00140             if(genPreviousState) {
00141                 /*à faire au moment de l'entrée dans l'état previousState*/
00142                 connect(genPreviousState, &QAbstractState::entered, [=]() {
00143                     connect(this, &SH_InOutStateMachine::replaceInput, [=](
00144                         QString field) {
00145                         /*après avoir demandé à revenir sur un état précédent, on attend la fin de l'état actuel
00146                         puis on retourne à l'historique de l'état désiré; celui-ci fini, on passe à l'état qui aurait du suivre celui
00147                         pendant lequel on a demandé à revenir sur un état précédent*/
00148                         QHistoryState* hState = historyValue(field);
00149                         if(hState) { /*si l'historique existe (on a déjà quitté l'état voulu)*/
00150                             hState->parentState()->addTransition(hState->parentState(), SIGNAL(
00151                                 next()), nextState);
00152                         genPreviousState->addTransition(genPreviousState, SIGNAL(
00153                             next()), hState);
00154                     });
00155                 });
00156             };
00157         };
00158     };
00159 };
00160 
```

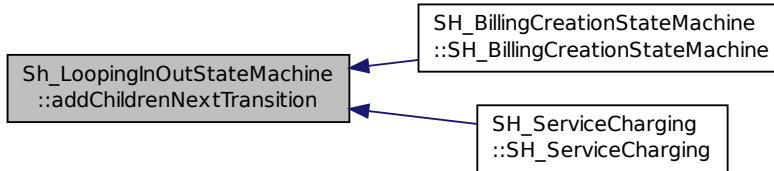
```

00149             }
00150         });
00151     });
00152 }
00153 }
```

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appels de cette fonction :



4.45.3.2 void SH_InOutStateMachine ::addIOState (SH_InOutState * state, QString field) [slot], [inherited]

Définition à la ligne 110 du fichier [SH_IOStateMachine.cpp](#).

Références [SH_ConfirmationState ::confirmInput\(\)](#), [SH_InOutState ::display\(\)](#), [SH_InOutStateMachine ::displayCalendar\(\)](#), [SH_InOutStateMachine ::displayFileDialog\(\)](#), [SH_InOutState ::output\(\)](#), [SH_InOutState ::rawInput\(\)](#), [SH_InOutStateMachine ::receiveInput\(\)](#), [SH_InOutState ::resendInput\(\)](#), [SH_InOutStateMachine ::resendText\(\)](#), [SH_InOutState ::sendOutput\(\)](#), [SH_InOutStateMachine ::sendText\(\)](#), [SH_InOutStateMachine ::setContentValue\(\)](#), [SH_InOutState ::setInput\(\)](#), [SH_InOutStateMachine ::setIOStateHistory\(\)](#), [SH_InOutStateMachine ::validateInput\(\)](#), et [SH_InOutState ::visibility\(\)](#).

Référencé par [SH_BillingCreationStateMachine ::SH_BillingCreationStateMachine\(\)](#), [SH_ClientCreationStateMachine ::SH_ClientCreationStateMachine\(\)](#), et [SH_ServiceCharging ::SH_ServiceCharging\(\)](#).

```

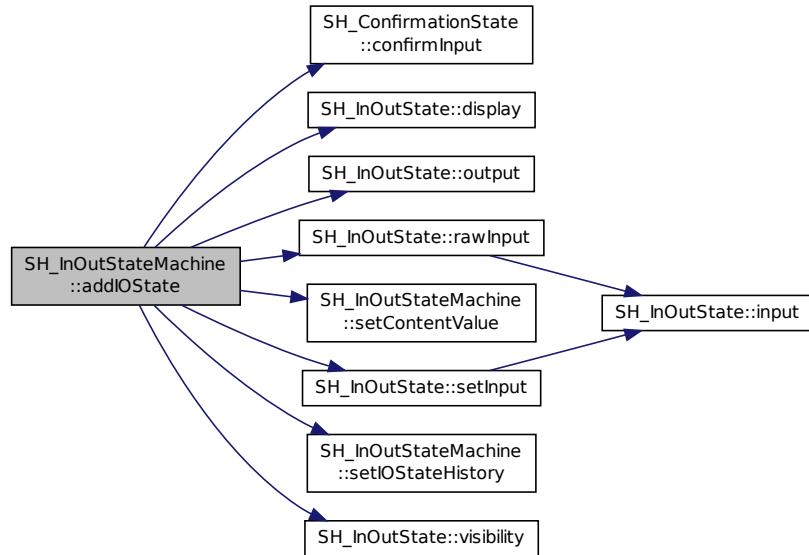
00111 {
00112     /*à faire au moment de l'entrée dans l'état state*/
00113     connect(state, &QState::entered, [=] () {
00114         qDebug() << "entered !";
00115         state->display(true);
00116         connect(this, &SH_InOutStateMachine::receiveInput, state, &
00117             SH_InOutState::setInput); /* la réception d'une valeur entraîne son enregistrement
00118             comme entrée de l'utilisateur auprès de l'état*/
00119         connect(this, &SH_InOutStateMachine::receiveInput, [=] (QString in
00120             ) { qDebug() << "hello world !"; state->setInput(in);}); /* la réception d'une valeur entraîne son
00121             enregistrement comme entrée de l'utilisateur auprès de l'état*/
00122         connect(state, &SH_InOutState::sendOutput, [=] (QVariant out) {qDebug() <<
00123             "connected !"; emit this->sendText(out.toString(), false);});
00124         connect(state, &SH_InOutState::resendInput, [=] (QVariant in) {emit this->
00125             resendText(in.toString(), true);});
00126         if(state->visibility()) {
00127             state->sendOutput(QVariant(state->output()));
00128         } else {
00129             qDebug() << "invisible";
00130         }
00131     });
00132 }
```

```

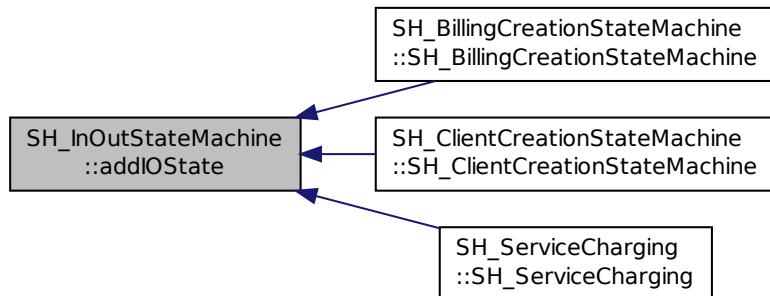
00125      });
00126      SH_ValidationState *validationState = qobject_cast<
00127          SH_ValidationState*>(state);
00128      if(validationState) {
00129          /*à faire au moment de l'entrée dans l'état state*/
00130          connect(validationState, &QState::entered, [=]() {
00131              connect(this, &SH_InOutStateMachine::validateInput,
00132                  validationState, &SH_ValidationState::confirmInput);
00133          });
00134      }
00135      SH_ConfirmationState *confirmationState = qobject_cast<
00136          SH_ConfirmationState*>(state);
00137      if(confirmationState) {
00138          /*à faire au moment de l'entrée dans l'état state*/
00139          connect(confirmationState, &QState::entered, [=]() {
00140              connect(this, &SH_InOutStateMachine::validateInput,
00141                  confirmationState, &SH_ConfirmationState::confirmInput);
00142          });
00143      }
00144      SH_DateQuestionState *dateState = qobject_cast<
00145          SH_DateQuestionState*>(state);
00146      if(dateState) {
00147          /*à faire au moment de l'entrée dans l'état state*/
00148          connect(dateState, &QState::entered, this, &
00149              SH_InOutStateMachine::displayCalendar);
00150      }
00151      /*à faire au moment de la sortie de l'état state*/
00152      connect(state, &QState::exited, [=]() {
00153          qDebug() << "exited !";
00154          if(!field.isEmpty()) {
00155              setContentValue(state->rawInput(), field);
00156              /*gestion de l'historique des états pour pouvoir revenir à l'état state après l'avoir quitté*/
00157              QHistoryState* hState = new QHistoryState(state);
00158              setIOStateHistory(hState, field);
00159          }
00160          state->disconnect(this); /*plus aucune action sur l'état ne pourra être provoquée par la machine*/
00161      });
00162
00163      QAbstractState* astate = qobject_cast<QAbstractState *>(state);
00164      if(astate) {
00165          addState(astate);
00166      }
00167  }

```

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :



4.45.3.3 void SH_InOutStateMachine ::addIOStateMachine (SH_InOutStateMachine * fsm) [slot], [inherited]

Définition à la ligne 175 du fichier [SH_IOStateMachine.cpp](#).

Références `SH_InOutStateMachine ::cancelReplacement()`, `SH_InOutStateMachine ::confirmInput()`, `SH_InOutStateMachine ::displayCalendar()`, `SH_InOutStateMachine ::receiveInput()`, `SH_InOutStateMachine ::replaceInput()`, `SH_InOutStateMachine ::resendText()`, `SH_InOutStateMachine ::sendText()`, et `SH_InOutStateMachine ::validateInput()`.

Référencé par [SH_BillingCreationStateMachine ::SH_BillingCreationStateMachine\(\)](#).

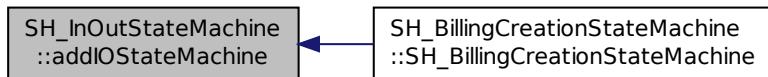
```

00176 {
00177     /*à faire au moment de l'entrée dans la machine d'état fsm*/
  
```

```

00178     connect(fsm, &QState::entered, [=]() {
00179         connect(this, &SH_InOutStateMachine::receiveInput, fsm, &
00180             SH_InOutStateMachine::receiveInput);
00180         connect(this, &SH_InOutStateMachine::sendText, fsm, &
00181             SH_InOutStateMachine::sendText);
00181         connect(this, &SH_InOutStateMachine::resendText, fsm, &
00182             SH_InOutStateMachine::resendText);
00182         connect(this, &SH_InOutStateMachine::confirmInput, fsm, &
00183             SH_InOutStateMachine::confirmInput);
00183         connect(this, &SH_InOutStateMachine::validateInput, fsm, &
00184             SH_InOutStateMachine::validateInput);
00184         connect(this, &SH_InOutStateMachine::replaceInput, fsm, &
00185             SH_InOutStateMachine::replaceInput);
00185         connect(this, &SH_InOutStateMachine::cancelReplacement, fsm,
00186             &SH_InOutStateMachine::cancelReplacement);
00186         connect(this, &SH_InOutStateMachine::displayCalendar, fsm, &
00187             SH_InOutStateMachine::displayCalendar);
00187     });
00188     /*à faire au moment de la sortie de la machine d'état fsm*/
00189     connect(fsm, &QState::exited, [=]() {
00190         fsm->disconnect(this); /*plus aucune action sur la machine d'état fille ne pourra être provoquée
00190                               par la machine mère*/
00191     });
00192 }
00193 }
```

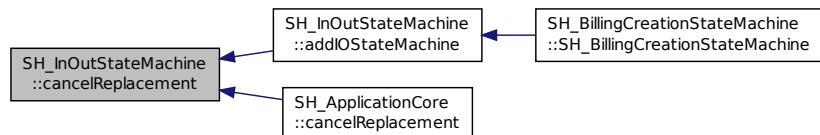
Voici le graphe des appels de cette fonction :



4.45.3.4 void SH_InOutStateMachine ::cancelReplacement() [signal], [inherited]

Référencé par [SH_InOutStateMachine ::addIOStateMachine\(\)](#), et [SH_ApplicationCore ::cancelReplacement\(\)](#).

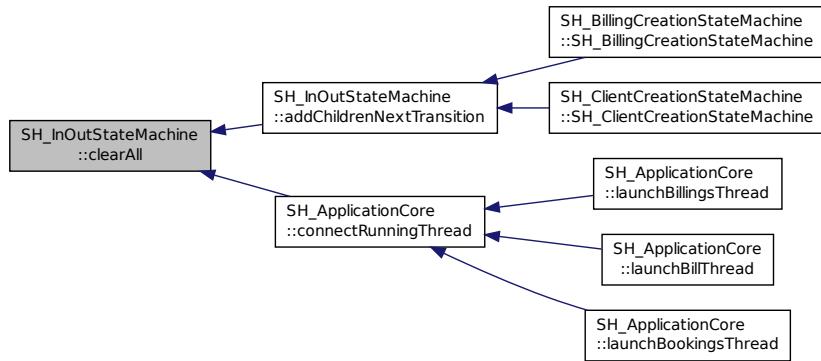
Voici le graphe des appels de cette fonction :



4.45.3.5 void SH_InOutStateMachine ::clearAll() [signal], [inherited]

Référencé par [SH_InOutStateMachine ::addChildrenNextTransition\(\)](#), et [SH_ApplicationCore ::connectRunningThread\(\)](#).

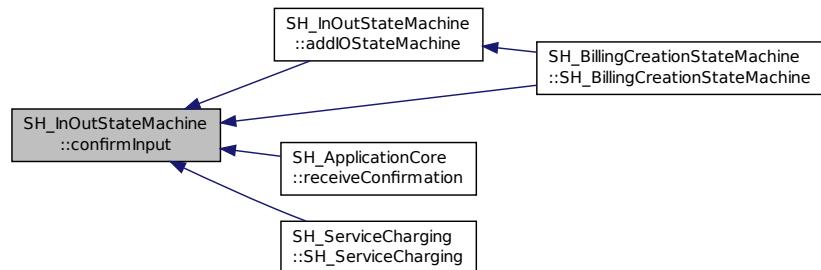
Voici le graphe des appelants de cette fonction :



4.45.3.6 void SH_InOutStateMachine ::confirmInput() [signal], [inherited]

Référencé par [SH_InOutStateMachine ::addIOStateMachine\(\)](#), [SH_ApplicationCore ::receiveConfirmation\(\)](#), [SH_BillingCreationStateMachine ::SH_BillingCreationStateMachine\(\)](#), et [SH_ServiceCharging ::SH_ServiceCharging\(\)](#).

Voici le graphe des appelants de cette fonction :



4.45.3.7 int Sh_LoopingInOutStateMachine ::current() const

Définition à la ligne 23 du fichier [SH_LoopingIOStateMachine.cpp](#).

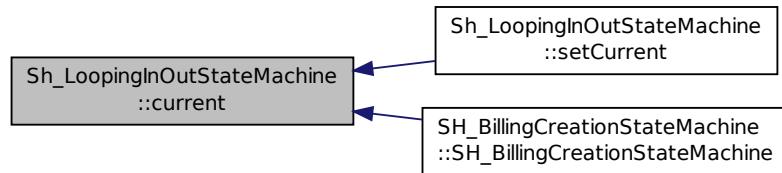
Références [m_current](#).

Référencé par [setCurrent\(\)](#), et [SH_BillingCreationStateMachine ::SH_BillingCreationStateMachine\(\)](#).

```

00024 {
00025     return m_current;
00026 }
```

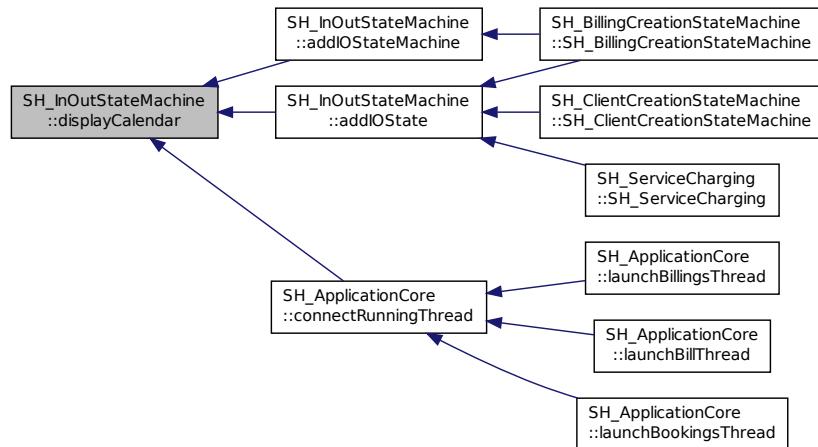
Voici le graphe des appelants de cette fonction :



4.45.3.8 void SH_InOutStateMachine ::displayCalendar() [signal], [inherited]

Référencé par [SH_InOutStateMachine ::addIOState\(\)](#), [SH_InOutStateMachine ::addIOSStateMachine\(\)](#), et [SH_ApplicationCore ::connectRunningThread\(\)](#).

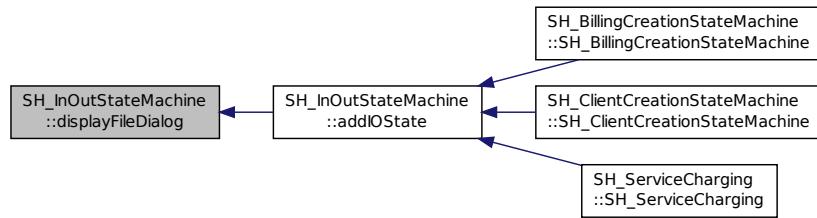
Voici le graphe des appelants de cette fonction :



4.45.3.9 void SH_InOutStateMachine ::displayFileDialog() [signal], [inherited]

Référencé par [SH_InOutStateMachine ::addIOState\(\)](#).

Voici le graphe des appelants de cette fonction :



4.45.3.10 QVariant SH_InOutStateMachine ::getContentValue (QString field) [inherited]

Définition à la ligne 65 du fichier [SH_IStateMachine.cpp](#).

Références [SH_InOutStateMachine ::m_ioContent](#).

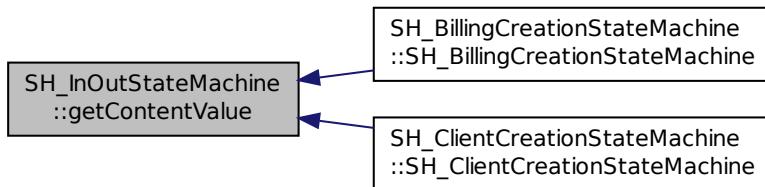
Référencé par [SH_BillingCreationStateMachine ::SH_BillingCreationStateMachine\(\)](#), et [SH_ClientCreationStateMachine ::SH_ClientCreationStateMachine\(\)](#).

```

00066 {
00067     return m_ioContent.value(field);
00068 }

```

Voici le graphe des appelants de cette fonction :



4.45.3.11 QHistoryState * SH_InOutStateMachine ::historyValue (QString field) [inherited]

Définition à la ligne 238 du fichier [SH_IStateMachine.cpp](#).

Références [SH_InOutStateMachine ::m_ioStatesHistory](#).

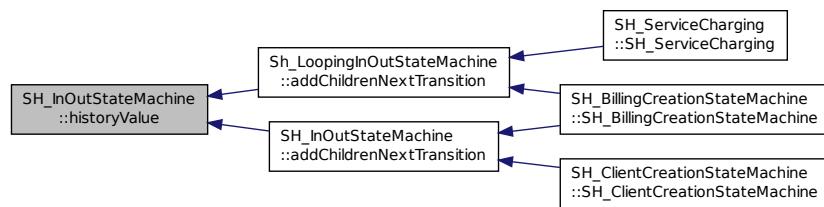
Référencé par [addChildrenNextTransition\(\)](#), et [SH_InOutStateMachine ::addChildrenNextTransition\(\)](#).

```

00239 {
00240     return m_ioStatesHistory.value(field);
00241 }

```

Voici le graphe des appels de cette fonction :



4.45.3.12 QVariantMap SH_InOutStateMachine ::ioContent () const [inherited]

Définition à la ligne 43 du fichier [SH_IOStateMachine.cpp](#).

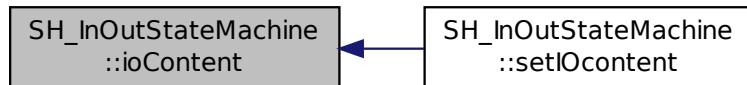
Références [SH_InOutStateMachine ::m_ioContent](#).

Référencé par [SH_InOutStateMachine ::setIOcontent\(\)](#).

```

00044 {
00045     return m_ioContent;
00046 }
```

Voici le graphe des appels de cette fonction :



4.45.3.13 QMap<QString, QHistoryState * > SH_InOutStateMachine ::ioStatesHistory () const [protected], [inherited]

Définition à la ligne 202 du fichier [SH_IOStateMachine.cpp](#).

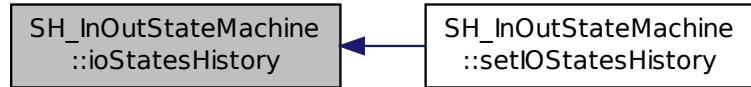
Références [SH_InOutStateMachine ::m_ioStatesHistory](#).

Référencé par [SH_InOutStateMachine ::setIOStatesHistory\(\)](#).

```

00203 {
00204     return m_ioStatesHistory;
00205 }
```

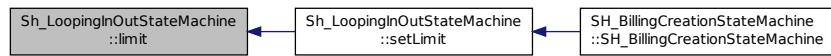
Voici le graphe des appelants de cette fonction :



4.45.3.14 int Sh_LoopingInOutStateMachine ::limit() const

Référencé par [setLimit\(\)](#).

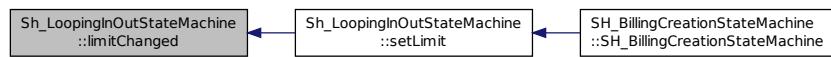
Voici le graphe des appelants de cette fonction :



4.45.3.15 void Sh_LoopingInOutStateMachine ::limitChanged() [signal]

Référencé par [setLimit\(\)](#).

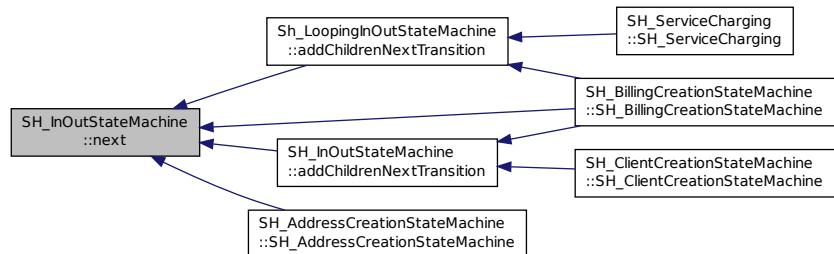
Voici le graphe des appelants de cette fonction :



4.45.3.16 void SH_InOutStateMachine ::next() [signal], [inherited]

Référencé par [addChildrenNextTransition\(\)](#), [SH_InOutStateMachine ::addChildrenNextTransition\(\)](#), [SH_AddressCreationStateMachine ::SH_AddressCreationStateMachine\(\)](#), et [SH_BillingCreationStateMachine ::SH_BillingCreationStateMachine\(\)](#).

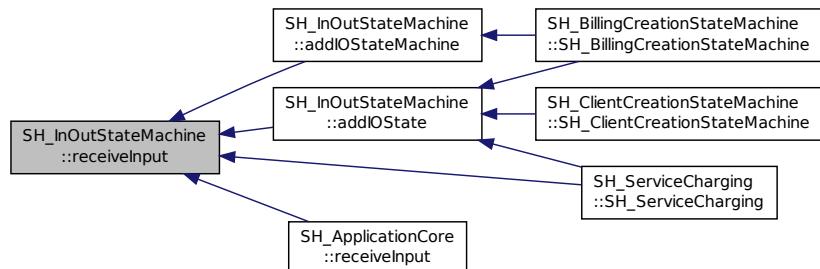
Voici le graphe des appelants de cette fonction :



4.45.3.17 void SH_InOutStateMachine ::receiveInput (QString *input*) [signal], [inherited]

Référencé par [SH_InOutStateMachine ::addIOState\(\)](#), [SH_InOutStateMachine ::addIOSubMachine\(\)](#), [SH_ApplicationCore ::receiveInput\(\)](#), et [SH_ServiceCharging ::SH_ServiceCharging\(\)](#).

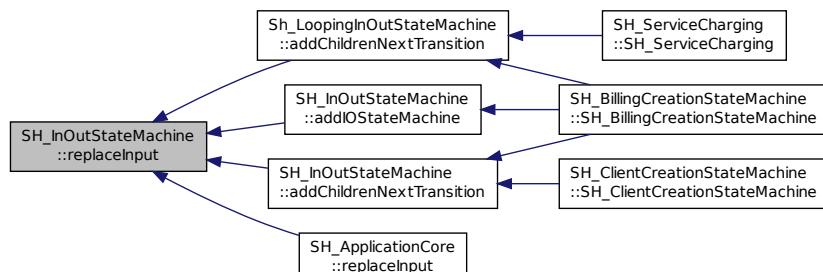
Voici le graphe des appelants de cette fonction :



4.45.3.18 void SH_InOutStateMachine ::replaceInput (QString *field*) [signal], [inherited]

Référencé par [addChildrenNextTransition\(\)](#), [SH_InOutStateMachine ::addChildrenNextTransition\(\)](#), [SH_InOutStateMachine ::addIOSubMachine\(\)](#), et [SH_ApplicationCore ::replaceInput\(\)](#).

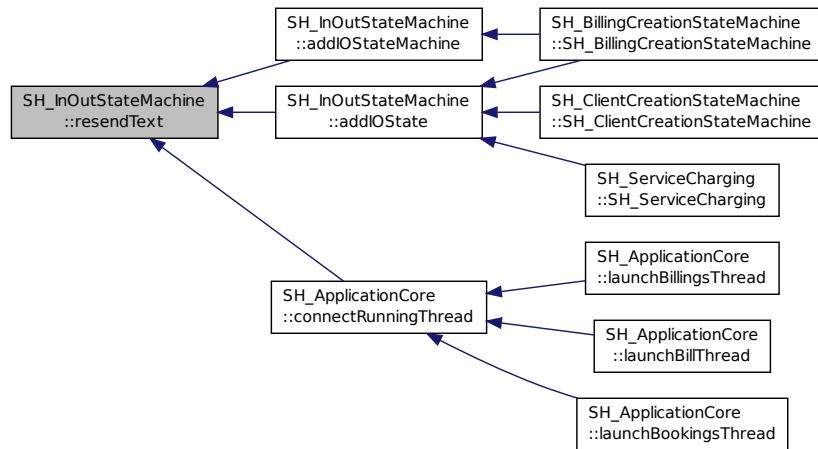
Voici le graphe des appelants de cette fonction :



4.45.3.19 void SH_InOutStateMachine ::resendText (QString *text*, bool *editable* = false) [signal], [inherited]

Référencé par [SH_InOutStateMachine ::addIOState\(\)](#), [SH_InOutStateMachine ::addIOStateMachine\(\)](#), et [SH_ApplicationCore ::connectRunningThread\(\)](#).

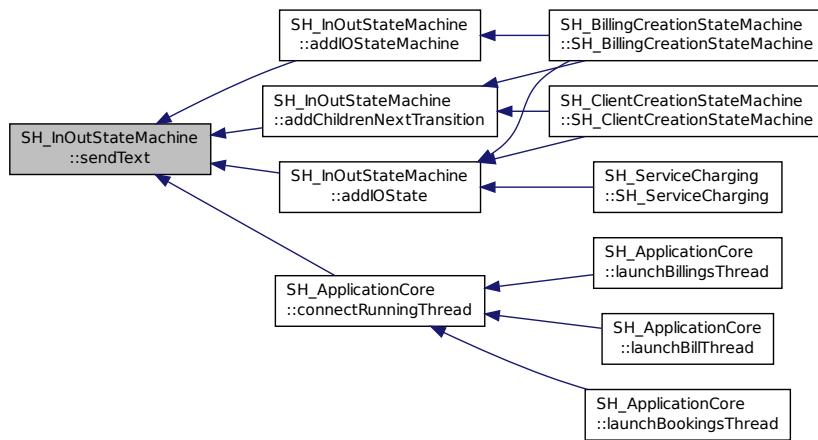
Voici le graphe des appelants de cette fonction :



4.45.3.20 void SH_InOutStateMachine ::sendText (QString *text*, bool *editable* = false) [signal], [inherited]

Référencé par [SH_InOutStateMachine ::addChildrenNextTransition\(\)](#), [SH_InOutStateMachine ::addIOState\(\)](#), [SH_InOutStateMachine ::addIOStateMachine\(\)](#), et [SH_ApplicationCore ::connectRunningThread\(\)](#).

Voici le graphe des appelants de cette fonction :



4.45.3.21 void SH_InOutStateMachine ::setContentValue (QVariant *content*, QString *field*) [slot], [inherited]

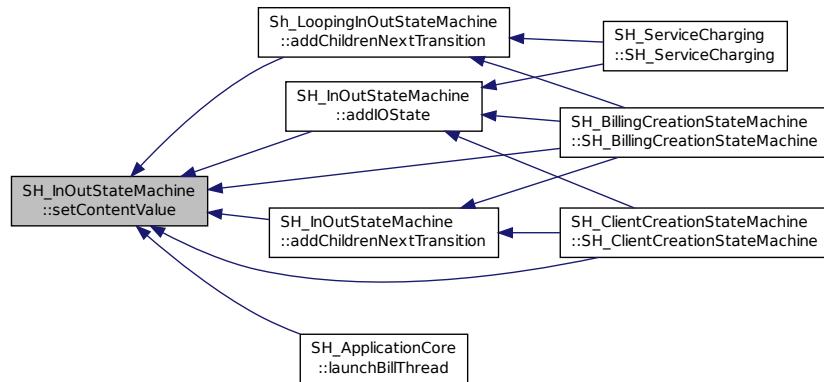
Définition à la ligne 99 du fichier [SH_IOStateMachine.cpp](#).

Références [SH_InOutStateMachine ::m_ioContent](#).

Référencé par [addChildrenNextTransition\(\)](#), [SH_InOutStateMachine ::addChildrenNextTransition\(\)](#), [SH_InOutStateMachine ::addIOState\(\)](#), [SH_ApplicationCore ::launchBillThread\(\)](#), [SH_BillingCreationStateMachine ::SH_BillingCreationStateMachine\(\)](#), et [SH_ClientCreationStateMachine ::SH_ClientCreationStateMachine\(\)](#).

```
00100 {
00101     m_ioContent.insert(field, content);
00102 }
```

Voici le graphe des appels de cette fonction :



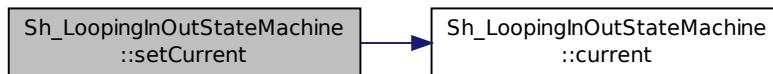
4.45.3.22 void Sh_LoopingInOutStateMachine ::setCurrent (int current)

Définition à la ligne 34 du fichier [SH_LoopingIOStateMachine.cpp](#).

Références [current\(\)](#), et [m_current](#).

```
00035 {
00036     m_current = current;
00037 }
```

Voici le graphe d'appel pour cette fonction :



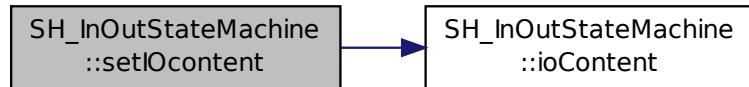
4.45.3.23 void SH_InOutStateMachine ::setIOcontent (const QVariantMap & ioContent) [inherited]

Définition à la ligne 54 du fichier [SH_IOStateMachine.cpp](#).

Références [SH_InOutStateMachine ::ioContent\(\)](#), et [SH_InOutStateMachine ::m_ioContent](#).

```
00055 {
00056     m_ioContent = ioContent;
00057 }
```

Voici le graphe d'appel pour cette fonction :



4.45.3.24 void SH_InOutStateMachine ::setIOStateHistory (QHistoryState * state, QString field) [inherited]

Définition à la ligne 226 du fichier [SH_IOStateMachine.cpp](#).

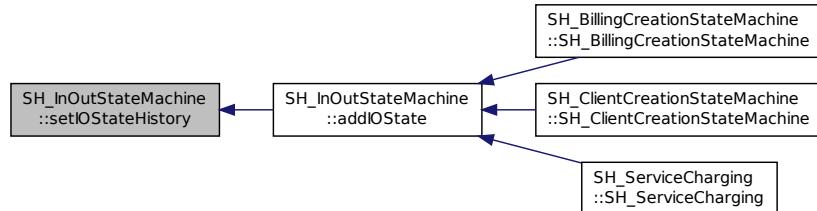
Références [SH_InOutStateMachine ::m_ioStatesHistory](#).

Référencé par [SH_InOutStateMachine ::addIOState\(\)](#).

```

00227 {
00228     m_ioStatesHistory.insert(field, state); /*remplacement si plusieurs fois*/
00229 }
  
```

Voici le graphe des appels de cette fonction :



4.45.3.25 void Sh_LoopingInOutStateMachine ::setLimit (int limit)

Définition à la ligne 61 du fichier [SH_LoopingIOStateMachine.cpp](#).

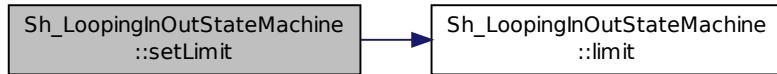
Références [limit\(\)](#), [limitChanged\(\)](#), et [m_limit](#).

Référencé par [SH_BillingCreationStateMachine ::SH_BillingCreationStateMachine\(\)](#).

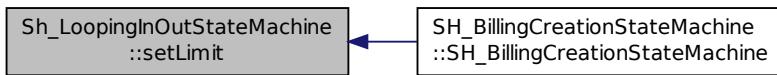
```

00062 {
00063     m_limit = limit;
00064     emit limitChanged();
00065 }
  
```

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appels de cette fonction :



4.45.3.26 void Sh_LoopingInOutStateMachine ::setPersistentContentValue (QVariant value, QString field)

Définition à la ligne 39 du fichier [SH_LoopingIOStateMachine.cpp](#).

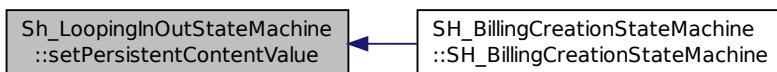
Références [m_persistentContent](#).

Référencé par [SH_BillingCreationStateMachine ::SH_BillingCreationStateMachine\(\)](#).

```

00040 {
00041     m_persistentContent.insert(field, value);
00042 }
  
```

Voici le graphe des appels de cette fonction :



4.45.3.27 void SH_InOutStateMachine ::setTableName (const QString & tableName) [inherited]

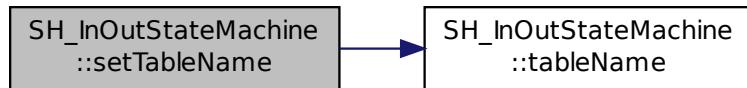
Définition à la ligne 87 du fichier [SH_IOStateMachine.cpp](#).

Références [SH_InOutStateMachine ::m_tableName](#), et [SH_InOutStateMachine ::tableName\(\)](#).

```

00088 {
00089     m_tableName = tableName;
00090 }
  
```

Voici le graphe d'appel pour cette fonction :



4.45.3.28 void Sh_LoopingInOutStateMachine ::stopLooping()

Définition à la ligne 72 du fichier [SH_LoopingIOStateMachine.cpp](#).

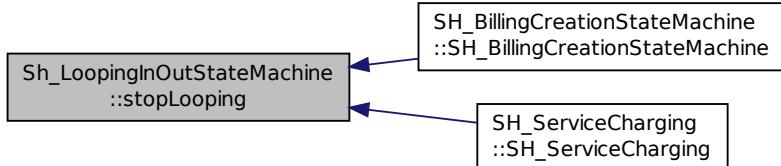
Références [m_current](#), et [m_limit](#).

Référencé par [SH_BillingCreationStateMachine ::SH_BillingCreationStateMachine\(\)](#), et [SH_ServiceCharging ::SH_ServiceCharging\(\)](#).

```

00072
00073     if(m_limit == 0) {
00074         m_limit = m_current + 1;
00075     } else {
00076         m_current = m_limit - 1;
00077     }
00078 }
```

Voici le graphe des appels de cette fonction :



4.45.3.29 QString SH_InOutStateMachine ::tableName() const [inherited]

Définition à la ligne 76 du fichier [SH_IOStateMachine.cpp](#).

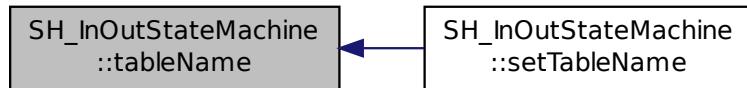
Références [SH_InOutStateMachine ::m_tableName](#).

Référencé par [SH_InOutStateMachine ::setTableName\(\)](#).

```

00077 {
00078     return m_tableName;
00079 }
```

Voici le graphe des appelants de cette fonction :



4.45.3.30 QString SH_InOutStateMachine ::toString() [virtual], [inherited]

Réimplémentée à partir de [SH_NamedObject](#).

Définition à la ligne 26 du fichier [SH_IOSStateMachine.cpp](#).

Références [SH_NamedObject ::toString\(\)](#), et [SH_GenericState ::toString\(\)](#).

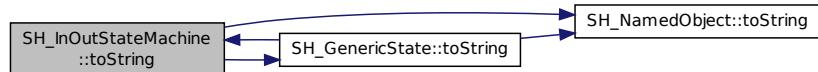
Référencé par [addChildrenNextTransition\(\)](#), [SH_InOutStateMachine ::addChildrenNextTransition\(\)](#), [SH_ApplicationCore ::launchBillingsThread\(\)](#), [SH_BillingCreationStateMachine ::SH_BillingCreationStateMachine\(\)](#), et [SH_GenericState ::toString\(\)](#).

```

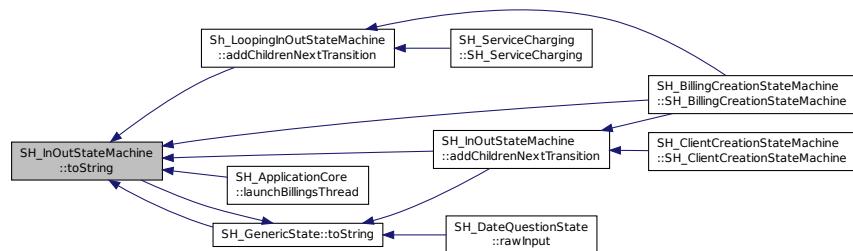
00027 {
00028     QObject* parent = this->parent();
00029     SH_GenericState* par = qobject_cast<SH_GenericState *>(parent);
00030     if(par) {
00031         return SH_NamedObject::toString() + " [descending from "+par->
00032             toString()+"] ";
00033     } else {
00034         return SH_NamedObject::toString();
00035     }

```

Voici le graphe d'appel pour cette fonction :



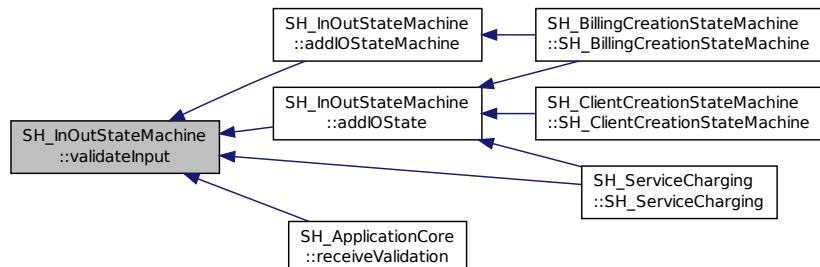
Voici le graphe des appelants de cette fonction :



4.45.3.31 void SH_InOutStateMachine ::validateInput() [signal], [inherited]

Référencé par [SH_InOutStateMachine ::addIOState\(\)](#), [SH_InOutStateMachine ::addIOStateMachine\(\)](#), [SH_ApplicationCore ::receiveValidation\(\)](#), et [SH_ServiceCharging ::SH_ServiceCharging\(\)](#).

Voici le graphe des appelants de cette fonction :



4.45.4 Documentation des données membres

4.45.4.1 QList<QVariantMap> Sh_LoopingInOutStateMachine ::m_contents [private]

`m_contents`

Définition à la ligne 99 du fichier [SH_LoopingIOStateMachine.h](#).

Référencé par [addChildrenNextTransition\(\)](#).

4.45.4.2 int Sh_LoopingInOutStateMachine ::m_current [private]

`m_current`

Définition à la ligne 95 du fichier [SH_LoopingIOStateMachine.h](#).

Référencé par [addChildrenNextTransition\(\)](#), [current\(\)](#), [setCurrent\(\)](#), et [stopLooping\(\)](#).

4.45.4.3 QVariantMap SH_InOutStateMachine ::m_ioContent [protected], [inherited]

`m_ioContent`

Définition à la ligne 209 du fichier [SH_IOStateMachine.h](#).

Référencé par [addChildrenNextTransition\(\)](#), [SH_InOutStateMachine ::addChildrenNextTransition\(\)](#), [SH_InOutStateMachine ::getContentValue\(\)](#), [SH_InOutStateMachine ::ioContent\(\)](#), [SH_InOutStateMachine ::setContentValue\(\)](#), [SH_InOutStateMachine ::setIOcontent\(\)](#), et [SH_BillingCreationStateMachine ::SH_BillingCreationStateMachine\(\)](#).

4.45.4.4 QMap<QString, QHistoryState*> SH_InOutStateMachine ::m_ioStatesHistory [protected], [inherited]

`m_ioStatesHistory`

Définition à la ligne 217 du fichier [SH_IOStateMachine.h](#).

Référencé par [SH_InOutStateMachine ::historyValue\(\)](#), [SH_InOutStateMachine ::ioStatesHistory\(\)](#), [SH_InOutStateMachine ::setIOStateHistory\(\)](#), et [SH_InOutStateMachine ::setIOStatesHistory\(\)](#).

4.45.4.5 int Sh_LoopingInOutStateMachine ::m_limit [private]

m_limit

Définition à la ligne 91 du fichier [SH_LoopingIOStateMachine.h](#).

Référencé par [addChildsNextTransition\(\)](#), [setLimit\(\)](#), et [stopLooping\(\)](#).

4.45.4.6 QVariantMap Sh_LoopingInOutStateMachine ::m_persistentContent [private]

m_persistentContent

Définition à la ligne 103 du fichier [SH_LoopingIOStateMachine.h](#).

Référencé par [addChildsNextTransition\(\)](#), et [setPersistentContentValue\(\)](#).

4.45.4.7 QString SH_InOutStateMachine ::m_tableName [protected], [inherited]

m_tableName

Définition à la ligne 213 du fichier [SH_IOStateMachine.h](#).

Référencé par [addChildsNextTransition\(\)](#), [SH_InOutStateMachine ::addChildsNextTransition\(\)](#), [SH_InOutStateMachine ::setTableName\(\)](#), [SH_BillingCreationStateMachine ::SH_BillingCreationStateMachine\(\)](#), et [SH_InOutStateMachine ::tableName\(\)](#).

4.45.5 Documentation des propriétés

4.45.5.1 int Sh_LoopingInOutStateMachine ::limit [read], [write]

Définition à la ligne 13 du fichier [SH_LoopingIOStateMachine.h](#).

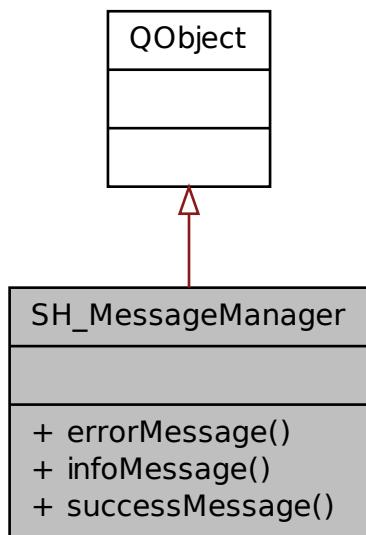
La documentation de cette classe a été générée à partir des fichiers suivants :

- logic/[SH_LoopingIOStateMachine.h](#)
- logic/[SH_LoopingIOStateMachine.cpp](#)

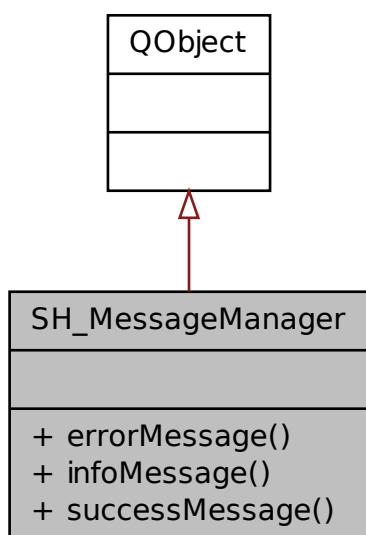
4.46 Référence de la classe SH_MessageManager

```
#include <SH_MessageManager.h>
```

Graphe d'héritage de SH_MessageManager :



Graphe de collaboration de SH_MessageManager :



Types publics

- enum **ErrorMode** {
 ERROR, **TEST**, **DEBUG**, **DEBUG_VERBOSE**,
RELEASE }

Fonctions membres publiques statiques

- static void **errorMessage** (QString message, QString title="Erreur")
- static void **infoMessage** (QString message, QString title="Info")
- static void **successMessage** (QString message, QString title="Réussite")

4.46.1 Description détaillée

Définition à la ligne 11 du fichier [SH_MessageManager.h](#).

4.46.2 Documentation des énumérations membres

4.46.2.1 enum SH_MessageManager : :ErrorMode

Valeurs énumérées

ERROR

TEST

DEBUG

DEBUG_VERBOSE

RELEASE

Définition à la ligne 23 du fichier [SH_MessageManager.h](#).

```
00023 { ERROR, TEST, DEBUG, DEBUG_VERBOSE, RELEASE };
```

4.46.3 Documentation des fonctions membres

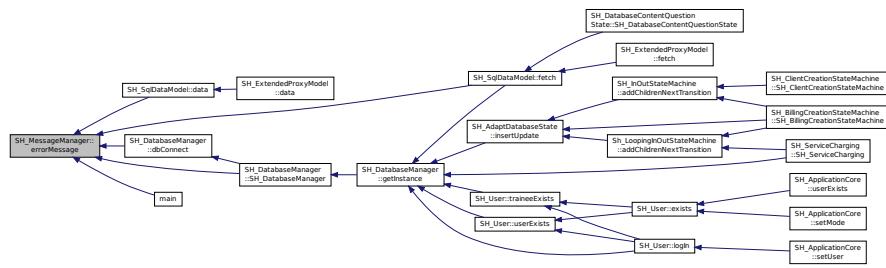
4.46.3.1 void SH_MessageManager : :errorMessage (QString message, QString title = "Erreur") [static]

Définition à la ligne 16 du fichier [SH_MessageManager.cpp](#).

Référencé par [SH_SqlDataManager](#) : [:data\(\)](#), [SH_DatabaseManager](#) : [:dbConnect\(\)](#), [SH_SqlDataManager](#) : [:fetch\(\)](#), [main\(\)](#), et [SH_DatabaseManager](#) : [:SH_DatabaseManager\(\)](#).

```
00017 {
00018     if(!message.isEmpty()) {
00019 #ifdef DEBUG
00020         qWarning() << QObject::tr("%L1 : %L2").arg(title).arg(message);
00021 #else
00022         QMessageBox::critical(0,QObject::tr("%L1").arg(title),QObject::tr("%L2").arg(message));
00023 #endif
00024     }
00025 }
```

Voici le graphe des appelants de cette fonction :



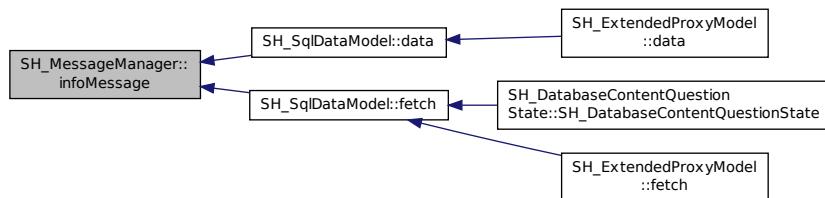
4.46.3.2 void SH_MessageManager : :infoMessage (QString message, QString title = "Info") [static]

Définition à la ligne 54 du fichier [SH_MessageManager.cpp](#).

Référencé par [SH_SqlDataModel](#) : `:data()`, et [SH_SqlDataModel](#) : `:fetch()`.

```
00055 {  
00056     if (!message.isEmpty() ) {  
00057         /*switch(errorMode) {  
  
00058             case DEBUG:/*/  
00059                 qDebug() << QObject::tr("%L1 : %L2").arg(title).arg(message);  
00060                 /*break;  
  
00061             case TEST:  
  
00062             case RELEASE:  
  
00063                 QMessageBox::information(0,QObject::tr("%L1").arg(title),QObject::tr("%L2").arg(message));  
  
00064             default:  
  
00065                 console.log(title+" : " + message);  
  
00066             }*/  
00067         }  
00068 }
```

Voici le graphe des appels de cette fonction :



4.46.3.3 void SH_MessageManager : successMessage (QString message, QString title = "Réussite") [static]

Définition à la ligne 33 du fichier [SH_MessageManager.cpp](#).

```
00034 {  
00035     if (!message.isEmpty()) {  
00036         /*switch(errorMode) {
```

```
00037     case DEBUG:*/
00038         qDebug() << QObject::tr("%L1 : %L2").arg(title).arg(message);
00039         /*break;
00040
00041     case TEST:
00042
00043     case RELEASE:
00044
00045     default:
00046
00047     {
00048         QMessageBox::information(0,QObject::tr("%L1").arg(title),QObject::tr("%L2").arg(message));
00049
00050         console.log(title+" : " + message);
00051
00052     } */
00053 }
```

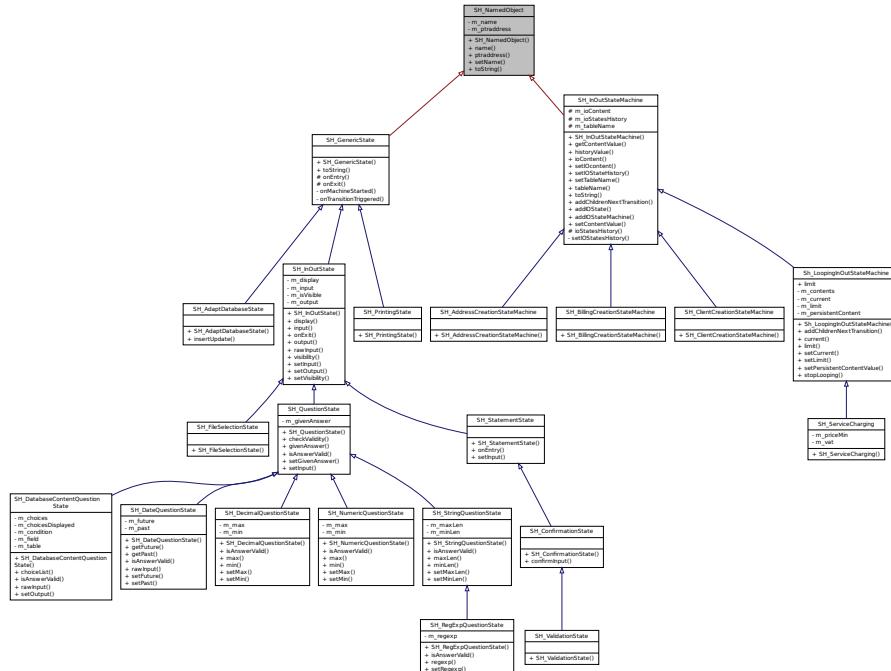
La documentation de cette classe a été générée à partir des fichiers suivants :

- [SH_MessageManager.h](#)
 - [SH_MessageManager.cpp](#)

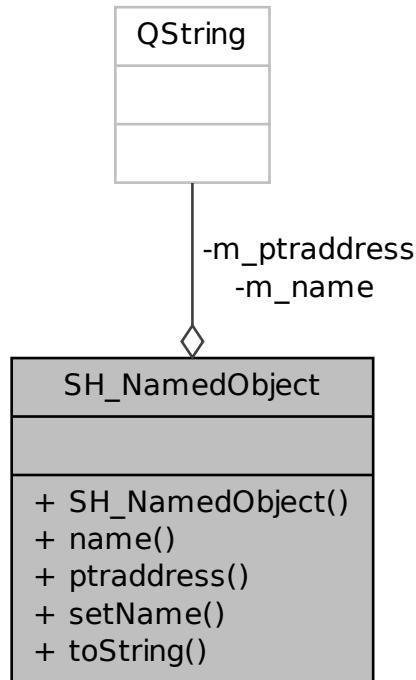
4.47 Référence de la classe SH_NamedObject

```
#include <SH_NamedObject.h>
```

Graphe d'héritage de SH_NamedObject :



Graphe de collaboration de SH_NamedObject :



Fonctions membres publiques

- `SH_NamedObject (QString name)`
- `virtual QString name () const`
- `QString ptraddress () const`
- `virtual void setName (const QString &name)`
- `virtual QString toString ()`

Attributs privés

- `QString m_name`
 `m_name`
- `QString m_ptraddress`
 `m_ptraddress`

4.47.1 Description détaillée

Définition à la ligne 10 du fichier [SH_NamedObject.h](#).

4.47.2 Documentation des constructeurs et destructeur

4.47.2.1 `SH_NamedObject ::SH_NamedObject (QString name)`

Paramètres

<i>name</i>	
-------------	--

Définition à la ligne 9 du fichier [SH_NamedObject.cpp](#).

```
00009      :
00010      m_name (name)
00011  {
00012  /*m_ptraddress = QString(&this);*/
00013 }
```

4.47.3 Documentation des fonctions membres

4.47.3.1 `QString SH_NamedObject::name () const [virtual]`

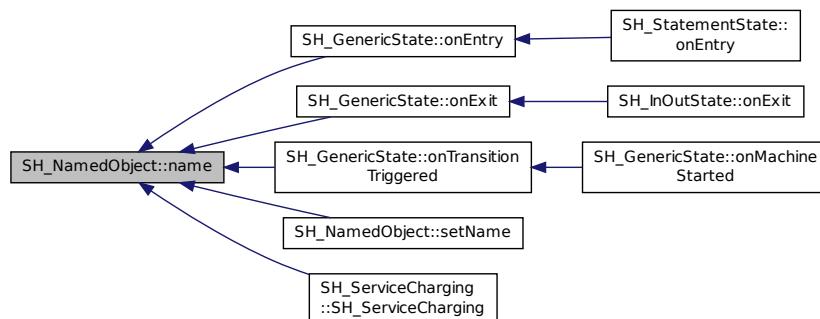
Définition à la ligne 32 du fichier [SH_NamedObject.cpp](#).

Références [m_name](#).

Référencé par [SH_GenericState::onEntry\(\)](#), [SH_GenericState::onExit\(\)](#), [SH_GenericState::onTransitionTriggered\(\)](#), [setName\(\)](#), et [SH_ServiceCharging::SH_ServiceCharging\(\)](#).

```
00033 {
00034     return m_name;
00035 }
```

Voici le graphe des appels de cette fonction :



4.47.3.2 `QString SH_NamedObject::ptraddress () const`

Définition à la ligne 54 du fichier [SH_NamedObject.cpp](#).

Références [m_ptraddress](#).

```
00055 {
00056     return m_ptraddress;
00057 }
```

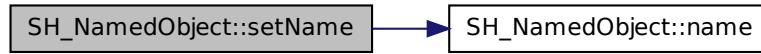
4.47.3.3 `void SH_NamedObject::setName (const QString & name) [virtual]`

Définition à la ligne 43 du fichier [SH_NamedObject.cpp](#).

Références [m_name](#), et [name\(\)](#).

```
00044 {
00045     m_name = name;
00046 }
```

Voici le graphe d'appel pour cette fonction :



4.47.3.4 QString SH_NamedObject ::toString() [virtual]

Réimplémentée dans [SH_InOutStateMachine](#), et [SH_GenericState](#).

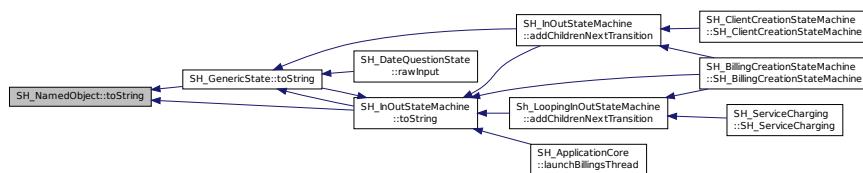
Définition à la ligne 21 du fichier [SH_NamedObject.cpp](#).

Références [m_name](#).

Référencé par [SH_GenericState ::toString\(\)](#), et [SH_InOutStateMachine ::toString\(\)](#).

```
00022 {
00023     return this->m_name;
00024 }
```

Voici le graphe des appels de cette fonction :



4.47.4 Documentation des données membres

4.47.4.1 QString SH_NamedObject ::m_name [private]

[m_name](#)

Définition à la ligne 55 du fichier [SH_NamedObject.h](#).

Référencé par [name\(\)](#), [setName\(\)](#), et [toString\(\)](#).

4.47.4.2 QString SH_NamedObject ::m_ptraddress [private]

[m_ptraddress](#)

Définition à la ligne 59 du fichier [SH_NamedObject.h](#).

Référencé par [ptraddress\(\)](#).

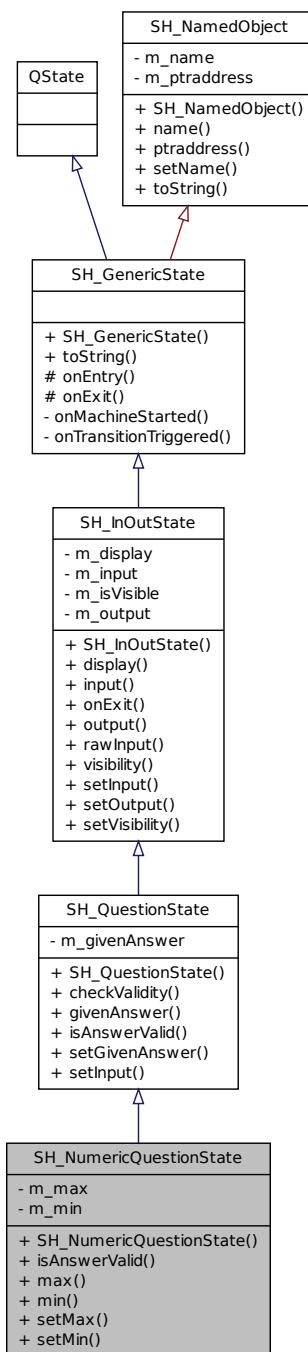
La documentation de cette classe a été générée à partir des fichiers suivants :

- logic/[SH_NamedObject.h](#)
- logic/[SH_NamedObject.cpp](#)

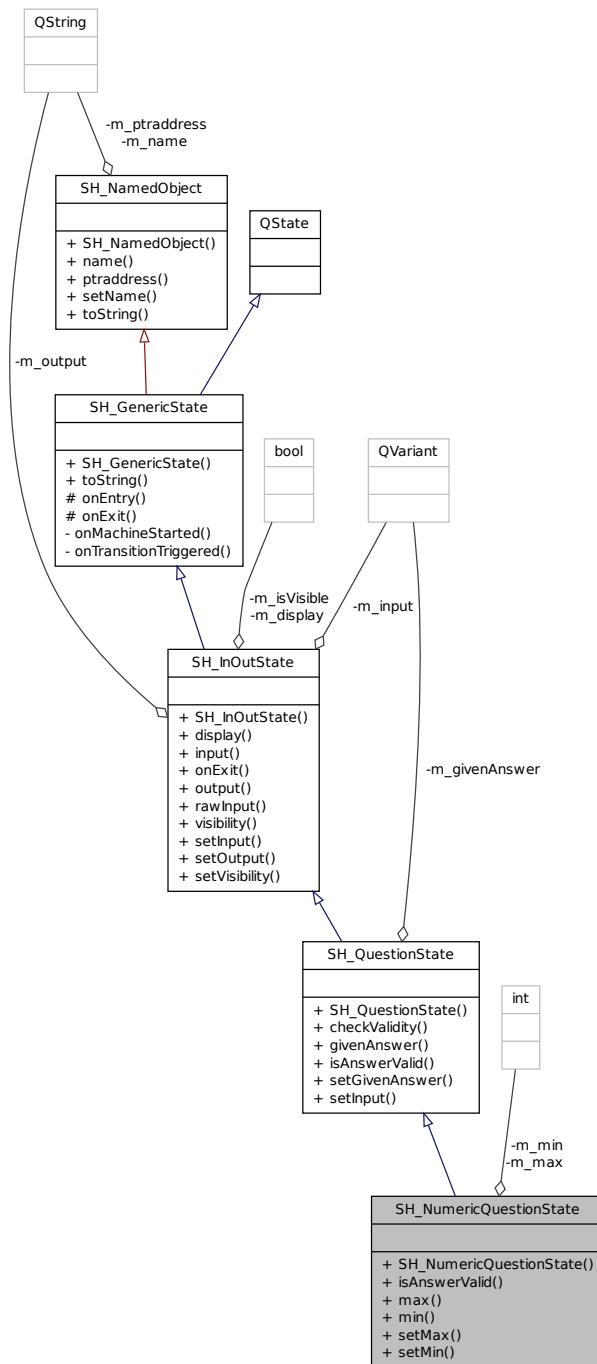
4.48 Référence de la classe SH_NumericQuestionState

```
#include <SH_NumericQuestionState.h>
```

Graphe d'héritage de SH_NumericQuestionState :



Graphe de collaboration de SH_NumericQuestionState :



Connecteurs publics

- virtual void `setOutput` (const `QString` &`output`)
- virtual void `setVisibility` (bool `isVisible`)

Signaux

- void `answerInvalid` ()

```

    answerInvalid
- void answerValid ()
    answerValid
- void next ()
- void resendInput (QVariant input)
- void sendOutput (QVariant output)

```

Fonctions membres publiques

```

- SH_NumericQuestionState (QString question, QString name, int min=0, int max=-1, QState *parent=0)
- bool checkValidity ()
- void display (bool canDisplay)
- virtual QVariant givenAnswer () const
- virtual QVariant input () const
- virtual bool isAnswerValid (const QVariant &givenAnswer)
- int max () const
- int min () const
- void onExit (QEvent *event)
- virtual QString output () const
- virtual QVariant rawInput () const
- virtual void setGivenAnswer (const QVariant &givenAnswer)
- virtual void setInput (const QVariant &input)
- void setMax (int max)
- void setMin (int min)
- QString toString ()
- bool visibility ()

```

Fonctions membres protégées

```
- void onEntry (QEvent *event)
```

Attributs privés

```

- int m_max
    m_max
- int m_min
    m_min

```

4.48.1 Description détaillée

Définition à la ligne 10 du fichier [SH_NumericQuestionState.h](#).

4.48.2 Documentation des constructeurs et destructeur

4.48.2.1 SH_NumericQuestionState : :SH_NumericQuestionState (QString *question*, QString *name*, int *min* = 0, int *max* = -1, QState * *parent* = 0)

Paramètres

<i>question</i>	
<i>name</i>	
<i>min</i>	
<i>max</i>	
<i>parent</i>	

Définition à la ligne 10 du fichier [SH_NumericQuestionState.cpp](#).

```

00010
:
00011     SH_QuestionState(question, name, parent), m_min(min),
    m_max(max)
00012 {
00013

```

```
00014 }
```

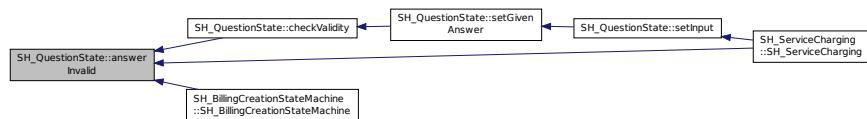
4.48.3 Documentation des fonctions membres

4.48.3.1 void SH_QuestionState ::answerInvalid() [signal], [inherited]

answerInvalid

Référencé par [SH_QuestionState ::checkValidity\(\)](#), [SH_BillingCreationStateMachine ::SH_BillingCreationStateMachine\(\)](#), et [SH_ServiceCharging ::SH_ServiceCharging\(\)](#).

Voici le graphe des appelants de cette fonction :

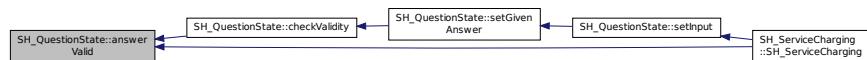


4.48.3.2 void SH_QuestionState ::answerValid() [signal], [inherited]

answerValid

Référencé par [SH_QuestionState ::checkValidity\(\)](#), et [SH_ServiceCharging ::SH_ServiceCharging\(\)](#).

Voici le graphe des appelants de cette fonction :



4.48.3.3 bool SH_QuestionState ::checkValidity() [inherited]

Définition à la ligne 20 du fichier [SH_QuestionState.cpp](#).

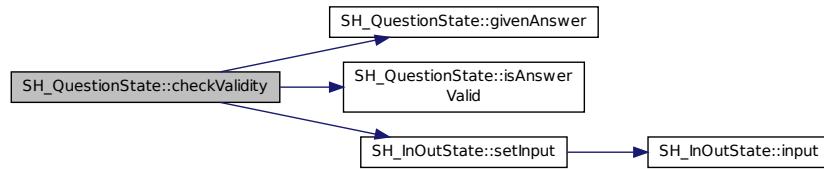
Références [SH_QuestionState ::answerInvalid\(\)](#), [SH_QuestionState ::answerValid\(\)](#), [SH_QuestionState ::givenAnswer\(\)](#), [SH_QuestionState ::isValid\(\)](#), [SH_GenericState ::next\(\)](#), et [SH_InOutState ::setInput\(\)](#).

Référencé par [SH_QuestionState ::setGivenAnswer\(\)](#).

```

00021 {
00022     bool ok = this->isValid(this->givenAnswer());
00023     if(ok) {
00024         qDebug() << this->givenAnswer() << " answer valid !";
00025         qDebug() << this->givenAnswer();
00026         SH_InOutState::setInput(this->givenAnswer());
00027         emit answerValid();
00028         emit next();
00029     } else {
00030         qDebug() << this->givenAnswer() << " answer invalid :-(";
00031         qDebug() << this->givenAnswer();
00032         emit answerInvalid();
00033     }
00034     return ok;
00035 }
```

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appels de cette fonction :



4.48.3.4 void SH_InOutState ::display (bool canDisplay) [inherited]

Définition à la ligne 95 du fichier [SH_IOState.cpp](#).

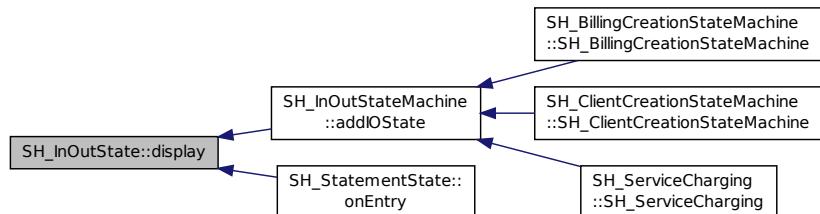
Références [SH_InOutState ::m_display](#), [SH_InOutState ::m_isVisible](#), [SH_InOutState ::m_output](#), et [SH_InOutState ::sendOutput\(\)](#).

Référencé par [SH_InOutStateMachine ::addIOState\(\)](#), et [SH_StatementState ::onEntry\(\)](#).

```

00096 {
00097     m_display=canDisplay;
00098     if(m_display && !m_output.isEmpty() && m_isVisible) {
00099         qDebug() << "resalut !" << QVariant(m_output);
00100         emit sendOutput(QVariant(m_output));
00101     }
00102 }
```

Voici le graphe des appels de cette fonction :



4.48.3.5 QVariant SH_QuestionState ::givenAnswer () const [virtual], [inherited]

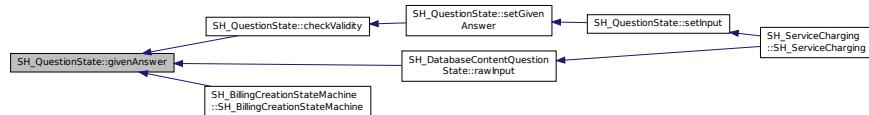
Définition à la ligne 55 du fichier [SH_QuestionState.cpp](#).

Références [SH_QuestionState ::m_givenAnswer](#).

Référencé par [SH_QuestionState ::checkValidity\(\)](#), [SH_DatabaseContentQuestionState ::rawInput\(\)](#), et [SH_BillingCreationStateMachine ::SH_BillingCreationStateMachine\(\)](#).

```
00056 {
00057     return this->m_givenAnswer;
00058 }
```

Voici le graphe des appels de cette fonction :



4.48.3.6 QVariant SH_InOutState::input() const [virtual], [inherited]

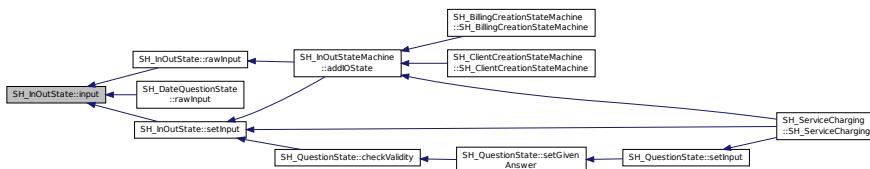
Définition à la ligne 20 du fichier [SH_IOState.cpp](#).

Références [SH_InOutState::m_input](#).

Référencé par [SH_InOutState::rawInput\(\)](#), [SH_DateQuestionState::rawInput\(\)](#), et [SH_InOutState::setInput\(\)](#).

```
00021 {
00022     return m_input;
00023 }
```

Voici le graphe des appels de cette fonction :



4.48.3.7 bool SH_NumericQuestionState::isValid (const QVariant & givenAnswer) [virtual]

Implémente [SH_QuestionState](#).

Définition à la ligne 22 du fichier [SH_NumericQuestionState.cpp](#).

Références [m_max](#), et [m_min](#).

```
00023 {
00024     qDebug() << "is answer valid";
00025     bool ok;
00026     int answer = givenAnswer.toInt(&ok);
00027     if(ok) {
00028         return ((m_max <= m_min || answer <= m_max) && answer >=
m_min);
00029     } else {
00030         return false;
00031     }
00032 }
```

4.48.3.8 int SH_NumericQuestionState::max() const

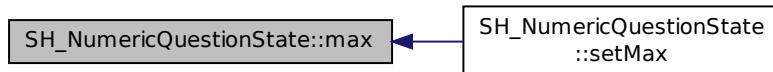
Définition à la ligne 61 du fichier [SH_NumericQuestionState.cpp](#).

Références [m_max](#).

Référencé par [setMax\(\)](#).

```
00062 {
00063     return m_max;
00064 }
```

Voici le graphe des appelants de cette fonction :



4.48.3.9 int SH_NumericQuestionState ::min () const

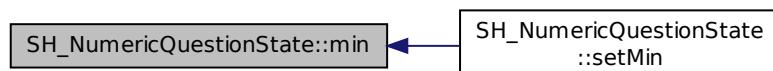
Définition à la ligne 40 du fichier [SH_NumericQuestionState.cpp](#).

Références [m_min](#).

Référencé par [setMin\(\)](#).

```
00041 {
00042     return m_min;
00043 }
```

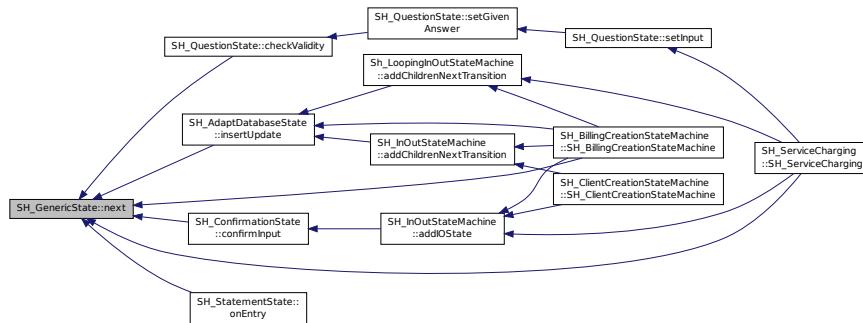
Voici le graphe des appelants de cette fonction :



4.48.3.10 void SH_GenericState ::next () [signal], [inherited]

Référencé par [SH_QuestionState ::checkValidity\(\)](#), [SH_ConfirmationState ::confirmInput\(\)](#), [SH_AdaptDatabaseState ::insertUpdate\(\)](#), [SH_StatementState ::onEntry\(\)](#), [SH_BillingCreationStateMachine ::SH_BillingCreationStateMachine\(\)](#), et [SH_ServiceCharging ::SH_ServiceCharging\(\)](#).

Voici le graphe des appelants de cette fonction :



4.48.3.11 void SH_GenericState ::onEntry (QEvent * event) [protected], [inherited]

Définition à la ligne 62 du fichier [SH_GenericDebugableState.cpp](#).

Références [SH_NamedObject ::name\(\)](#).

Référencé par [SH_StatementState ::onEntry\(\)](#).

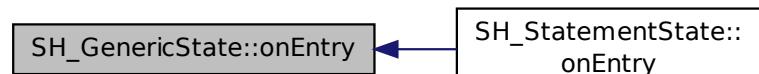
```

00063 {
00064     Q_UNUSED(event);
00065     qDebug() << "Machine: " << machine()->objectName() << " entered " << name();
00066 }
  
```

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :



4.48.3.12 void SH_InOutState ::onExit (QEvent * event) [inherited]

Définition à la ligne 110 du fichier [SH_IOState.cpp](#).

Références [SH_InOutState ::m_input](#), [SH_InOutState ::m_isVisible](#), [SH_GenericState ::onExit\(\)](#), et [SH_InOutState ::resendInput\(\)](#).

```

00111 {
00112     SH_GenericState::onExit(event);
00113     if(m_isVisible) {
00114         emit resendInput(m_input);
00115     }
00116 }
```

Voici le graphe d'appel pour cette fonction :



4.48.3.13 QString SH_InOutState ::output() const [virtual], [inherited]

Définition à la ligne 56 du fichier [SH_IOState.cpp](#).

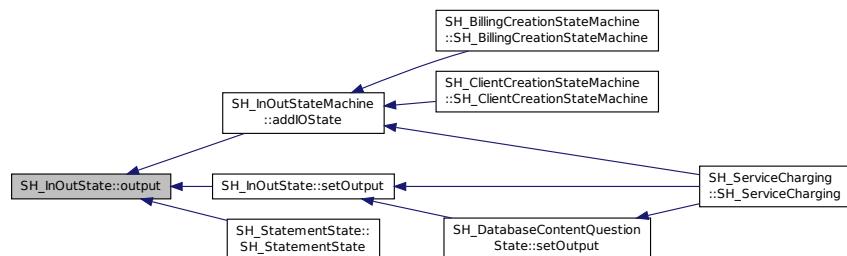
Références [SH_InOutState ::m_output](#).

Référencé par [SH_InOutStateMachine ::addIOState\(\)](#), [SH_InOutState ::setOutput\(\)](#), et [SH_StatementState ::SH_StatementState\(\)](#).

```

00057 {
00058     return m_output;
00059 }
```

Voici le graphe des appels de cette fonction :



4.48.3.14 QVariant SH_InOutState ::rawInput() const [virtual], [inherited]

Réimplémentée dans [SH_DateQuestionState](#), et [SH_DatabaseContentQuestionState](#).

Définition à la ligne 30 du fichier [SH_IOState.cpp](#).

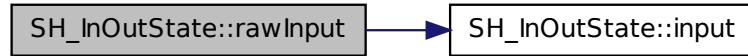
Références [SH_InOutState ::input\(\)](#).

Référencé par [SH_InOutStateMachine ::addIOState\(\)](#).

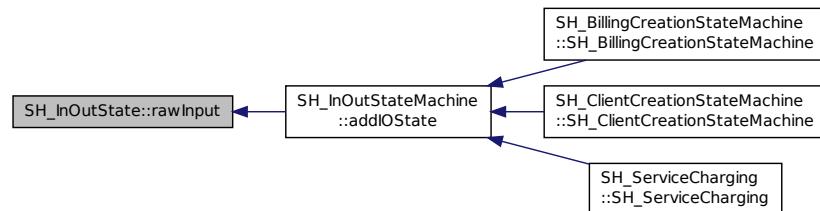
```

00031 {
00032     return input();
00033 }
```

Voici le graphe d'appel pour cette fonction :



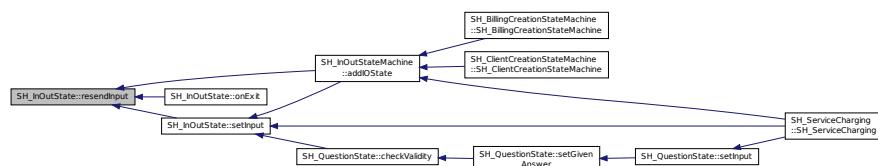
Voici le graphe des appelants de cette fonction :



4.48.3.15 void SH_InOutState ::resendInput (QVariant *input*) [signal], [inherited]

Référencé par [SH_InOutStateMachine ::addIOState\(\)](#), [SH_InOutState ::onExit\(\)](#), et [SH_InOutState ::setInput\(\)](#).

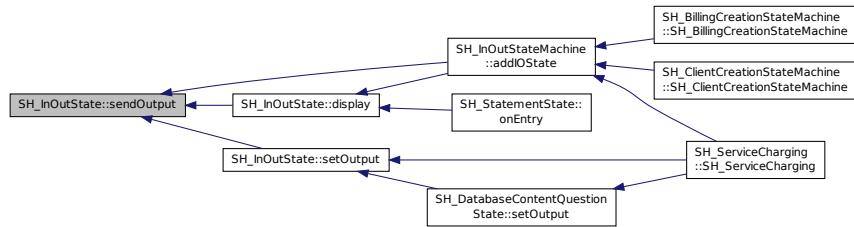
Voici le graphe des appelants de cette fonction :



4.48.3.16 void SH_InOutState ::sendOutput (QVariant *output*) [signal], [inherited]

Référencé par [SH_InOutStateMachine ::addIOState\(\)](#), [SH_InOutState ::display\(\)](#), et [SH_InOutState ::setOutput\(\)](#).

Voici le graphe des appelants de cette fonction :



4.48.3.17 void SH_QuestionState : :setGivenAnswer (const QVariant & givenAnswer) [virtual], [inherited]

Définition à la ligne 66 du fichier [SH_QuestionState.cpp](#).

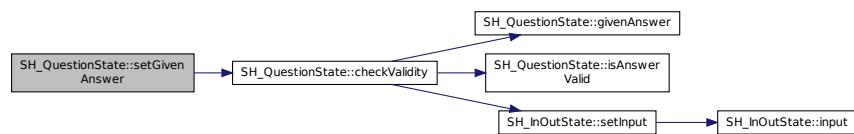
Références [SH_QuestionState : :checkValidity\(\)](#), et [SH_QuestionState : :m_givenAnswer](#).

Référencé par [SH_QuestionState : :setInput\(\)](#).

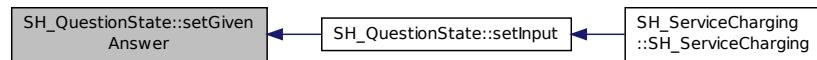
```

00067 {
00068     this->m_givenAnswer = givenAsnwer;
00069     this->checkValidity();
00070 }
  
```

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :



4.48.3.18 void SH_QuestionState : :setInput (const QVariant & input) [virtual], [inherited]

Réimplémentée à partir de [SH_InOutState](#).

Définition à la ligne 43 du fichier [SH_QuestionState.cpp](#).

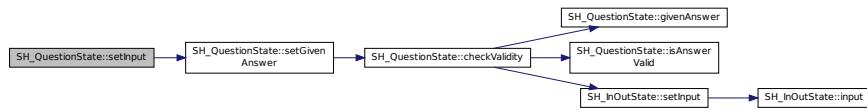
Références [SH_QuestionState : :setGivenAnswer\(\)](#).

Référencé par [SH_ServiceCharging : :SH_ServiceCharging\(\)](#).

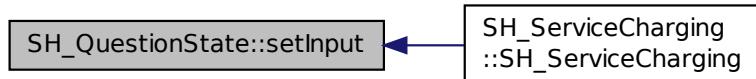
```

00044 {
00045     qDebug() << "new answer " << input.toString();
00046     this->setGivenAnswer(input);
00047 }
  
```

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :



4.48.3.19 void SH_NumericQuestionState ::setMax (int max)

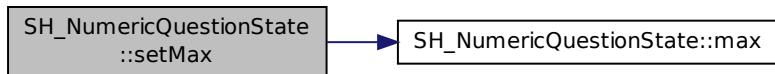
Définition à la ligne 72 du fichier [SH_NumericQuestionState.cpp](#).

Références [m_max](#), et [max\(\)](#).

```

00073 {
00074     m_max = max;
00075 }
```

Voici le graphe d'appel pour cette fonction :



4.48.3.20 void SH_NumericQuestionState ::setMin (int min)

Paramètres

<i>min</i>	
------------	--

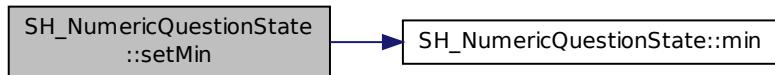
Définition à la ligne 50 du fichier [SH_NumericQuestionState.cpp](#).

Références [m_min](#), et [min\(\)](#).

```

00051 {
00052     m_min = min;
00053 }
```

Voici le graphe d'appel pour cette fonction :



4.48.3.21 void SH_InOutState : :setOutput (const QString & output) [virtual], [slot], [inherited]

Réimplémentée dans [SH_DatabaseContentQuestionState](#).

Définition à la ligne 68 du fichier [SH_IOState.cpp](#).

Références [SH_InOutState : :m_isVisible](#), [SH_InOutState : :m_output](#), [SH_InOutState : :output\(\)](#), et [SH_InOutState : :sendOutput\(\)](#).

Référencé par [SH_DatabaseContentQuestionState : :setOutput\(\)](#), et [SH_ServiceCharging : :SH_ServiceCharging\(\)](#).

```

00069 {
00070     m_output = output;
00071     if(m_isVisible) {
00072         emit sendOutput(QVariant(m_output));
00073     }
00074 }
```

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appels de cette fonction :



4.48.3.22 void SH_InOutState : :setVisibility (bool isVisible) [virtual], [slot], [inherited]

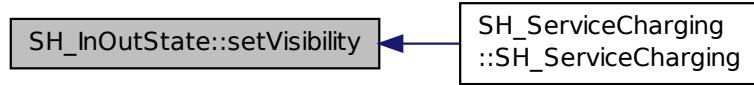
Définition à la ligne 81 du fichier [SH_IOState.cpp](#).

Références [SH_InOutState : :m_isVisible](#).

Référencé par [SH_ServiceCharging : :SH_ServiceCharging\(\)](#).

```
00082 {
00083     m_isVisible = isVisible;
00084 }
```

Voici le graphe des appels de cette fonction :



4.48.3.23 QString SH_GenericState : toString() [virtual], [inherited]

Réimplémentée à partir de [SH_NamedObject](#).

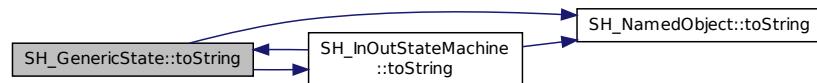
Définition à la ligne 21 du fichier [SH_GenericDebugableState.cpp](#).

Références [SH_NamedObject : toString\(\)](#), et [SH_InOutStateMachine : toString\(\)](#).

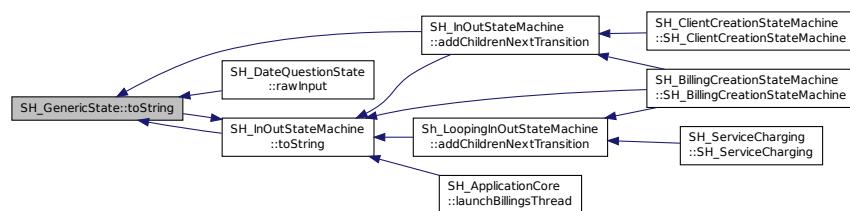
Référencé par [SH_InOutStateMachine : addChildrenNextTransition\(\)](#), [SH_DateQuestionState : rawInput\(\)](#), et [SH_InOutStateMachine : toString\(\)](#).

```
00022 {
00023     QStateMachine* machine = this->machine();
00024     SH_InOutStateMachine* mach = qobject_cast<
00025         SH_InOutStateMachine *>(machine);
00026     if(mach) {
00027         return SH_NamedObject::toString() + " [in "+mach->
00028             toString()+"] ";
00029     } else {
00030         return SH_NamedObject::toString();
00031     }
00030 }
```

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appels de cette fonction :



4.48.3.24 bool SH_InOutState ::visibility() [inherited]

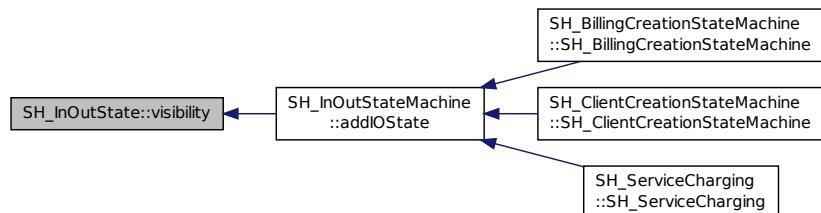
Définition à la ligne 91 du fichier [SH_IOState.cpp](#).

Références [SH_InOutState ::m_isVisible](#).

Référencé par [SH_InOutStateMachine ::addIOState\(\)](#).

```
00091     {
00092         return m_isVisible;
00093     }
```

Voici le graphe des appels de cette fonction :



4.48.4 Documentation des données membres

4.48.4.1 int SH_NumericQuestionState ::m_max [private]

m_max

Définition à la ligne 76 du fichier [SH_NumericQuestionState.h](#).

Référencé par [isAnswerValid\(\)](#), [max\(\)](#), et [setMax\(\)](#).

4.48.4.2 int SH_NumericQuestionState ::m_min [private]

m_min

Définition à la ligne 72 du fichier [SH_NumericQuestionState.h](#).

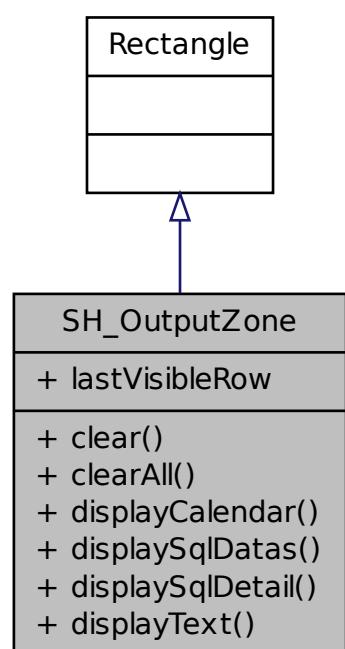
Référencé par [isAnswerValid\(\)](#), [min\(\)](#), et [setMin\(\)](#).

La documentation de cette classe a été générée à partir des fichiers suivants :

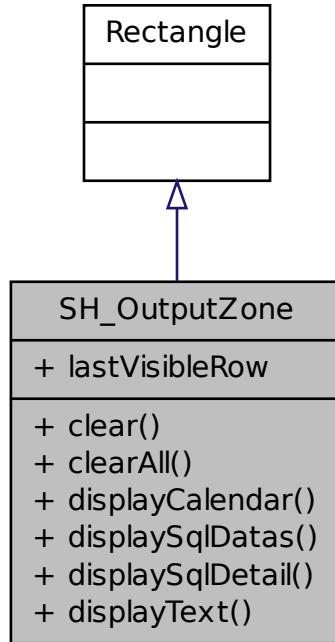
- logic/[SH_NumericQuestionState.h](#)
- logic/[SH_NumericQuestionState.cpp](#)

4.49 Référence de la classe SH_OutputZone

Graphe d'héritage de SH_OutputZone :



Graphe de collaboration de SH_OutputZone :



Signaux

- void **display** (string text)
- void **displayNew** (string text, bool editable)
- void **displayNewFixed** (string text)
- void **replace** (string text)
- void **selected** (string selectedItem)

Fonctions membres publiques

- void **clear** (row)
- void **clearAll** ()
- void **displayCalendar** ()
- void **displaySqlDatas** (sqlData, sqlDelegate)
- void **displaySqlDetail** (sqlData)
- void **displayText** (text)

Propriétés

- int **lastVisibleRow**

4.49.1 Description détaillée

Définition à la ligne 4 du fichier [SH_OutputZone.qml](#).

4.49.2 Documentation des fonctions membres

4.49.2.1 void SH_OutputZone ::clear (row)

4.49.2.2 void SH_OutputZone ::clearAll ()

4.49.2.3 void SH_OutputZone ::display (string *text*) [signal]

4.49.2.4 void SH_OutputZone ::displayCalendar ()

4.49.2.5 void SH_OutputZone ::displayNew (string *text*, bool *editable*) [signal]

4.49.2.6 void SH_OutputZone ::displayNewFixed (string *text*) [signal]

4.49.2.7 void SH_OutputZone ::displaySqlDatas (sqlData , sqlDelegate)

4.49.2.8 void SH_OutputZone ::displaySqlDetail (sqlData)

4.49.2.9 void SH_OutputZone ::displayText (text)

4.49.2.10 void SH_OutputZone ::replace (string *text*) [signal]

4.49.2.11 void SH_OutputZone ::selected (string *selectedItem*) [signal]

4.49.3 Documentation des propriétés

4.49.3.1 int SH_OutputZone ::lastVisibleRow

Définition à la ligne 7 du fichier [SH_OutputZone.qml](#).

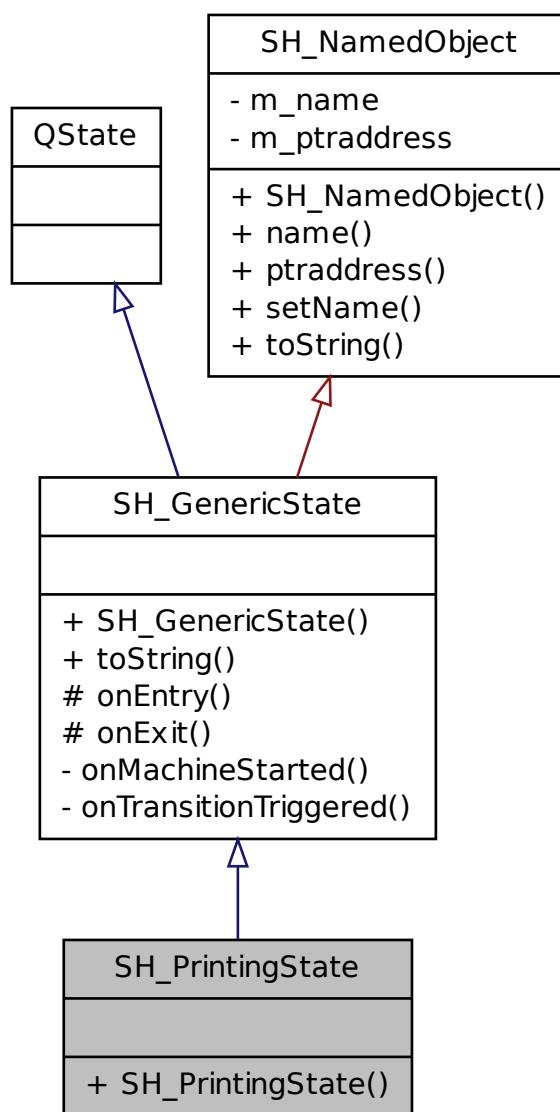
La documentation de cette classe a été générée à partir du fichier suivant :

– views/qml/[SH_OutputZone.qml](#)

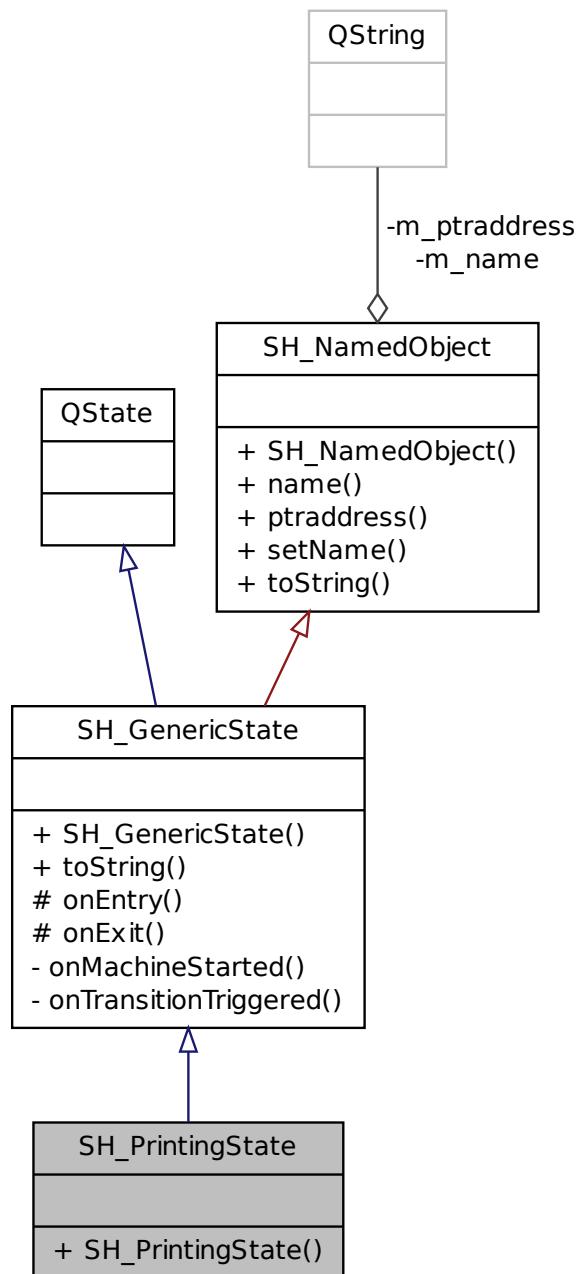
4.50 Référence de la classe SH_PrintingState

```
#include <SH_PrintingState.h>
```

Graphe d'héritage de SH_PrintingState :



Graphe de collaboration de SH_PrintingState :



Signaux

- void `next ()`
- void `printFinished ()`
- void `printStarted ()`

Fonctions membres publiques

- [SH_PrintingState](#) (QString *name*, QState **parent*=0)
- [QString toString \(\)](#)

Fonctions membres protégées

- void [onEntry](#) (QEvent **event*)
- void [onExit](#) (QEvent **event*)

4.50.1 Description détaillée

Définition à la ligne 10 du fichier [SH_PrintingState.h](#).

4.50.2 Documentation des constructeurs et destructeur

4.50.2.1 SH_PrintingState : :SH_PrintingState (QString *name*, QState * *parent* = 0)

Paramètres

<i>name</i>	
<i>parent</i>	

Définition à la ligne 9 du fichier [SH_PrintingState.cpp](#).

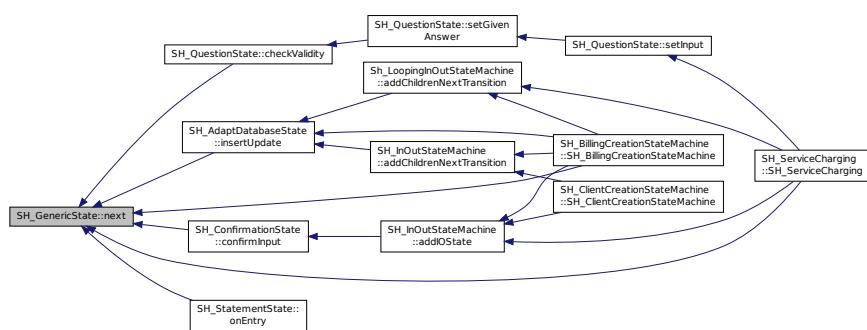
```
00009
00010     SH_GenericState(name, parent)
00011 {
00012 }
```

4.50.3 Documentation des fonctions membres

4.50.3.1 void SH_GenericState : :next() [signal], [inherited]

Référencé par [SH_QuestionState : :checkValidity\(\)](#), [SH_ConfirmationState : :confirmInput\(\)](#), [SH_AdaptDatabaseState : :insertUpdate\(\)](#), [SH_StatementState : :onEntry\(\)](#), [SH_BillingCreationStateMachine : :SH_BillingCreationStateMachine\(\)](#), et [SH_ServiceCharging : :SH_ServiceCharging\(\)](#).

Voici le graphe des appels de cette fonction :



4.50.3.2 void SH_GenericState : :onEntry (QEvent * *event*) [protected], [inherited]

Définition à la ligne 62 du fichier [SH_GenericDebugableState.cpp](#).

Références [SH_NamedObject ::name\(\)](#).

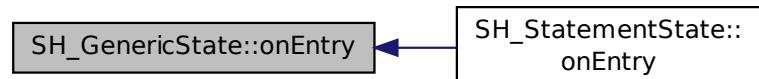
Référencé par [SH_StatementState ::onEntry\(\)](#).

```
00063 {
00064     Q_UNUSED(event);
00065     qDebug() << "Machine: " << machine()->objectName() << " entered " << name();
00066 }
```

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appels de cette fonction :



4.50.3.3 void SH_GenericState ::onExit (QEvent * event) [protected], [inherited]

Définition à la ligne 73 du fichier [SH_GenericDebugableState.cpp](#).

Références [SH_NamedObject ::name\(\)](#).

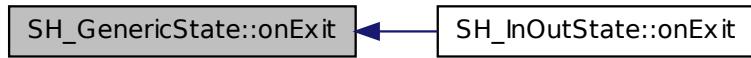
Référencé par [SH_InOutState ::onExit\(\)](#).

```
00074 {
00075     Q_UNUSED(event);
00076     qDebug() << "Machine: " << machine()->objectName() << " exited " << name();
00077 }
```

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :



4.50.3.4 void SH_PrintingState ::printFinished() [signal]

4.50.3.5 void SH_PrintingState ::printStarted() [signal]

4.50.3.6 QString SH_GenericState ::toString() [virtual], [inherited]

Réimplémentée à partir de [SH_NamedObject](#).

Définition à la ligne 21 du fichier [SH_GenericDebugableState.cpp](#).

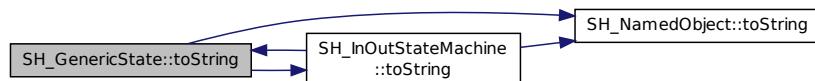
Références [SH_NamedObject ::toString\(\)](#), et [SH_InOutStateMachine ::toString\(\)](#).

Référencé par [SH_InOutStateMachine ::addChildsNextTransition\(\)](#), [SH_DateQuestionState ::rawInput\(\)](#), et [SH_InOutStateMachine ::toString\(\)](#).

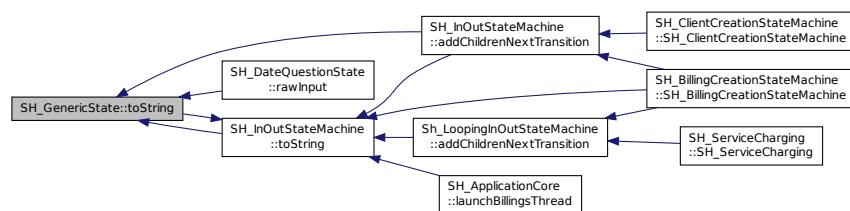
```

00022 {
00023     QStateMachine* machine = this->machine();
00024     SH_InOutStateMachine* mach = qobject_cast<
00025         SH_InOutStateMachine * >(machine);
00026     if(mach) {
00027         return SH_NamedObject::toString() + " [in "+mach->
00028             toString()+"] ";
00029     } else {
00030         return SH_NamedObject::toString();
00029     }
00030 }
```

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :



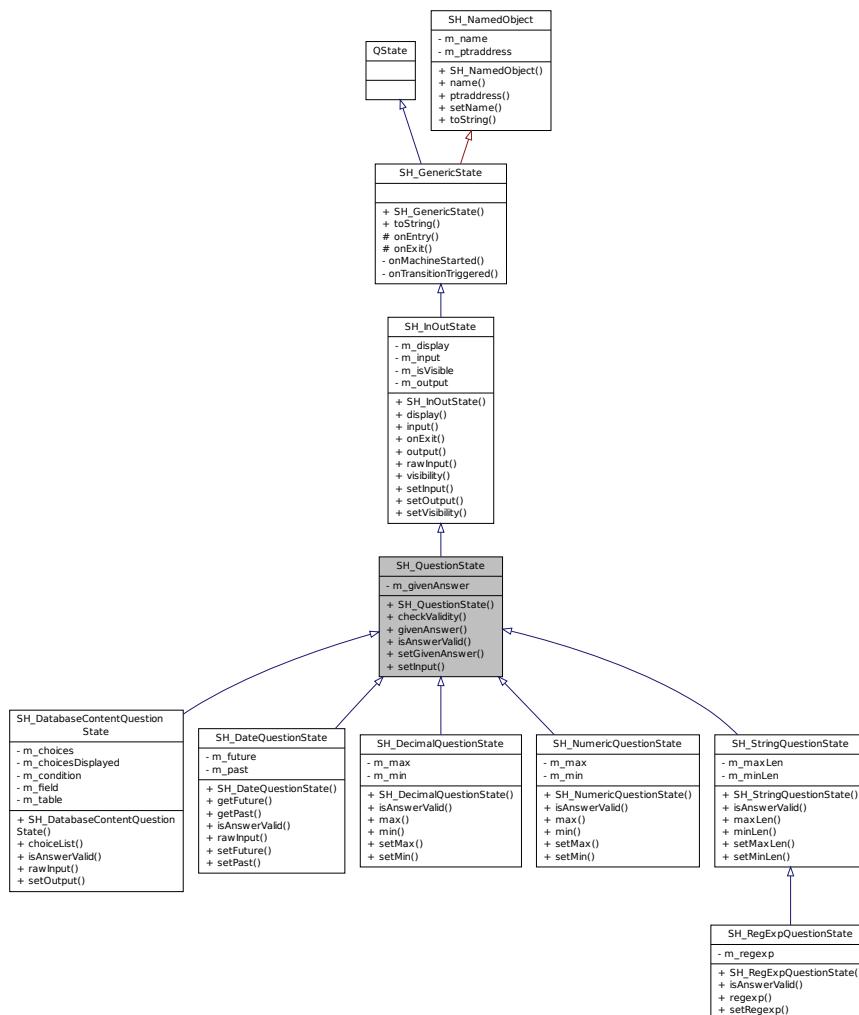
La documentation de cette classe a été générée à partir des fichiers suivants :

- logic/[SH_PrintingState.h](#)
- logic/[SH_PrintingState.cpp](#)

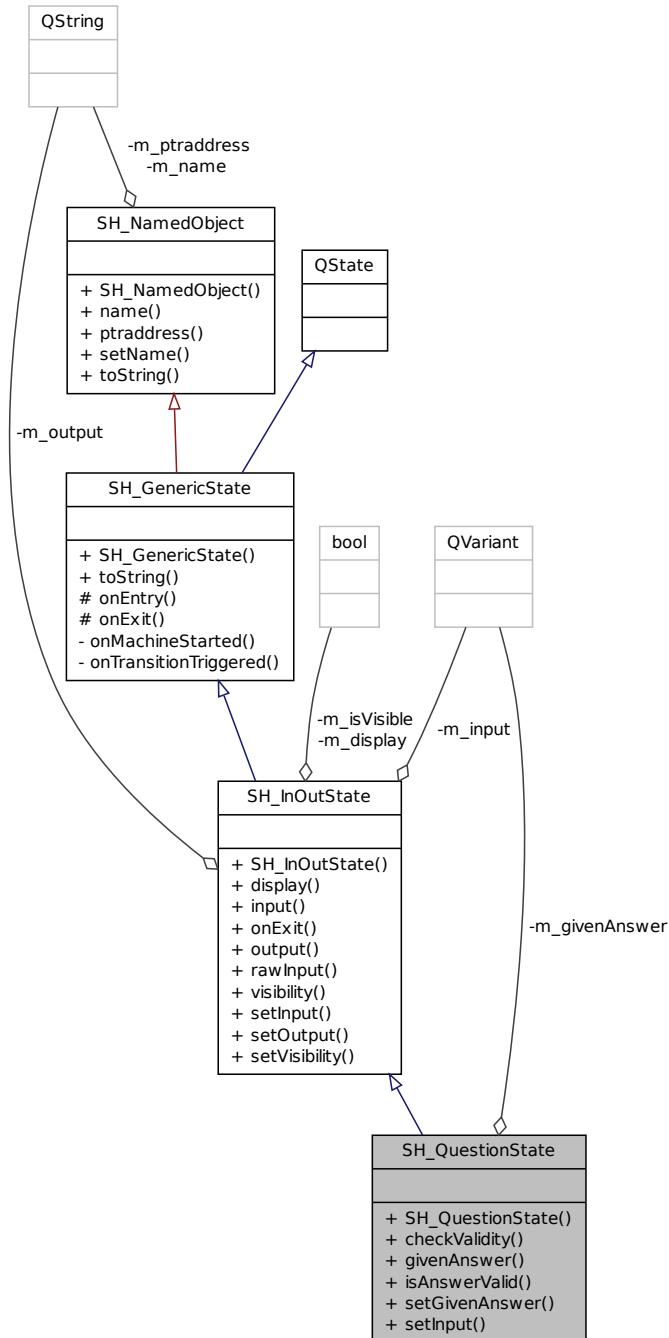
4.51 Référence de la classe SH_QuestionState

```
#include <SH_QuestionState.h>
```

Graphe d'héritage de SH_QuestionState :



Graphe de collaboration de SH_QuestionState :



Connecteurs publics

- `virtual void setOutput (const QString &output)`
- `virtual void setVisibility (bool isVisible)`

Signaux

- `void answerInvalid ()`

- *answerInvalid*
- void **answerValid** ()
 answerValid
- void **next** ()
- void **resendInput** (QVariant *input*)
- void **sendOutput** (QVariant *output*)

Fonctions membres publiques

- **SH_QuestionState** (QString *question*, QString *name*, QState **parent*=0)
- bool **checkValidity** ()
- void **display** (bool *canDisplay*)
- virtual QVariant **givenAnswer** () const
- virtual QVariant **input** () const
- virtual bool **isValid** (const QVariant &*givenAnswer*)=0
- void **onExit** (QEvent **event*)
- virtual QString **output** () const
- virtual QVariant **rawInput** () const
- virtual void **setGivenAnswer** (const QVariant &*givenAnswer*)
- virtual void **setInput** (const QVariant &*input*)
- QString **toString** ()
- bool **visibility** ()

Fonctions membres protégées

- void **onEntry** (QEvent **event*)

Attributs privés

- QVariant **m_givenAnswer**
 m_givenAnswer

4.51.1 Description détaillée

Définition à la ligne 11 du fichier [SH_QuestionState.h](#).

4.51.2 Documentation des constructeurs et destructeur

4.51.2.1 SH_QuestionState : :SH_QuestionState (QString *question*, QString *name*, QState * *parent* = 0)

Paramètres

<i>question</i>	
<i>name</i>	
<i>parent</i>	

Définition à la ligne 9 du fichier [SH_QuestionState.cpp](#).

```
00009
00010     SH_InOutState(question, name, parent)
00011 {
00012 }
```

:

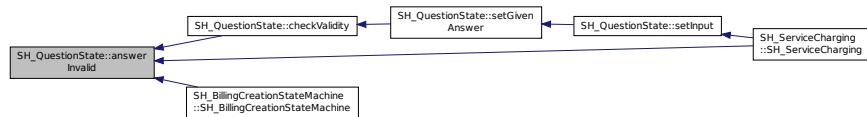
4.51.3 Documentation des fonctions membres

4.51.3.1 void SH_QuestionState : :answerInvalid () [signal]

answerInvalid

Référencé par `checkValidity()`, `SH_BillingCreationStateMachine` : `:SH_BillingCreationStateMachine()`, et `SH_ServiceCharging` : `:SH_ServiceCharging()`.

Voici le graphe des appels de cette fonction :

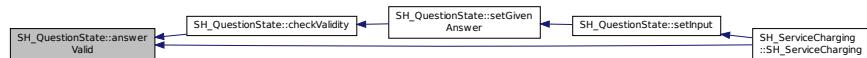


4.51.3.2 void SH_QuestionState ::answerValid() [signal]

`answerValid`

Référencé par `checkValidity()`, et `SH_ServiceCharging` : `:SH_ServiceCharging()`.

Voici le graphe des appels de cette fonction :



4.51.3.3 bool SH_QuestionState ::checkValidity()

Définition à la ligne 20 du fichier `SH_QuestionState.cpp`.

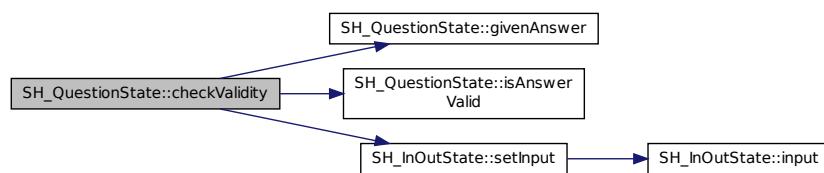
Références `answerInvalid()`, `answerValid()`, `givenAnswer()`, `isAnswerValid()`, `SH_GenericState ::next()`, et `SH_InOutState ::setInput()`.

Référencé par `setGivenAnswer()`.

```

00021 {
00022     bool ok = this->isAnswerValid(this->givenAnswer());
00023     if(ok) {
00024         qDebug() << this->givenAnswer() << " answer valid !";
00025         qDebug() << this->givenAnswer();
00026         SH_InOutState::setInput(this->givenAnswer());
00027         emit answerValid();
00028         emit next();
00029     } else {
00030         qDebug() << this->givenAnswer() << " answer invalid :-(";
00031         qDebug() << this->givenAnswer();
00032         emit answerInvalid();
00033     }
00034     return ok;
00035 }
```

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :



4.51.3.4 void SH_InOutState ::display (bool canDisplay) [inherited]

Définition à la ligne 95 du fichier [SH_IOSState.cpp](#).

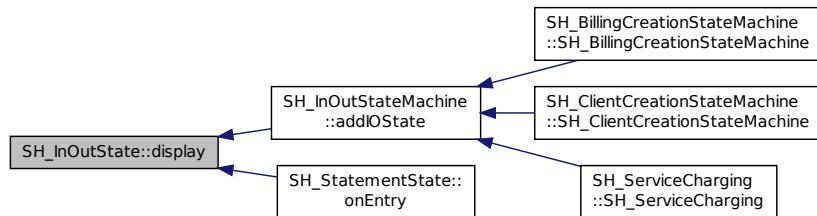
Références [SH_InOutState ::m_display](#), [SH_InOutState ::m_isVisible](#), [SH_InOutState ::m_output](#), et [SH_InOutState ::sendOutput\(\)](#).

Référencé par [SH_InOutStateMachine ::addIOState\(\)](#), et [SH_StatementState ::onEntry\(\)](#).

```

00096 {
00097     m_display=canDisplay;
00098     if(m_display && !m_output.isEmpty() && m_isVisible) {
00099         qDebug() << "resalut !" << QVariant(m_output);
00100         emit sendOutput(QVariant(m_output));
00101     }
00102 }
```

Voici le graphe des appelants de cette fonction :



4.51.3.5 QVariant SH_QuestionState ::givenAnswer () const [virtual]

Définition à la ligne 55 du fichier [SH_QuestionState.cpp](#).

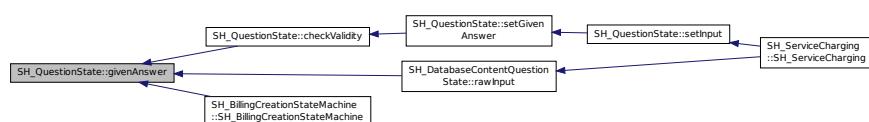
Références [m_givenAnswer](#).

Référencé par [checkValidity\(\)](#), [SH_DatabaseContentQuestionState ::rawInput\(\)](#), et [SH_BillingCreationStateMachine ::SH_BillingCreationStateMachine\(\)](#).

```

00056 {
00057     return this->m_givenAnswer;
00058 }
```

Voici le graphe des appelants de cette fonction :



4.51.3.6 QVariant SH_InOutState ::input() const [virtual], [inherited]

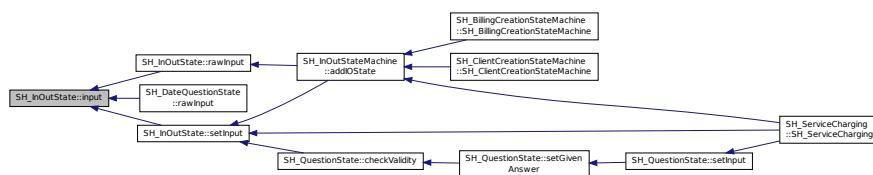
Définition à la ligne 20 du fichier [SH_IOState.cpp](#).

Références [SH_InOutState ::m_input](#).

Référencé par [SH_InOutState ::rawInput\(\)](#), [SH_DateQuestionState ::rawInput\(\)](#), et [SH_InOutState ::setInput\(\)](#).

```
00021 {
00022     return m_input;
00023 }
```

Voici le graphe des appelants de cette fonction :



4.51.3.7 virtual bool SH_QuestionState ::isValid(const QVariant & givenAnswer) [pure virtual]

Implémenté dans [SH_DatabaseContentQuestionState](#), [SH.DecimalQuestionState](#), [SH.DateQuestionState](#), [SH.NumericQuestionState](#), [SH.StringQuestionState](#), et [SH.RegExpQuestionState](#).

Référencé par [checkValidity\(\)](#).

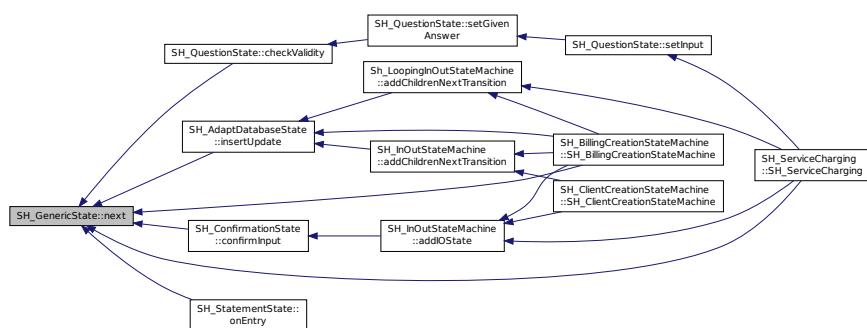
Voici le graphe des appelants de cette fonction :



4.51.3.8 void SH_GenericState ::next() [signal], [inherited]

Référencé par [checkValidity\(\)](#), [SH_ConfirmationState ::confirmInput\(\)](#), [SH_AdaptDatabaseState ::insertUpdate\(\)](#), [SH_StatementState ::onEntry\(\)](#), [SH_BillingCreationStateMachine ::SH_BillingCreationStateMachine\(\)](#), et [SH_ServiceCharging ::SH_ServiceCharging\(\)](#).

Voici le graphe des appelants de cette fonction :



4.51.3.9 void SH_GenericState ::onEntry (QEvent * event) [protected], [inherited]

Définition à la ligne 62 du fichier [SH_GenericDebugableState.cpp](#).

Références [SH_NamedObject ::name\(\)](#).

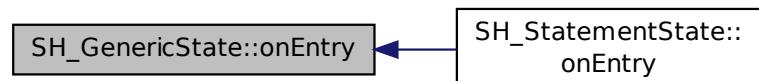
Référencé par [SH_StatementState ::onEntry\(\)](#).

```
00063 {
00064     Q_UNUSED(event);
00065     qDebug() << "Machine: " << machine()->objectName() << " entered " << name();
00066 }
```

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :



4.51.3.10 void SH_InOutState ::onExit (QEvent * event) [inherited]

Définition à la ligne 110 du fichier [SH_IOState.cpp](#).

Références [SH_InOutState ::m_input](#), [SH_InOutState ::m_isVisible](#), [SH_GenericState ::onExit\(\)](#), et [SH_InOutState ::resendInput\(\)](#).

```
00111 {
00112     SH_GenericState::onExit(event);
00113     if(m_isVisible) {
00114         emit resendInput(m_input);
00115     }
00116 }
```

Voici le graphe d'appel pour cette fonction :



4.51.3.11 QString SH_InOutState ::output() const [virtual], [inherited]

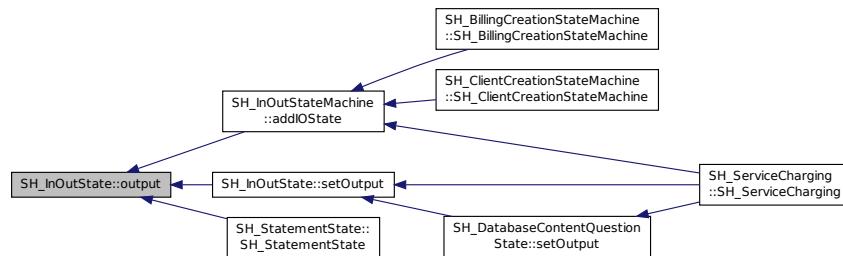
Définition à la ligne 56 du fichier [SH_IOState.cpp](#).

Références [SH_InOutState ::m_output](#).

Référencé par [SH_InOutStateMachine ::addIOState\(\)](#), [SH_InOutState ::setOutput\(\)](#), et [SH_StatementState ::SH_StatementState\(\)](#).

```
00057 {
00058     return m_output;
00059 }
```

Voici le graphe des appels de cette fonction :



4.51.3.12 QVariant SH_InOutState ::rawInput() const [virtual], [inherited]

Réimplémentée dans [SH_DateQuestionState](#), et [SH_DatabaseContentQuestionState](#).

Définition à la ligne 30 du fichier [SH_IOState.cpp](#).

Références [SH_InOutState ::input\(\)](#).

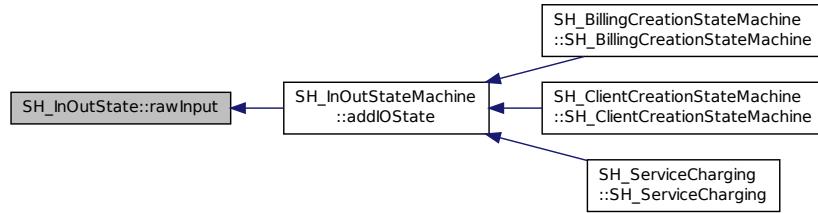
Référencé par [SH_InOutStateMachine ::addIOState\(\)](#).

```
00031 {
00032     return input();
00033 }
```

Voici le graphe d'appel pour cette fonction :



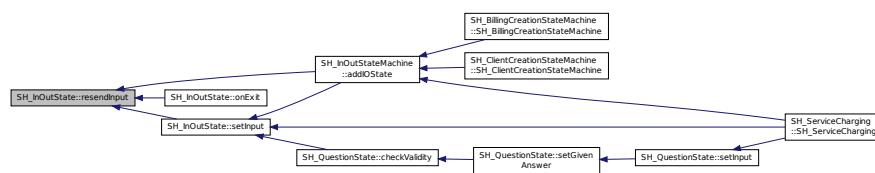
Voici le graphe des appelants de cette fonction :



4.51.3.13 void SH_InOutState ::resendInput (QVariant *input*) [signal], [inherited]

Référencé par [SH_InOutStateMachine ::addIOState\(\)](#), [SH_InOutState ::onExit\(\)](#), et [SH_InOutState ::setInput\(\)](#).

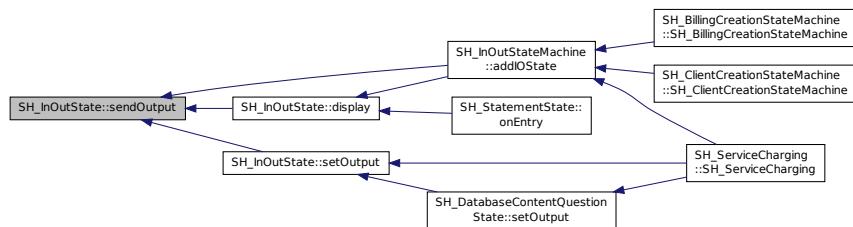
Voici le graphe des appelants de cette fonction :



4.51.3.14 void SH_InOutState ::sendOutput (QVariant *output*) [signal], [inherited]

Référencé par [SH_InOutStateMachine ::addIOState\(\)](#), [SH_InOutState ::display\(\)](#), et [SH_InOutState ::setOutput\(\)](#).

Voici le graphe des appelants de cette fonction :



4.51.3.15 void SH_QuestionState ::setGivenAnswer (const QVariant & *givenAnswer*) [virtual]

Définition à la ligne 66 du fichier [SH_QuestionState.cpp](#).

Références [checkValidity\(\)](#), et [m_givenAnswer](#).

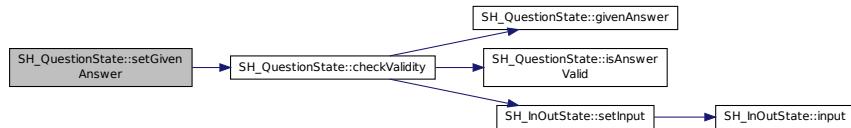
Référencé par [setInput\(\)](#).

```

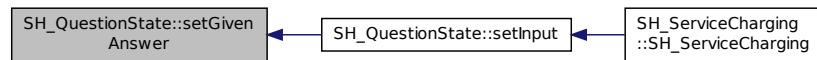
00067 {
00068     this->m_givenAnswer = givenAnswer;
00069     this->checkValidity();
00070 }

```

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :



4.51.3.16 void SH_QuestionState ::setInput (const QVariant & input) [virtual]

Réimplémentée à partir de [SH_InOutState](#).

Définition à la ligne 43 du fichier [SH_QuestionState.cpp](#).

Références [setGivenAnswer\(\)](#).

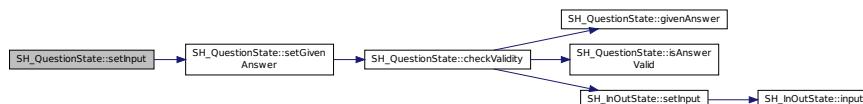
Référencé par [SH_ServiceCharging ::SH_ServiceCharging\(\)](#).

```

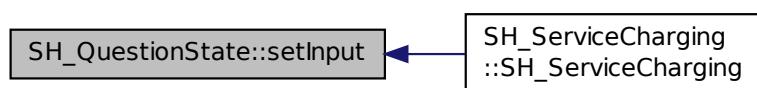
00044 {
00045     qDebug() << "new answer " << input.toString();
00046     this->setGivenAnswer(input);
00047 }

```

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :



4.51.3.17 void SH_InOutState : :setOutput (const QString & *output*) [virtual], [slot], [inherited]

Réimplémentée dans [SH_DatabaseContentQuestionState](#).

Définition à la ligne 68 du fichier [SH_IOState.cpp](#).

Références [SH_InOutState : :m_isVisible](#), [SH_InOutState : :m_output](#), [SH_InOutState : :output\(\)](#), et [SH_InOutState : :sendOutput\(\)](#).

Référencé par [SH_DatabaseContentQuestionState : :setOutput\(\)](#), et [SH_ServiceCharging : :SH_ServiceCharging\(\)](#).

```
00069 {
00070     m_output = output;
00071     if(m_isVisible) {
00072         emit sendOutput(QVariant(m_output));
00073     }
00074 }
```

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :



4.51.3.18 void SH_InOutState : :setVisibility (bool *isVisible*) [virtual], [slot], [inherited]

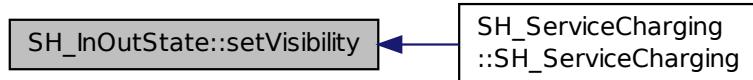
Définition à la ligne 81 du fichier [SH_IOState.cpp](#).

Références [SH_InOutState : :m_isVisible](#).

Référencé par [SH_ServiceCharging : :SH_ServiceCharging\(\)](#).

```
00082 {
00083     m_isVisible = isVisible;
00084 }
```

Voici le graphe des appelants de cette fonction :



4.51.3.19 QString SH_GenericState : :toString() [virtual], [inherited]

Réimplémentée à partir de [SH_NamedObject](#).

Définition à la ligne 21 du fichier [SH_GenericDebugableState.cpp](#).

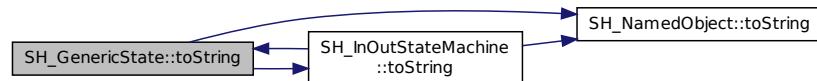
Références [SH_NamedObject : :toString\(\)](#), et [SH_InOutStateMachine : :toString\(\)](#).

Référencé par [SH_InOutStateMachine : :addChildsNextTransition\(\)](#), [SH_DateQuestionState : :rawInput\(\)](#), et [SH_InOutStateMachine : :toString\(\)](#).

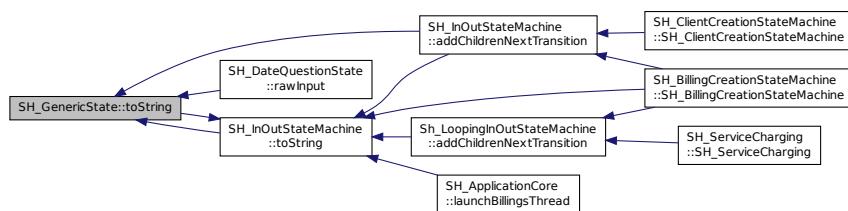
```

00022 {
00023     QStateMachine* machine = this->machine();
00024     SH_InOutStateMachine* mach = qobject_cast<
00025         SH_InOutStateMachine * >(machine);
00026     if(mach) {
00027         return SH_NamedObject::toString() + " [in "+mach->
00028             toString()+"] ";
00029     } else {
00030         return SH_NamedObject::toString();
00031     }
00030 }
  
```

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :



4.51.3.20 bool SH_InOutState::visibility() [inherited]

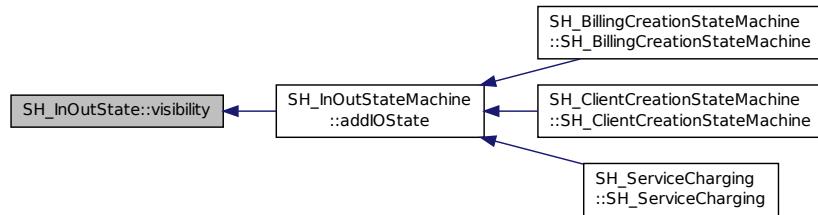
Définition à la ligne 91 du fichier [SH_IOState.cpp](#).

Références [SH_InOutState::m_isVisible](#).

Référencé par [SH_InOutStateMachine::addIOState\(\)](#).

```
00091     {
00092         return m_isVisible;
00093     }
```

Voici le graphe des appels de cette fonction :



4.51.4 Documentation des données membres

4.51.4.1 QVariant SH_QuestionState::m_givenAnswer [private]

`m_givenAnswer`

Définition à la ligne 80 du fichier [SH_QuestionState.h](#).

Référencé par [givenAnswer\(\)](#), et [setGivenAnswer\(\)](#).

La documentation de cette classe a été générée à partir des fichiers suivants :

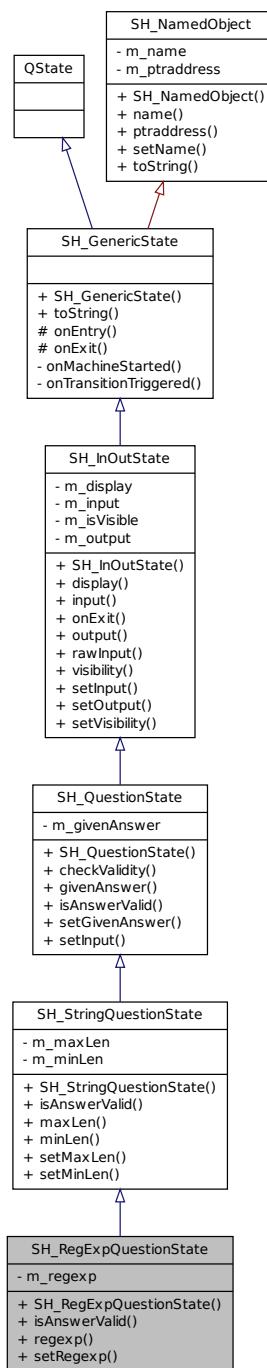
- logic/[SH_QuestionState.h](#)
- logic/[SH_QuestionState.cpp](#)

4.52 Référence de la classe SH_RegExpQuestionState

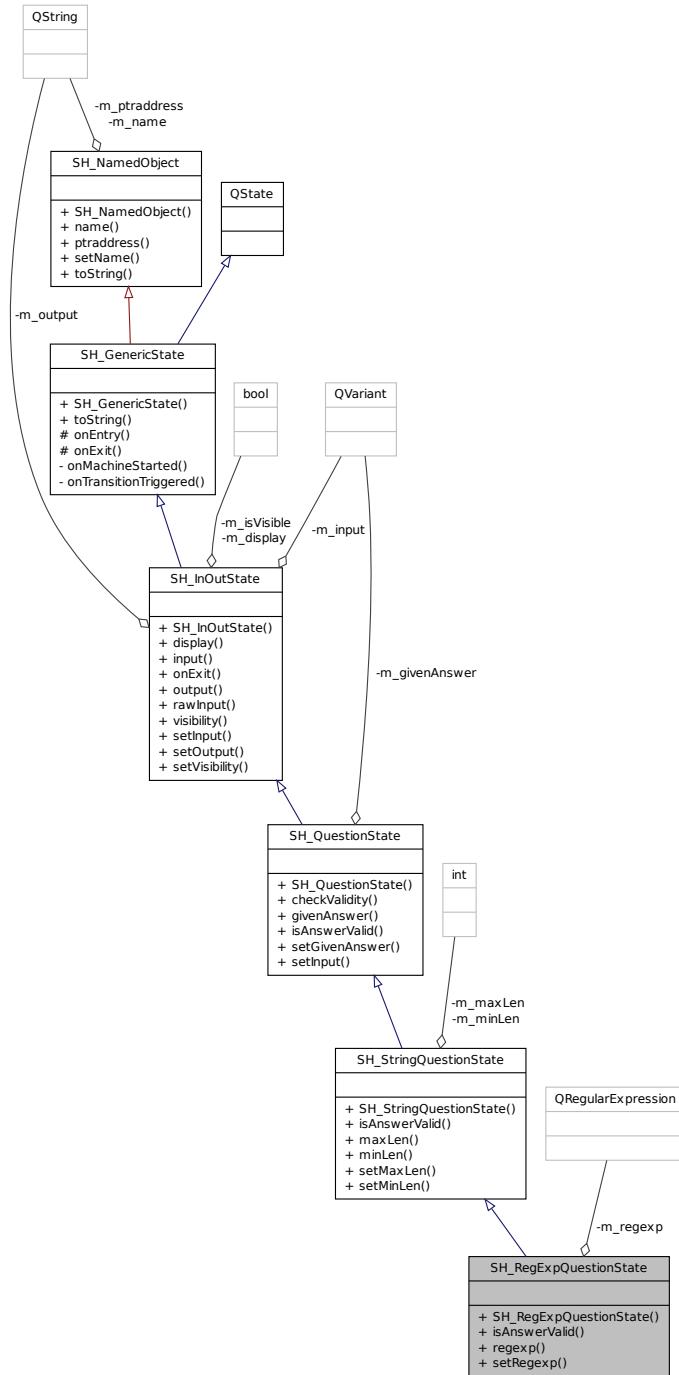
The [SH_RegExpQuestionState](#) class.

```
#include <SH_RegExpQuestionState.h>
```

Graphe d'héritage de SH_RegExpQuestionState :



Graphe de collaboration de SH_RegExpQuestionState :



Connecteurs publics

- `virtual void setOutput (const QString &output)`
- `virtual void setVisibility (bool isVisible)`

Signaux

- `void answerInvalid ()`

```

    answerInvalid
- void answerValid ()
    answerValid
- void next ()
- void resendInput (QVariant input)
- void sendOutput (QVariant output)

```

Fonctions membres publiques

- **SH_RegExpQuestionState** (QString question, QString name, QRegularExpression regex=QRegularExpression(), QState *parent=0)
- bool **checkValidity** ()
- void **display** (bool canDisplay)
- virtual QVariant **givenAnswer** () const
- virtual QVariant **input** () const
- virtual bool **isValid** (const QVariant &givenAnswer)
- int **maxLen** () const
- int **minLen** () const
- void **onExit** (QEvent *event)
- virtual QString **output** () const
- virtual QVariant **rawInput** () const
- QRegularExpression **regexp** () const
- virtual void **setGivenAnswer** (const QVariant &givenAnswer)
- virtual void **setInput** (const QVariant &input)
- void **setMaxLen** (int maxLen)
- void **setMinLen** (int minLen)
- void **setRegexp** (const QRegularExpression ®exp)
- QString **toString** ()
- bool **visibility** ()

Fonctions membres protégées

- void **onEntry** (QEvent *event)

Attributs privés

- QRegularExpression **m_regexp**
 m_regexp

4.52.1 Description détaillée

The **SH_RegExpQuestionState** class.

Définition à la ligne 8 du fichier **SH_RegExpQuestionState.h**.

4.52.2 Documentation des constructeurs et destructeur

4.52.2.1 SH_RegExpQuestionState : :SH_RegExpQuestionState (QString question, QString name, QRegularExpression regex = QRegularExpression (), QState * parent = 0)

Définition à la ligne 8 du fichier **SH_RegExpQuestionState.cpp**.

```

00008
:
00009     SH_StringQuestionState(question, name, 0, -1, parent)
00010 {
00011 }

```

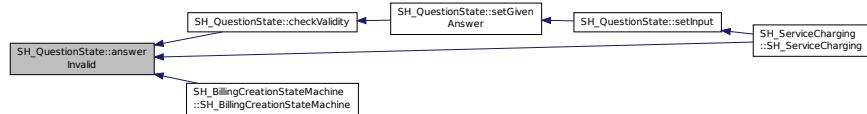
4.52.3 Documentation des fonctions membres

4.52.3.1 void SH_QuestionState ::answerInvalid() [signal], [inherited]

answerInvalid

Référencé par [SH_QuestionState ::checkValidity\(\)](#), [SH_BillingCreationStateMachine ::SH_BillingCreationStateMachine\(\)](#), et [SH_ServiceCharging ::SH_ServiceCharging\(\)](#).

Voici le graphe des appels de cette fonction :

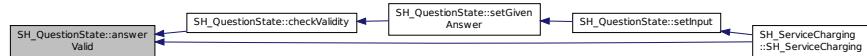


4.52.3.2 void SH_QuestionState ::answerValid() [signal], [inherited]

answerValid

Référencé par [SH_QuestionState ::checkValidity\(\)](#), et [SH_ServiceCharging ::SH_ServiceCharging\(\)](#).

Voici le graphe des appels de cette fonction :



4.52.3.3 bool SH_QuestionState ::checkValidity() [inherited]

Définition à la ligne 20 du fichier [SH_QuestionState.cpp](#).

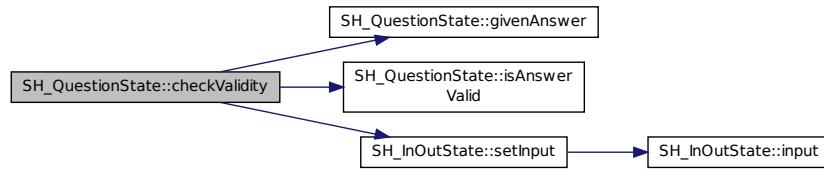
Références [SH_QuestionState ::answerInvalid\(\)](#), [SH_QuestionState ::answerValid\(\)](#), [SH_QuestionState ::givenAnswer\(\)](#), [SH_QuestionState ::isValid\(\)](#), [SH_GenericState ::next\(\)](#), et [SH_InOutState ::setInput\(\)](#).

Référencé par [SH_QuestionState ::setGivenAnswer\(\)](#).

```

00021 {
00022     bool ok = this->isValid(this->givenAnswer());
00023     if(ok) {
00024         qDebug() << this->givenAnswer() << " answer valid !";
00025         qDebug() << this->givenAnswer();
00026         SH_InOutState::setInput(this->givenAnswer());
00027         emit answerValid();
00028         emit next();
00029     } else {
00030         qDebug() << this->givenAnswer() << " answer invalid :-(";
00031         qDebug() << this->givenAnswer();
00032         emit answerInvalid();
00033     }
00034     return ok;
00035 }
  
```

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appels de cette fonction :



4.52.3.4 void SH_InOutState ::display (bool canDisplay) [inherited]

Définition à la ligne 95 du fichier [SH_IOState.cpp](#).

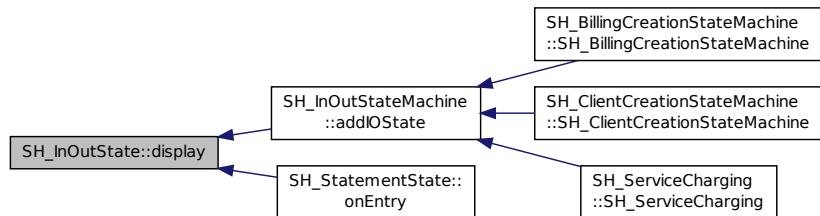
Références [SH_InOutState ::m_display](#), [SH_InOutState ::m_isVisible](#), [SH_InOutState ::m_output](#), et [SH_InOutState ::sendOutput\(\)](#).

Référencé par [SH_InOutStateMachine ::addIOState\(\)](#), et [SH_StatementState ::onEntry\(\)](#).

```

00096 {
00097     m_display=canDisplay;
00098     if(m_display && !m_output.isEmpty() && m_isVisible) {
00099         qDebug() << "resalut !" << QVariant(m_output);
00100         emit sendOutput(QVariant(m_output));
00101     }
00102 }
```

Voici le graphe des appels de cette fonction :



4.52.3.5 QVariant SH_QuestionState ::givenAnswer () const [virtual], [inherited]

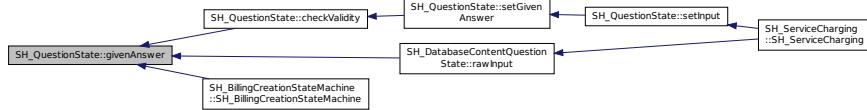
Définition à la ligne 55 du fichier [SH_QuestionState.cpp](#).

Références [SH_QuestionState ::m_givenAnswer](#).

Référencé par [SH_QuestionState ::checkValidity\(\)](#), [SH_DatabaseContentQuestionState ::rawInput\(\)](#), et [SH_BillingCreationStateMachine ::SH_BillingCreationStateMachine\(\)](#).

```
00056 {
00057     return this->m_givenAnswer;
00058 }
```

Voici le graphe des appels de cette fonction :



4.52.3.6 QVariant SH_InOutState ::input() const [virtual], [inherited]

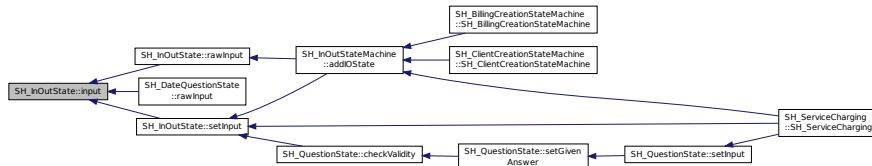
Définition à la ligne 20 du fichier [SH_IOState.cpp](#).

Références [SH_InOutState ::m_input](#).

Référencé par [SH_InOutState ::rawInput\(\)](#), [SH_DateQuestionState ::rawInput\(\)](#), et [SH_InOutState ::setInput\(\)](#).

```
00021 {
00022     return m_input;
00023 }
```

Voici le graphe des appels de cette fonction :



4.52.3.7 bool SH_RegExpQuestionState ::isValid(const QVariant & givenAnswer) [virtual]

Réimplémentée à partir de [SH_StringQuestionState](#).

Définition à la ligne 18 du fichier [SH_RegExpQuestionState.cpp](#).

Références [m_regex](#).

```
00019 {
00020     QString answer = givenAnswer.toString();
00021     QRegularExpressionMatch found = m_regex.match(answer);
00022     return (found.hasMatch() && (found.captured(0) == answer));
00023 }
```

4.52.3.8 int SH_StringQuestionState ::maxLen() const [inherited]

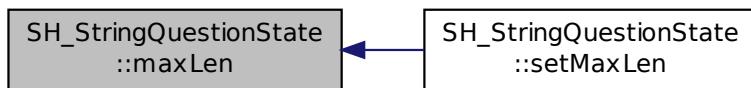
Définition à la ligne 39 du fichier [SH_StringQuestionState.cpp](#).

Références [SH_StringQuestionState ::m_maxLen](#).

Référencé par [SH_StringQuestionState ::setMaxLen\(\)](#).

```
00040 {
00041     return m_maxLen;
00042 }
```

Voici le graphe des appelants de cette fonction :



4.52.3.9 int SH_StringQuestionState ::minLen() const [inherited]

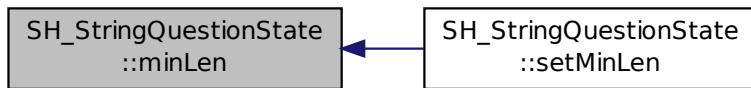
Définition à la ligne 61 du fichier [SH_StringQuestionState.cpp](#).

Références [SH_StringQuestionState ::m_minLen](#).

Référencé par [SH_StringQuestionState ::setMinLen\(\)](#).

```
00062 {
00063     return m_minLen;
00064 }
```

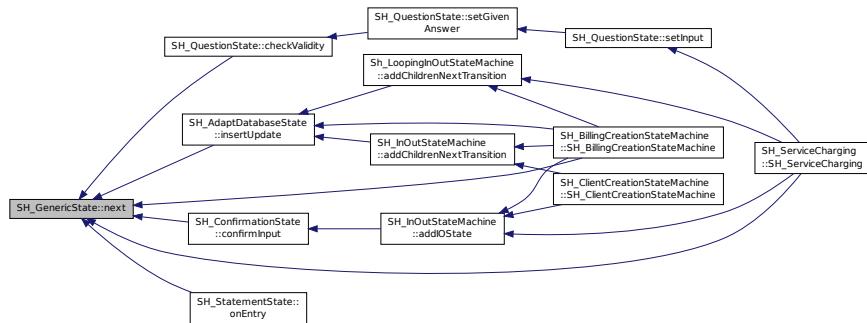
Voici le graphe des appelants de cette fonction :



4.52.3.10 void SH_GenericState ::next() [signal], [inherited]

Référencé par [SH_QuestionState ::checkValidity\(\)](#), [SH_ConfirmationState ::confirmInput\(\)](#), [SH_AdaptDatabaseState ::insertUpdate\(\)](#), [SH_StatementState ::onEntry\(\)](#), [SH_BillingCreationStateMachine ::SH_BillingCreationStateMachine\(\)](#), et [SH_ServiceCharging ::SH_ServiceCharging\(\)](#).

Voici le graphe des appelants de cette fonction :



4.52.3.11 void SH_GenericState ::onEntry (QEvent * event) [protected], [inherited]

Définition à la ligne 62 du fichier [SH_GenericDebugableState.cpp](#).

Références [SH_NamedObject ::name\(\)](#).

Référencé par [SH_StatementState ::onEntry\(\)](#).

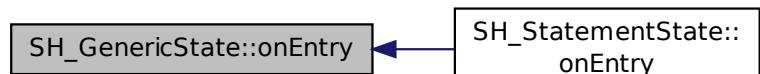
```

00063 {
00064     Q_UNUSED(event);
00065     qDebug() << "Machine: " << machine()->objectName() << " entered " << name();
00066 }
```

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :



4.52.3.12 void SH_InOutState ::onExit (QEvent * event) [inherited]

Définition à la ligne 110 du fichier [SH_IOState.cpp](#).

Références [SH_InOutState ::m_input](#), [SH_InOutState ::m_isVisible](#), [SH_GenericState ::onExit\(\)](#), et [SH_InOutState ::resendInput\(\)](#).

```

00111 {
00112     SH_GenericState::onExit(event);
00113     if(m_isVisible) {
00114         emit resendInput(m_input);
00115     }
00116 }
```

Voici le graphe d'appel pour cette fonction :



4.52.3.13 QString SH_InOutState::output() const [virtual], [inherited]

Définition à la ligne 56 du fichier [SH_IOState.cpp](#).

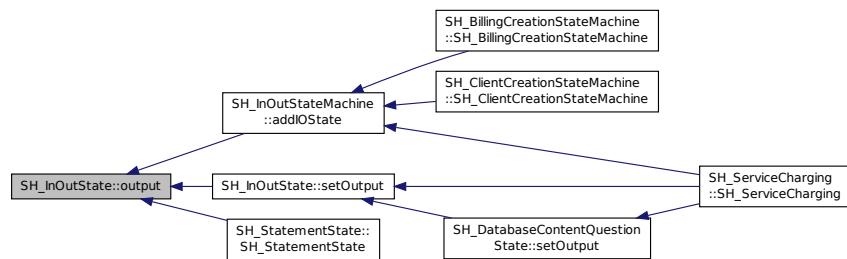
Références [SH_InOutState::m_output](#).

Référencé par [SH_InOutStateMachine::addIOState\(\)](#), [SH_InOutState::setOutput\(\)](#), et [SH_StatementState::SH_StatementState\(\)](#).

```

00057 {
00058     return m_output;
00059 }
```

Voici le graphe des appels de cette fonction :



4.52.3.14 QVariant SH_InOutState::rawInput() const [virtual], [inherited]

Réimplémentée dans [SH_DateQuestionState](#), et [SH_DatabaseContentQuestionState](#).

Définition à la ligne 30 du fichier [SH_IOState.cpp](#).

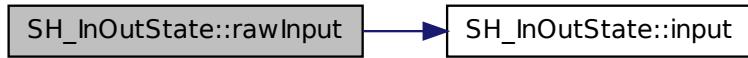
Références [SH_InOutState::input\(\)](#).

Référencé par [SH_InOutStateMachine::addIOState\(\)](#).

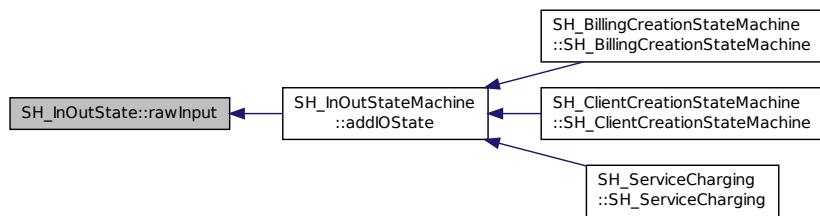
```

00031 {
00032     return input();
00033 }
```

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :



4.52.3.15 QRegularExpression SH_RegExpQuestionState ::regexp() const

Définition à la ligne 31 du fichier [SH_RegExpQuestionState.cpp](#).

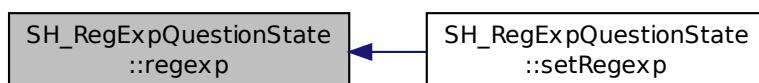
Références [m_regexp](#).

Référencé par [setRegexp\(\)](#).

```

00032 {
00033     return m_regexp;
00034 }
  
```

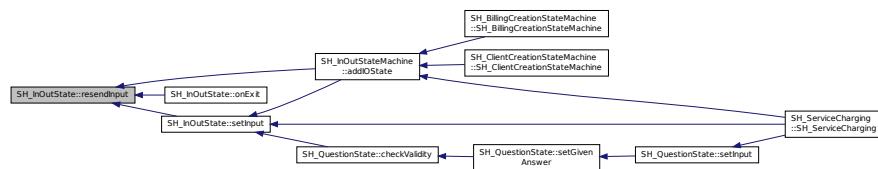
Voici le graphe des appelants de cette fonction :



4.52.3.16 void SH_InOutState ::resendInput(QVariant input) [signal], [inherited]

Référencé par [SH_InOutStateMachine ::addIOState\(\)](#), [SH_InOutState ::onExit\(\)](#), et [SH_InOutState ::setInput\(\)](#).

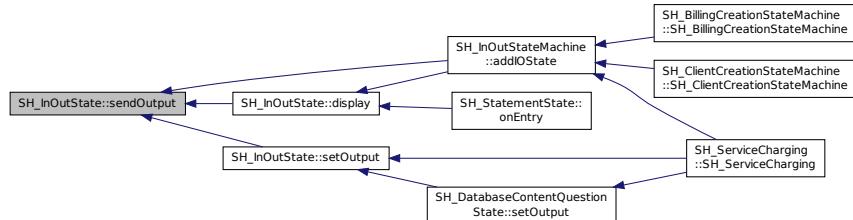
Voici le graphe des appels de cette fonction :



4.52.3.17 void SH_InOutState ::sendOutput(QVariant output) [signal], [inherited]

Référencé par `SH_InOutStateMachine` : `:addIOState()`, `SH_InOutState` : `:display()`, et `SH_InOutState` : `:setOutput()`.

Voici le graphe des appels de cette fonction :



4.52.3.18 void SH_QuestionState ::setGivenAnswer (const QVariant & givenAnswer) [virtual], [inherited]

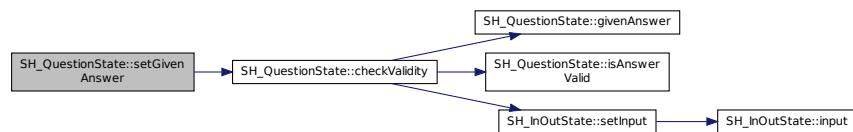
Définition à la ligne 66 du fichier SH_QuestionState.cpp.

Références `SH_QuestionState::checkValidity()`, et `SH_QuestionState::m_givenAnswer`.

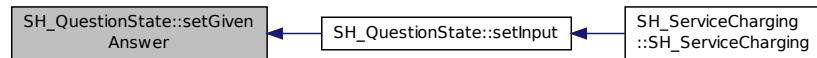
Référencé par [SH_QuestionState](#) ::[setInput\(\)](#).

```
00067 {  
00068     this->m_givenAnswer = givenAnswer;  
00069     this->checkValidity();  
00070 }
```

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :



4.52.3.19 void SH_QuestionState ::setInput (const QVariant & input) [virtual], [inherited]

Réimplémentée à partir de [SH_InOutState](#).

Définition à la ligne 43 du fichier [SH_QuestionState.cpp](#).

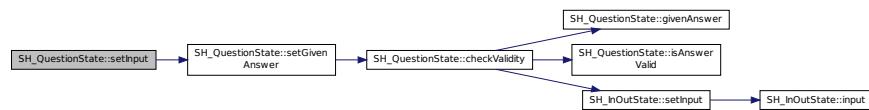
Références [SH_QuestionState ::setGivenAnswer\(\)](#).

Référencé par [SH_ServiceCharging ::SH_ServiceCharging\(\)](#).

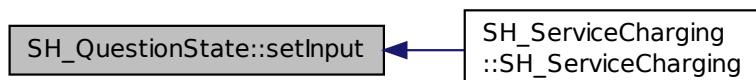
```

00044 {
00045     qDebug() << "new answer " << input.toString();
00046     this->setGivenAnswer(input);
00047 }
  
```

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :



4.52.3.20 void SH_StringQuestionState ::setMaxLen (int maxlen) [inherited]

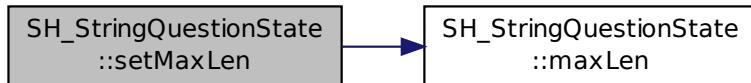
Définition à la ligne 50 du fichier [SH_StringQuestionState.cpp](#).

Références [SH_StringQuestionState ::m_maxLen](#), et [SH_StringQuestionState ::maxLen\(\)](#).

```

00051 {
00052     m_maxLen = maxlen;
00053 }
  
```

Voici le graphe d'appel pour cette fonction :



4.52.3.21 void SH_StringQuestionState ::setMinLen (int minLen) [inherited]

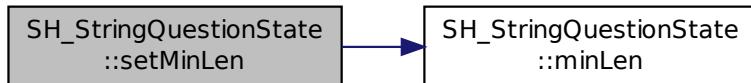
Définition à la ligne 72 du fichier [SH_StringQuestionState.cpp](#).

Références [SH_StringQuestionState ::m_minLen](#), et [SH_StringQuestionState ::minLen\(\)](#).

```

00073 {
00074     m_minLen = minLen;
00075 }
```

Voici le graphe d'appel pour cette fonction :



4.52.3.22 void SH_InOutState ::setOutput (const QString & output) [virtual], [slot], [inherited]

Réimplémentée dans [SH_DatabaseContentQuestionState](#).

Définition à la ligne 68 du fichier [SH_IOState.cpp](#).

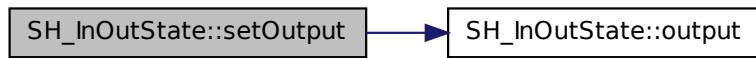
Références [SH_InOutState ::m_isVisible](#), [SH_InOutState ::m_output](#), [SH_InOutState ::output\(\)](#), et [SH_InOutState ::sendOutput\(\)](#).

Référencé par [SH_DatabaseContentQuestionState ::setOutput\(\)](#), et [SH_ServiceCharging ::SH_ServiceCharging\(\)](#).

```

00069 {
00070     m_output = output;
00071     if(m_isVisible) {
00072         emit sendOutput(QVariant(m_output));
00073     }
00074 }
```

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appels de cette fonction :



4.52.3.23 void SH_RegExpQuestionState ::setRegexp (const QRegularExpression & regexp)

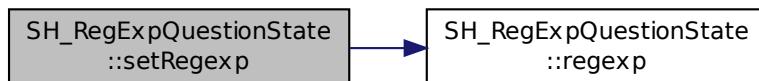
Définition à la ligne 41 du fichier [SH_RegExpQuestionState.cpp](#).

Références [m_regexp](#), et [regexp\(\)](#).

```

00042 {
00043     m_regexp = regexp;
00044 }
```

Voici le graphe d'appel pour cette fonction :



4.52.3.24 void SH_InOutState ::setVisibility (bool isVisible) [virtual], [slot], [inherited]

Définition à la ligne 81 du fichier [SH_IOState.cpp](#).

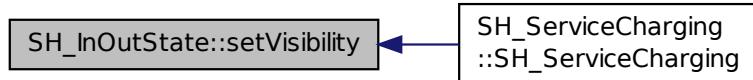
Références [SH_InOutState ::m_isVisible](#).

Référencé par [SH_ServiceCharging ::SH_ServiceCharging\(\)](#).

```

00082 {
00083     m_isVisible = isVisible;
00084 }
```

Voici le graphe des appelants de cette fonction :



4.52.3.25 QString SH_GenericState : :toString() [virtual], [inherited]

Réimplémentée à partir de [SH_NamedObject](#).

Définition à la ligne 21 du fichier [SH_GenericDebugableState.cpp](#).

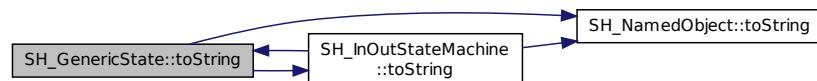
Références [SH_NamedObject : :toString\(\)](#), et [SH_InOutStateMachine : :toString\(\)](#).

Référencé par [SH_InOutStateMachine : :addChildsNextTransition\(\)](#), [SH_DateQuestionState : :rawInput\(\)](#), et [SH_InOutStateMachine : :toString\(\)](#).

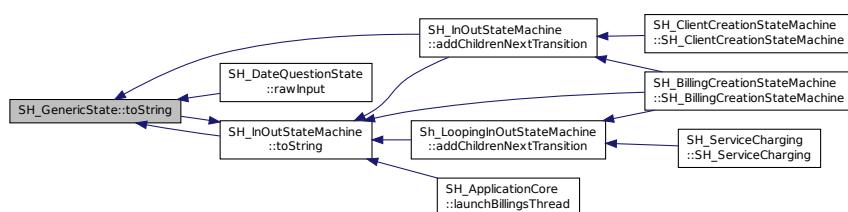
```

00022 {
00023     QStateMachine* machine = this->machine();
00024     SH_InOutStateMachine* mach = qobject_cast<
00025         SH_InOutStateMachine * >(machine);
00026     if(mach) {
00027         return SH_NamedObject::toString() + " [in "+mach->
00028             toString()+"] ";
00029     } else {
00030         return SH_NamedObject::toString();
00031     }
00030 }
```

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :



4.52.3.26 bool SH_InOutState::visibility() [inherited]

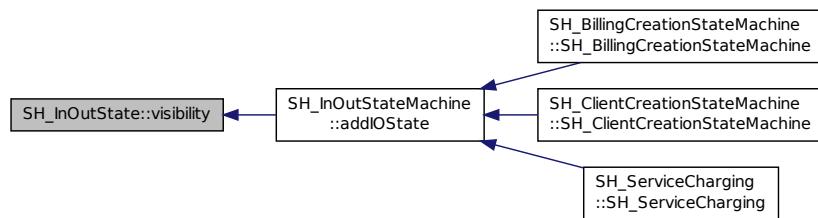
Définition à la ligne 91 du fichier [SH_IOState.cpp](#).

Références [SH_InOutState::m_isVisible](#).

Référencé par [SH_InOutStateMachine::addIOState\(\)](#).

```
00091     {  
00092         return m_isVisible;  
00093     }
```

Voici le graphe des appels de cette fonction :



4.52.4 Documentation des données membres

4.52.4.1 QRegularExpression SH_RegExpQuestionState::m_regexp [private]

m_regexp

Définition à la ligne 51 du fichier [SH_RegExpQuestionState.h](#).

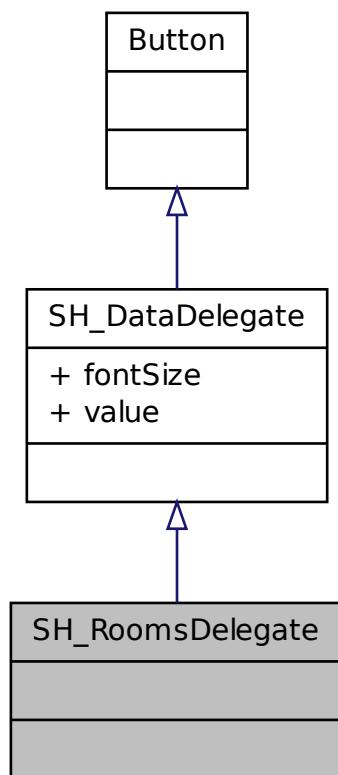
Référencé par [isAnswerValid\(\)](#), [regexp\(\)](#), et [setRegexp\(\)](#).

La documentation de cette classe a été générée à partir des fichiers suivants :

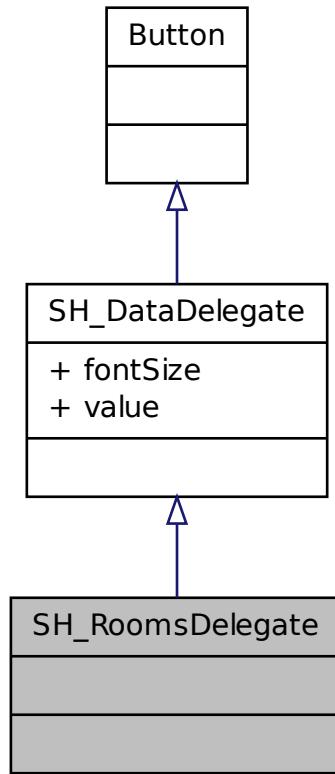
- logic/[SH_RegExpQuestionState.h](#)
- logic/[SH_RegExpQuestionState.cpp](#)

4.53 Référence de la classe SH_RoomsDelegate

Graphe d'héritage de SH_RoomsDelegate :



Graphe de collaboration de SH_RoomsDelegate :



Propriétés

- int `fontSize`
- string `value`

4.53.1 Description détaillée

Définition à la ligne 4 du fichier [SH_RoomsDelegate.qml](#).

4.53.2 Documentation des propriétés

4.53.2.1 int SH_DataDelegate ::fontSize [inherited]

Définition à la ligne 9 du fichier [SH_DataDelegate.qml](#).

4.53.2.2 string SH_DataDelegate ::value [inherited]

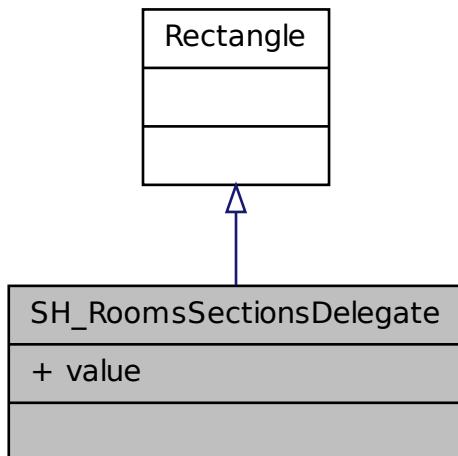
Définition à la ligne 7 du fichier [SH_DataDelegate.qml](#).

La documentation de cette classe a été générée à partir du fichier suivant :

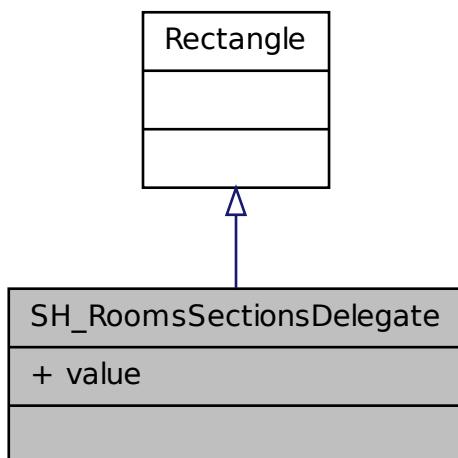
- [views/qml/SH_RoomsDelegate.qml](#)

4.54 Référence de la classe SH_RoomsSectionsDelegate

Graphe d'héritage de SH_RoomsSectionsDelegate :



Graphe de collaboration de SH_RoomsSectionsDelegate :



Propriétés

- string [value](#)

4.54.1 Description détaillée

Définition à la ligne 4 du fichier [SH_RoomsSectionsDelegate.qml](#).

4.54.2 Documentation des propriétés

4.54.2.1 string SH_RoomsSectionsDelegate::value

Définition à la ligne 7 du fichier [SH_RoomsSectionsDelegate.qml](#).

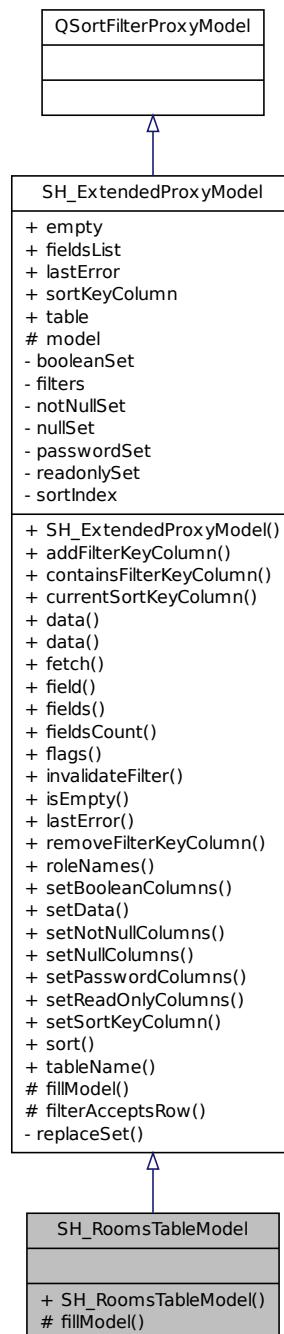
La documentation de cette classe a été générée à partir du fichier suivant :

– views/qml/[SH_RoomsSectionsDelegate.qml](#)

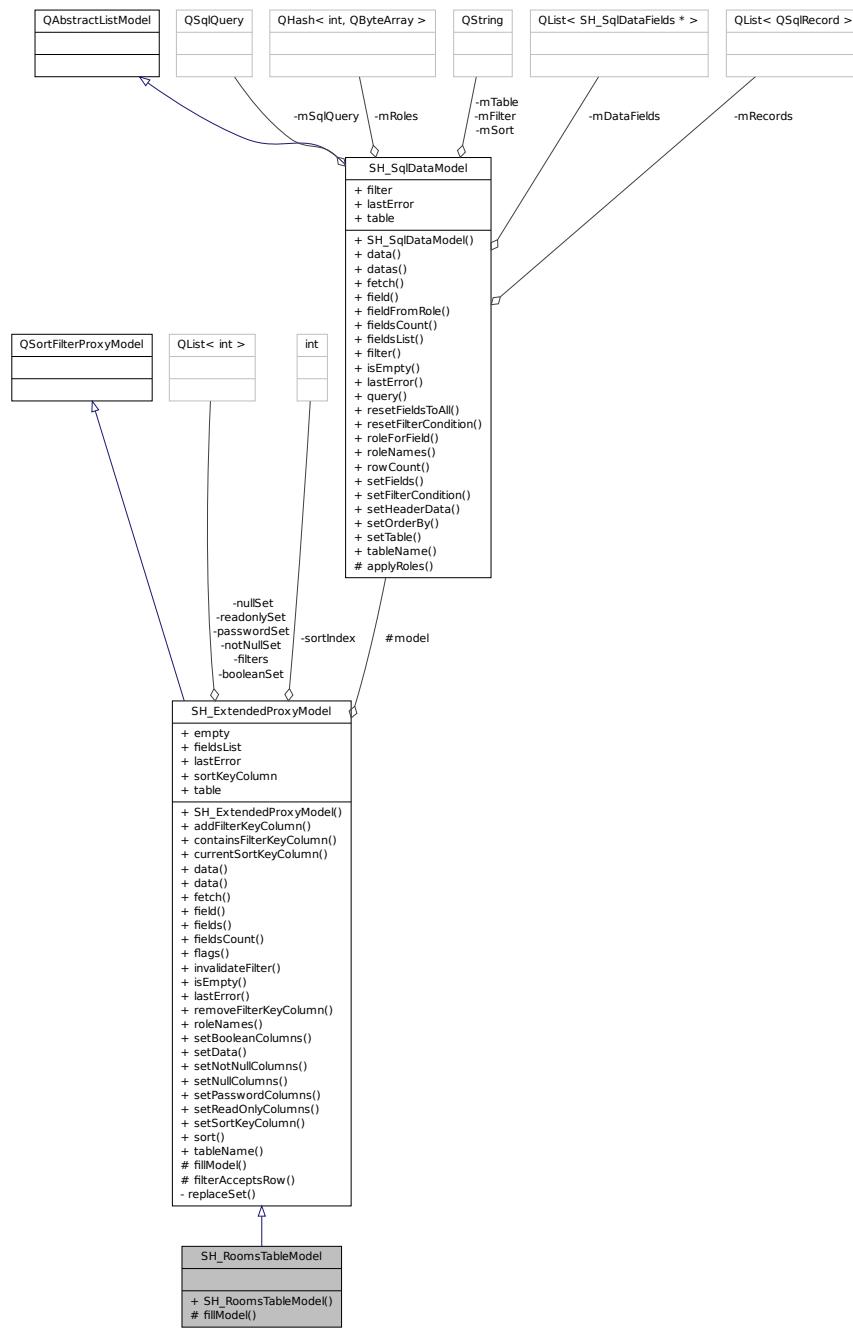
4.55 Référence de la classe SH_RoomsTableModel

```
#include <SH_RoomsTableModel.h>
```

Graphe d'héritage de SH_RoomsTableModel :



Graphe de collaboration de SH_RoomsTableModel :



Signaux

- void `sortChanged ()`

Fonctions membres publiques

- `SH_RoomsTableModel (QObject *parent=0)`
- `Q_INVOKABLE void addFilterKeyColumn (int column)`
- `Q_INVOKABLE bool containsFilterKeyColumn (int column)`
- `const int currentSortKeyColumn () const`

- Q_INVOKABLE QVariant **data** (int row, int column) const
- Q_INVOKABLE QVariant **data** (const QModelIndex &index, int role=Qt::DisplayRole) const
- Q_INVOKABLE bool **fetch** (QString **tableName**="", QString **filter**="", QString **sort**="", QStringList **fields**=QStringList())
- Q_INVOKABLE SH_SQLDataFields * **field** (int i) const
- const QString **fields** () const
- Q_INVOKABLE int **fieldsCount** () const
- Q_INVOKABLE Qt::ItemFlags **flags** (const QModelIndex &index) const
- void **invalidateFilter** ()
- const bool **isEmpty** () const
- const QString **lastError** () const
- Q_INVOKABLE void **removeFilterKeyColumn** (int column)
- virtual Q_INVOKABLE QHash<int, QByteArray> **roleNames** () const
- void **setBooleanColumns** (QList<int> **boolCols**)
- Q_INVOKABLE bool **setData** (const QModelIndex &index, const QVariant &value, int role=Qt::EditRole)
- void **setNotNullColumns** (QList<int> **notNullCols**)
- void **setNullColumns** (QList<int> **nullCols**)
- void **setPasswordColumns** (QList<int> **passwordCols**)
- void **setReadOnlyColumns** (QList<int> **readonlyCols**)
- void **setSortKeyColumn** (int column)
- Q_INVOKABLE void **sort** (int column, Qt::SortOrder newOrder=Qt::AscendingOrder)
- const QString **tableName** () const

Fonctions membres protégées

- virtual void **fillModel** ()
- bool **filterAcceptsRow** (int source_row, const QModelIndex &source_parent) const

Attributs protégés

- SH_SQLDataModel * **model**

Propriétés

- bool **empty**
- QString **fieldsList**
- QString **lastError**
- int **sortKeyColumn**
- QString **table**

4.55.1 Description détaillée

Définition à la ligne 16 du fichier [SH_RoomsTableModel.h](#).

4.55.2 Documentation des constructeurs et destructeur

4.55.2.1 SH_RoomsTableModel : :SH_RoomsTableModel (QObject * parent = 0)

Paramètres

<code>parent</code>	
---------------------	--

Définition à la ligne 10 du fichier [SH_RoomsTableModel.cpp](#).

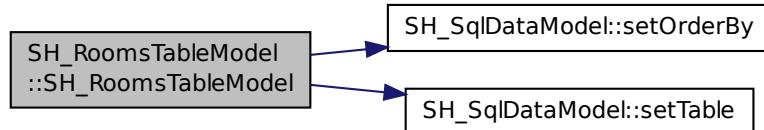
Références [SH_ExtendedProxyModel](#) : :model, [SH_SQLDataModel](#) : :setOrderBy(), et [SH_SQLDataModel](#) : :setTable().

```

00010 :
00011     SH_ExtendedProxyModel (parent)
00012 {
00013     SH_ExtendedProxyModel::model->setTable ("ROOMSINFOS");
00014     SH_ExtendedProxyModel::model->setOrderBy ("FLOOR ASC, NUMBER ASC")
00015 ;
}

```

Voici le graphe d'appel pour cette fonction :



4.55.3 Documentation des fonctions membres

4.55.3.1 void SH_ExtendedProxyModel : :addFilterKeyColumn (int column) [inherited]

Définition à la ligne 259 du fichier [SH_ExtendedProxyModel.cpp](#).

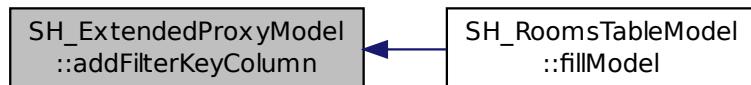
Références [SH_ExtendedProxyModel : :filters](#).

Référencé par [fillModel\(\)](#).

```

00260 {
00261     this->filters.append\(column\);
00262 }
```

Voici le graphe des appels de cette fonction :



4.55.3.2 bool SH_ExtendedProxyModel : :containsFilterKeyColumn (int column) [inherited]

Définition à la ligne 225 du fichier [SH_ExtendedProxyModel.cpp](#).

Références [SH_ExtendedProxyModel : :filters](#).

```

00226 {
00227     return this->filters.contains\(column\);
00228 }
```

4.55.3.3 const int SH_ExtendedProxyModel : :currentSortKeyColumn () const [inline], [inherited]

Définition à la ligne 38 du fichier [SH_ExtendedSqlProxyModel.h](#).

Références [SH_ExtendedProxyModel : :sortIndex](#).

```
00038 {return this->sortIndex;}
```

4.55.3.4 QVariant SH_ExtendedProxyModel ::data (int row, int column) const [inherited]

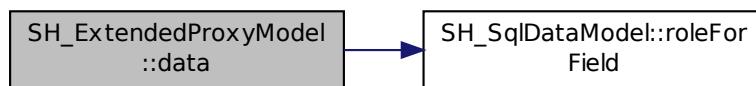
Définition à la ligne 269 du fichier [SH_ExtendedProxyModel.cpp](#).

Références [SH_ExtendedProxyModel ::model](#), et [SH_SqlDataModel ::roleForField\(\)](#).

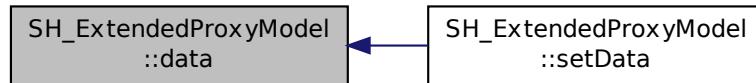
Référencé par [SH_ExtendedProxyModel ::setData\(\)](#).

```
00270 {
00271     QModelIndex modelIndex = this->index(row, 0);
00272     return this->data(modelIndex, this->model->roleForField(column));
00273 }
```

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appels de cette fonction :



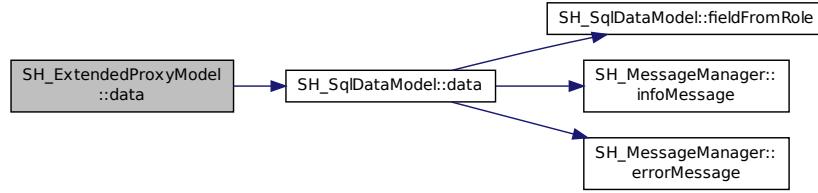
4.55.3.5 QVariant SH_ExtendedProxyModel ::data (const QModelIndex & index, int role = Qt ::DisplayRole) const [inherited]

Définition à la ligne 127 du fichier [SH_ExtendedProxyModel.cpp](#).

Références [SH_ExtendedProxyModel ::booleanSet](#), [SH_SqlDataModel ::data\(\)](#), [SH_ExtendedProxyModel ::filters](#), [SH_ExtendedProxyModel ::model](#), et [SH_ExtendedProxyModel ::passwordSet](#).

```
00128 {
00129     if (index.isValid())
00130     {
00131         if (this->booleanSet.contains(role))
00132         {
00133             return index.data(Qt::EditRole).toBool() ? QVariant(Qt::Checked) : QVariant(Qt::Unchecked);
00134         }
00135         else if (this->passwordSet.contains(role))
00136         {
00137             return QVariant("****");
00138         }
00139         else if (!this->filters.contains(role))
00140         {
00141             QModelIndex source_index = QSortFilterProxyModel::mapToSource(index);
00142             if (source_index.isValid())
00143                 return this->model->data(source_index, role);
00144             }
00145         }
00146     }
00147     return QVariant();
00148 }
```

Voici le graphe d'appel pour cette fonction :



4.55.3.6 bool SH_ExtendedProxyModel::fetch (QString tableName = "", QString filter = "", QString sort = "", QStringList fields = QStringList ()) [inherited]

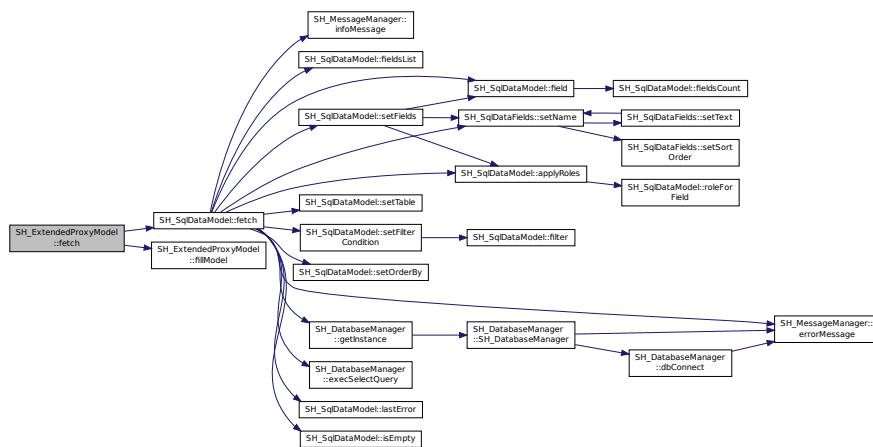
Définition à la ligne 280 du fichier [SH_ExtendedProxyModel.cpp](#).

Références [SH_SqlDataModel::fetch\(\)](#), [SH_ExtendedProxyModel::fillModel\(\)](#), et [SH_ExtendedProxyModel::model](#).

```

00281 {
00282     bool fetched = this->model->fetch(tableName, filter, sort,
00283         fields);
00284     if (fetched)
00285         this->fillModel();
00286     this->setSourceModel(this->model);
00287     return fetched;
00288 }
00289 }
```

Voici le graphe d'appel pour cette fonction :



4.55.3.7 Q_INVOKABLE SH_SqlDataFields* SH_ExtendedProxyModel::field (int i) const [inline], [inherited]

Définition à la ligne 82 du fichier [SH_ExtendedSqlProxyModel.h](#).

Références [SH_SqlDataModel::field\(\)](#), et [SH_ExtendedProxyModel::model](#).

```
00082 { return this->model->field(i); }
```

Voici le graphe d'appel pour cette fonction :



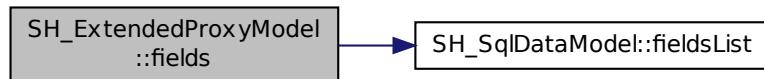
4.55.3.8 const QString SH_ExtendedProxyModel ::fields() const [inline], [inherited]

Définition à la ligne 52 du fichier [SH_ExtendedSqlProxyModel.h](#).

Références [SH_SqlDataModel ::fieldsList\(\)](#), et [SH_ExtendedProxyModel ::model](#).

```
00052 { if(this->model->fieldsList().isEmpty()){ return "*"; } else { return this->model->fieldsList().join(", "); } }
```

Voici le graphe d'appel pour cette fonction :



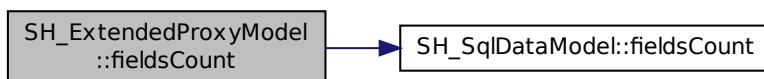
4.55.3.9 Q_INVOKABLE int SH_ExtendedProxyModel ::fieldsCount() const [inline], [inherited]

Définition à la ligne 89 du fichier [SH_ExtendedSqlProxyModel.h](#).

Références [SH_SqlDataModel ::fieldsCount\(\)](#), et [SH_ExtendedProxyModel ::model](#).

```
00089 { return this->model->fieldsCount(); }
```

Voici le graphe d'appel pour cette fonction :



4.55.3.10 void SH_RoomsTableModel::fillModel() [protected], [virtual]

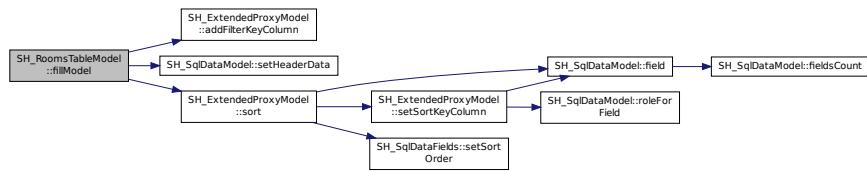
Implémente SH_ExtendedProxyModel.

Définition à la ligne 22 du fichier SH_RoomsTableModel.cpp.

Références SH_ExtendedProxyModel : :addFilterKeyColumn(), SH_ExtendedProxyModel : :model, SH_SqlDataModel : :setHeaderData(), et SH_ExtendedProxyModel : :sort().

```
00023 {
00024     SH_ExtendedProxyModel::model->setHeaderData(0, Qt::Horizontal,
00025     QObject::tr("ID"));
00026     SH_ExtendedProxyModel::model->setHeaderData(1, Qt::Horizontal,
00027     QObject::tr("Numéro de chambre"));
00028     SH_ExtendedProxyModel::model->setHeaderData(2, Qt::Horizontal,
00029     QObject::tr("Étage"));
00030     SH_ExtendedProxyModel::model->setHeaderData(3, Qt::Horizontal,
00031     QObject::tr("Type de chambre"));
00032     SH_ExtendedProxyModel::model->setHeaderData(4, Qt::Horizontal,
00033     QObject::tr("Détail"));
00034     SH_ExtendedProxyModel::model->setHeaderData(5, Qt::Horizontal,
00035     QObject::tr("Prix minimum"));
00036     SH_ExtendedProxyModel::model->setHeaderData(6, Qt::Horizontal,
00037     QObject::tr("Prix maximum"));
00038     SH_ExtendedProxyModel::sort(2,Qt::AscendingOrder);
00039     SH_ExtendedProxyModel::addFilterKeyColumn(0);
00040     SH_ExtendedProxyModel::addFilterKeyColumn(7);
00041 }
```

Voici le graphe d'appel pour cette fonction :



4.55.3.11 bool SH_ExtendedProxyModel::filterAcceptsRow (int source_row, const QModelIndex & source_parent) const [protected], [inherited]

Définition à la ligne 92 du fichier SH_ExtendedProxyModel.cpp.

Références SH_ExtendedProxyModel : :notNullSet, et SH_ExtendedProxyModel : :nullSet.

```
00093 {
00094     Q_UNUSED(source_parent);
00095
00096     if (!this->notNullSet.isEmpty())
00097     {
00098         QSqlQueryModel *m = static_cast<QSqlQueryModel *>(sourceModel());
00099         foreach(int column, this->notNullSet)
00100         {
00101             if (m->record(source_row).isNull(column))
00102             {
00103                 return false;
00104             }
00105         }
00106     }
00107
00108     if (!this->nullSet.isEmpty())
00109     {
00110         QSqlQueryModel *m = static_cast<QSqlQueryModel *>(sourceModel());
00111         foreach(int column, this->nullSet)
00112         {
00113             if (!m->record(source_row).isNull(column))
00114             {
00115                 return false;
00116             }
00117         }
00118     }
00119 }
```

```
00118     }
00119     return true;
00120 }
```

4.55.3.12 Qt::ItemFlags SH_ExtendedProxyModel::flags (const QModelIndex & index) const [inherited]

Définition à la ligne 179 du fichier [SH_ExtendedProxyModel.cpp](#).

Références [SH_ExtendedProxyModel::booleanSet](#), et [SH_ExtendedProxyModel::readonlySet](#).

```
00180 {
00181     if (!index.isValid())
00182     {
00183         return Qt::ItemIsEnabled;
00184     }
00185     if (!this->booleanSet.isEmpty())
00186     {
00187         return Qt::ItemIsUserCheckable | Qt::ItemIsSelectable | Qt::ItemIsEnabled;
00188     }
00189     else if (!this->readonlySet.isEmpty())
00190     {
00191         return Qt::ItemIsSelectable;
00192     }
00193     else
00194     {
00195         return QSortFilterProxyModel::flags(index);
00196     }
00197 }
00198 }
```

4.55.3.13 void SH_ExtendedProxyModel::invalidateFilter() [inherited]

Définition à la ligne 205 du fichier [SH_ExtendedProxyModel.cpp](#).

Références [SH_ExtendedProxyModel::filters](#).

```
00206 {
00207     this->filters.clear();
00208 }
```

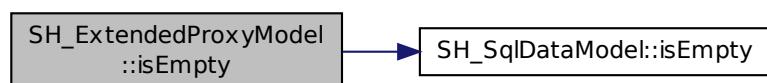
4.55.3.14 const bool SH_ExtendedProxyModel::isEmpty() const [inline], [inherited]

Définition à la ligne 66 du fichier [SH_ExtendedSqlProxyModel.h](#).

Références [SH_SqlDataManager::isEmpty\(\)](#), et [SH_ExtendedProxyModel::model](#).

```
00066 { return this->model->isEmpty(); }
```

Voici le graphe d'appel pour cette fonction :



4.55.3.15 const QString SH_ExtendedProxyModel::lastError() const [inline], [inherited]

Définition à la ligne 59 du fichier [SH_ExtendedSqlProxyModel.h](#).

Références [SH_SqlDataModel::lastError](#), et [SH_ExtendedProxyModel::model](#).

```
00059 { return this->model->lastError(); }
```

4.55.3.16 void SH_ExtendedProxyModel::removeFilterKeyColumn(int column) [inherited]

Définition à la ligne 215 du fichier [SH_ExtendedProxyModel.cpp](#).

Références [SH_ExtendedProxyModel::filters](#).

```
00216 {
00217     this->filters.removeAt(this->filters.indexOf(column));
00218 }
```

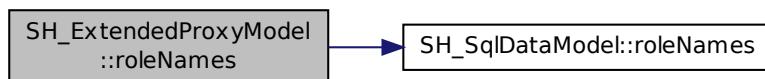
4.55.3.17 virtual Q_INVOKABLE QHash<int, QByteArray> SH_ExtendedProxyModel::roleNames() const [inline], [virtual], [inherited]

Définition à la ligne 165 du fichier [SH_ExtendedSqlProxyModel.h](#).

Références [SH_ExtendedProxyModel::model](#), et [SH_SqlDataModel::roleNames\(\)](#).

```
00165 { return this->model->roleNames(); }
```

Voici le graphe d'appel pour cette fonction :



4.55.3.18 void SH_ExtendedProxyModel::setBooleanColumns(QList< int > boolCols) [inherited]

Définition à la ligne 41 du fichier [SH_ExtendedProxyModel.cpp](#).

Références [SH_ExtendedProxyModel::booleanSet](#), et [SH_ExtendedProxyModel::replaceSet\(\)](#).

```
00041 {
00042     replaceSet(this->booleanSet, boolCols);
00043 }
```

Voici le graphe d'appel pour cette fonction :



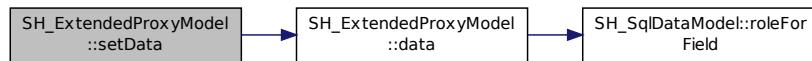
4.55.3.19 `bool SH_ExtendedProxyModel ::setData (const QModelIndex & index, const QVariant & value, int role = Qt :: EditRole) [inherited]`

Définition à la ligne 156 du fichier [SH_ExtendedProxyModel.cpp](#).

Références [SH_ExtendedProxyModel ::booleanSet](#), et [SH_ExtendedProxyModel ::data\(\)](#).

```
00157 {
00158     if (!index.isValid())
00159         return false;
00160
00161     if (this->booleanSet.contains(role))
00162     {
00163         QVariant data = (value.toInt() == Qt::Checked) ? QVariant(1) : QVariant(0);
00164         return QSortFilterProxyModel::setData(index, data, role);
00165     }
00166     else
00167     {
00168         return QSortFilterProxyModel::setData(index, value, role);
00169     }
00170 }
00171 }
```

Voici le graphe d'appel pour cette fonction :



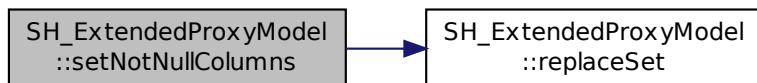
4.55.3.20 `void SH_ExtendedProxyModel ::setNotNullColumns (QList< int > notNullCols) [inherited]`

Définition à la ligne 80 du fichier [SH_ExtendedProxyModel.cpp](#).

Références [SH_ExtendedProxyModel ::notNullSet](#), et [SH_ExtendedProxyModel ::replaceSet\(\)](#).

```
00080
00081     if (sourceModel() ->inherits("QSqlQueryModel")) {
00082         replaceSet(this->notNullSet, notNullCols);
00083     }
00084 }
```

Voici le graphe d'appel pour cette fonction :



4.55.3.21 `void SH_ExtendedProxyModel ::setNullColumns (QList< int > nullCols) [inherited]`

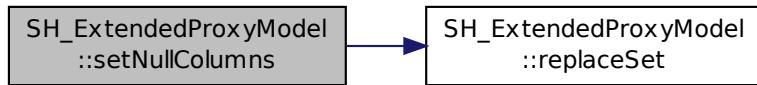
Définition à la ligne 68 du fichier [SH_ExtendedProxyModel.cpp](#).

Références [SH_ExtendedProxyModel ::nullSet](#), et [SH_ExtendedProxyModel ::replaceSet\(\)](#).

```

00068
00069     if (sourceModel() ->inherits("QSqlQueryModel")) {
00070         replaceSet(this->nullSet, nullCols);
00071     }
00072 }
```

Voici le graphe d'appel pour cette fonction :



4.55.3.22 void SH_ExtendedProxyModel ::setPasswordColumns (QList< int > passwordCols) [inherited]

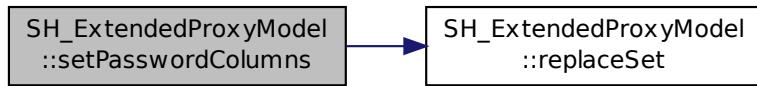
Définition à la ligne 59 du fichier [SH_ExtendedProxyModel.cpp](#).

Références [SH_ExtendedProxyModel ::passwordSet](#), et [SH_ExtendedProxyModel ::replaceSet\(\)](#).

```

00059
00060     replaceSet(this->passwordSet, passwordCols);
00061 }
```

Voici le graphe d'appel pour cette fonction :



4.55.3.23 void SH_ExtendedProxyModel ::setReadOnlyColumns (QList< int > readonlyCols) [inherited]

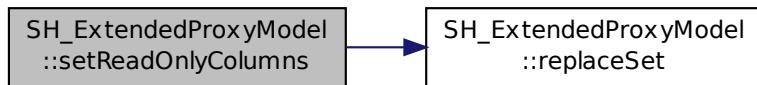
Définition à la ligne 50 du fichier [SH_ExtendedProxyModel.cpp](#).

Références [SH_ExtendedProxyModel ::readonlySet](#), et [SH_ExtendedProxyModel ::replaceSet\(\)](#).

```

00050
00051     replaceSet(this->readonlySet, readonlyCols);
00052 }
```

Voici le graphe d'appel pour cette fonction :



4.55.3.24 void SH_ExtendedProxyModel : :setSortKeyColumn (int column) [inherited]

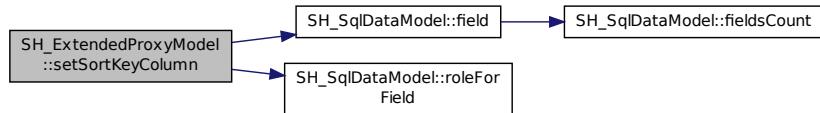
Définition à la ligne 246 du fichier [SH_ExtendedProxyModel.cpp](#).

Références [SH_SqlDataModel : :field\(\)](#), [SH_ExtendedProxyModel : :model](#), [SH_SqlDataModel : :roleForField\(\)](#), [SH_ExtendedProxyModel : :sortChanged\(\)](#), [SH_ExtendedProxyModel : :sortIndex](#), et [SH_SqlDataFields : :sortOrder](#).

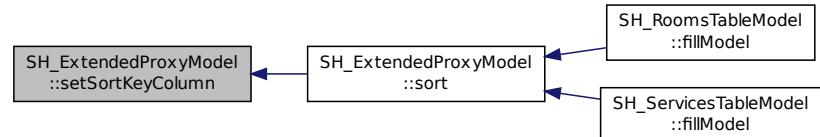
Référencé par [SH_ExtendedProxyModel : :sort\(\)](#).

```
00247 {
00248     this->sortIndex = column;
00249     QSortFilterProxyModel::setSortRole(this->model->roleForField(column));
00250     QSortFilterProxyModel::sort(0, this->model->field(column)->
00251         sortOrder());
00251     emit sortChanged();
00252 }
```

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :



4.55.3.25 void SH_ExtendedProxyModel : :sort (int column, Qt : :SortOrder newOrder = Qt : :AscendingOrder) [inherited]

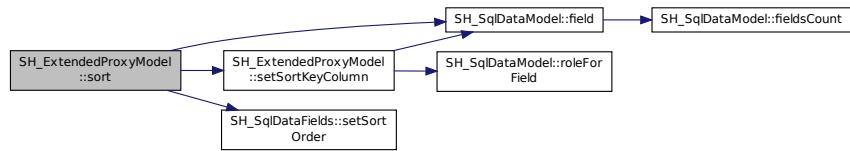
Définition à la ligne 235 du fichier [SH_ExtendedProxyModel.cpp](#).

Références [SH_SqlDataModel : :field\(\)](#), [SH_ExtendedProxyModel : :model](#), [SH_ExtendedProxyModel : :setSortKeyColumn\(\)](#), et [SH_SqlDataFields : :setSortOrder\(\)](#).

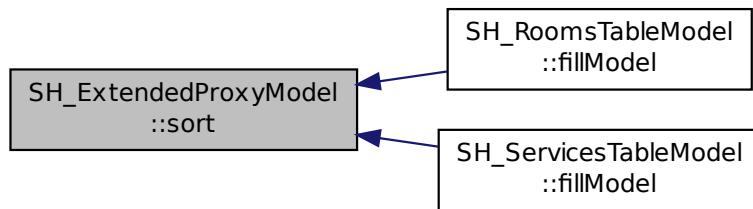
Référencé par [fillModel\(\)](#), et [SH_ServicesTableModel : :fillModel\(\)](#).

```
00236 {
00237     this->model->field(column)->setSortOrder(newOrder);
00238     SH_ExtendedProxyModel::setSortKeyColumn(column);
00239 }
```

Voici le graphe d'appel pour cette fonction :



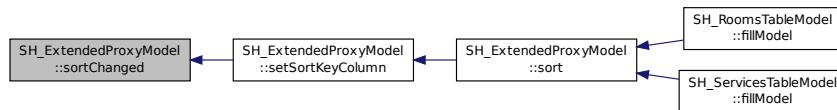
Voici le graphe des appels de cette fonction :



4.55.3.26 void SH_ExtendedProxyModel ::sortChanged() [signal], [inherited]

Référencé par [SH_ExtendedProxyModel ::setSortKeyColumn\(\)](#).

Voici le graphe des appels de cette fonction :



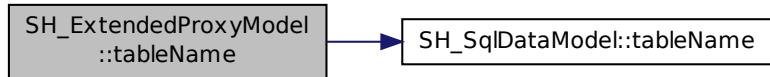
4.55.3.27 const QString SH_ExtendedProxyModel ::tableName() const [inline], [inherited]

Définition à la ligne 45 du fichier [SH_ExtendedSqlProxyModel.h](#).

Références [SH_ExtendedProxyModel ::model](#), et [SH_SqlDataModel ::tableName\(\)](#).

```
00045 { return this->model->tableName(); }
```

Voici le graphe d'appel pour cette fonction :



4.55.4 Documentation des données membres

4.55.4.1 `SH_SqlDataModel* SH_ExtendedProxyModel::model` [protected], [inherited]

Définition à la ligne 241 du fichier [SH_ExtendedSqlProxyModel.h](#).

Référencé par `SH_ExtendedProxyModel::data()`, `SH_ExtendedProxyModel::fetch()`, `SH_ExtendedProxyModel::field()`, `SH_ExtendedProxyModel::fields()`, `SH_ExtendedProxyModel::fieldsCount()`, `SH_BillingsTableModel::fillModel()`, `fillModel()`, `SH_BookingsTableModel::fillModel()`, `SH_ExtendedProxyModel::isEmpty()`, `SH_ExtendedProxyModel::lastError()`, `SH_ExtendedProxyModel::roleNames()`, `SH_ExtendedProxyModel::setSortKeyColumn()`, `SH_BillingsTableModel::SH_BillingsTableModel()`, `SH_BillsTableModel::SH_BillsTableModel()`, `SH_BookingsTableModel::SH_BookingsTableModel()`, `SH_ClientsTableModel::SH_ClientsTableModel()`, `SH_ExtendedProxyModel::SH_ExtendedProxyModel()`, `SH_GroupsTableModel::SH_GroupsTableModel()`, `SH_RoomsTableModel::SH_RoomsTableModel()`, `SH_ServicesTableModel::SH_ServicesTableModel()`, `SH_ExtendedProxyModel::sort()`, et `SH_ExtendedProxyModel::tableName()`.

4.55.5 Documentation des propriétés

4.55.5.1 `bool SH_ExtendedProxyModel::empty` [read], [inherited]

Définition à la ligne 21 du fichier [SH_ExtendedSqlProxyModel.h](#).

4.55.5.2 `QString SH_ExtendedProxyModel::fieldsList` [read], [inherited]

Définition à la ligne 18 du fichier [SH_ExtendedSqlProxyModel.h](#).

4.55.5.3 `QString SH_ExtendedProxyModel::lastError` [read], [inherited]

Définition à la ligne 19 du fichier [SH_ExtendedSqlProxyModel.h](#).

4.55.5.4 `int SH_ExtendedProxyModel::sortKeyColumn` [read], [write], [inherited]

Définition à la ligne 20 du fichier [SH_ExtendedSqlProxyModel.h](#).

4.55.5.5 `QString SH_ExtendedProxyModel::table` [read], [inherited]

Définition à la ligne 17 du fichier [SH_ExtendedSqlProxyModel.h](#).

La documentation de cette classe a été générée à partir des fichiers suivants :

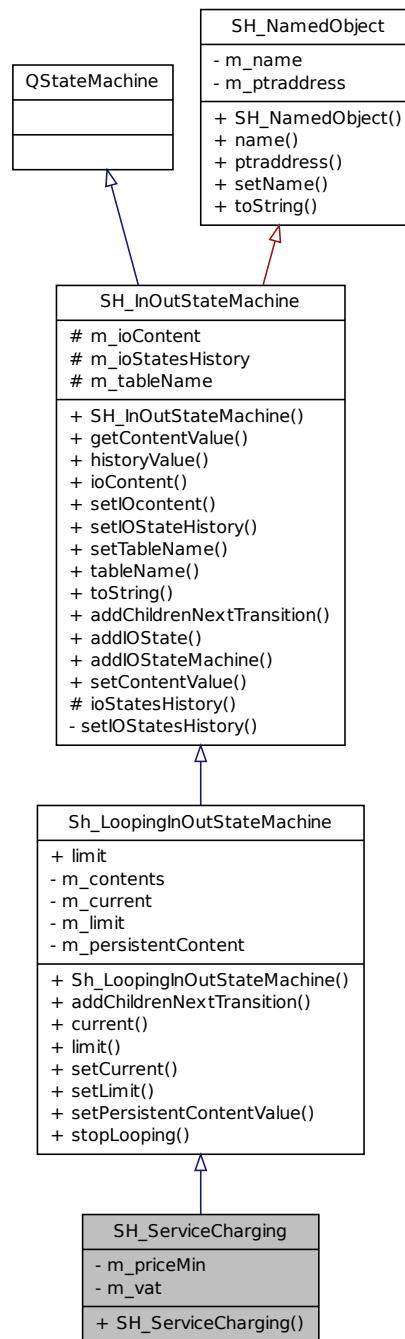
- models/[SH_RoomsTableModel.h](#)
- models/[SH_RoomsTableModel.cpp](#)

4.56 Référence de la classe SH_ServiceCharging

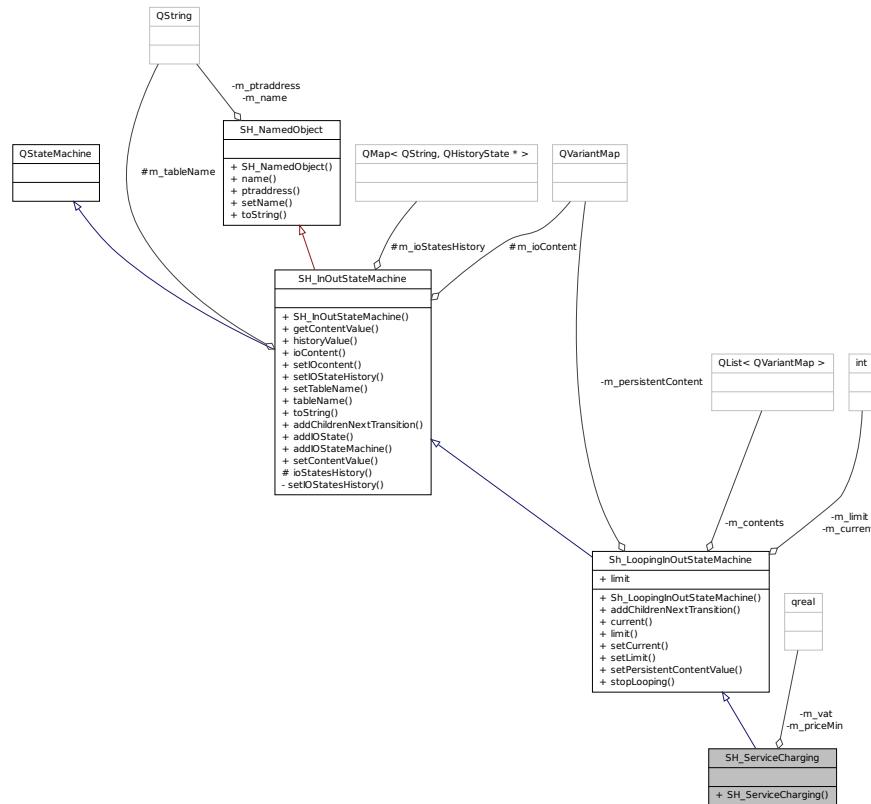
The [SH_ServiceCharging](#) class.

```
#include <SH_ServiceCharging.h>
```

Graphe d'héritage de SH_ServiceCharging :



Graphe de collaboration de SH_ServiceCharging :



Connecteurs publics

- void `addIOState (SH_InOutState *state, QString field)`
- void `addIOSMachine (SH_InOutStateMachine *fsm)`
- void `setContentValue (QVariant content, QString field)`

Signaux

- void `cancelReplacement ()`
- void `clearAll ()`
- void `confirmInput ()`
- void `displayCalendar ()`
- void `displayFileDialog ()`
- void `limitChanged ()`
- void `next ()`
- void `receiveInput (QString input)`
- void `replaceInput (QString field)`
- void `resendText (QString text, bool editable=false)`
- void `sendText (QString text, bool editable=false)`
- void `validateInput ()`

Fonctions membres publiques

- `SH_ServiceCharging (QString name, QObject *parent=0)`
 SH_ServiceCharging
- void `addChildrenNextTransition (QAbstractState *previousState, QAbstractState *nextState)`
- int `current () const`
- QVariant `getContentValue (QString field)`
- QHistoryState * `historyValue (QString field)`
- QVariantMap `ioContent () const`

- int `limit () const`
- void `setCurrent (int current)`
- void `setIOContent (const QVariantMap &ioContent)`
- void `setIOSTateHistory (QHistoryState *state, QString field)`
- void `setLimit (int limit)`
- void `setPersistentContentValue (QVariant value, QString field)`
- void `setTableName (const QString &tableName)`
- void `stopLooping ()`
- QString `tableName () const`
- QString `toString ()`

Fonctions membres protégées

- QMap< QString, QHistoryState * > `ioStatesHistory () const`

Attributs protégés

- QVariantMap `m_ioContent`
`m_ioContent`
- QMap< QString, QHistoryState * > `m_ioStatesHistory`
`m_ioStatesHistory`
- QString `m_tableName`
`m_tableName`

Propriétés

- int `limit`

Attributs privés

- qreal `m_priceMin`
`m_priceMin`
- qreal `m_vat`
`m_vat`

4.56.1 Description détaillée

The `SH_ServiceCharging` class.

Définition à la ligne 8 du fichier `SH_ServiceCharging.h`.

4.56.2 Documentation des constructeurs et destructeur

4.56.2.1 `SH_ServiceCharging ::SH_ServiceCharging (QString name, QObject * parent = 0)`

`SH_ServiceCharging`.

Paramètres

<code>name</code>	
<code>parent</code>	<code>SH_ServiceCharging ::SH_ServiceCharging</code>

Définition à la ligne 12 du fichier `SH_ServiceCharging.cpp`.

Références `Sh_LoopingInOutStateMachine ::addChildrenNextTransition()`, `SH_InOutStateMachine ::addIOSState()`, `SH_QuestionState ::answerInvalid()`, `SH_QuestionState ::answerValid()`, `SH_InOutStateMachine ::confirmInput()`, `SH_DatabaseManager ::execSelectQuery()`, `SH_DatabaseManager ::getInstance()`, `m_priceMin`, `m_vat`, `SH_NamedObject ::name()`, `SH_GenericState ::next()`, `SH_DatabaseContentQuestionState ::rawInput()`, `SH_InOutStateMachine ::receiveInput()`, `SH_QuestionState ::setInput()`, `SH_InOutState ::setInput()`, `SH_DatabaseContent-`

QuestionState : `:setOutput()`, SH_InOutState : `:setOutput()`, SH_InOutState : `:setVisibility()`, Sh_LoopingInOutStateMachine : `:stopLooping()`, et SH_InOutStateMachine : `:validateInput()`.

```

00012
00013     Sh_LoopingInOutStateMachine("CHARGEDSERVICES",
00014     name, 0, parent), m_priceMin(0.0)
00015 {
00016     SH_DatabaseContentQuestionState* service = new
00017     SH_DatabaseContentQuestionState("Veuillez sélectionner une prestation ou
00018     appuyer sur la touche \"VALIDER\" pour cesser d'ajouter des prestations", "choose service in service charging",
00019     "SERVICES", "CODE");
00020     SH_InOutState*serviceId = new SH_InOutState("", "service id in service
00021     charging");
00022     serviceId->setVisibility(false);
00023     SH_StringQuestionState* serviceName = new
00024     SH_StringQuestionState("Veuillez entrer ce qui sera affiché sur la facture", "service
00025     name in service charging", 1);
00026     SH_DecimalQuestionState* price = new
00027     SH_DecimalQuestionState("", "price in service charging", -Q_INFINITY, Q_INFINITY);
00028     SH_DecimalQuestionState* quantity = new
00029     SH_DecimalQuestionState("", "quantity in service charging", 1);
00030     SH_DatabaseContentQuestionState* vat = new
00031     SH_DatabaseContentQuestionState("", "vat in service charging", "TAXES", "
00032     PERCENTAGE", "ENABLED='1'");
00033     QFinalState* final = new QFinalState();
00034
00035     connect(service, &SH_QuestionState::answerInvalid, [=] () {
00036         int in = service->rawInput().toInt();
00037         if(in == -1 || in == 0) {
00038             emit service->next();
00039         }
00040     });
00041     connect(service, &SH_QuestionState::answerValid, [=] () {
00042         if(service->rawInput().toInt() > -1) {
00043             QString name;
00044             QStringList list;
00045             list.append("PRINTEDNAME");
00046             list.append("PRICEMIN");
00047             list.append("PRICEMAX");
00048             list.append("VAT_PERCENTAGE");
00049             list.append("ID");
00050             QSqlQuery result = SH_DatabaseManager::getInstance()->
00051             execSelectQuery("SERVICESINFOS", list, QString("CODE=%1").arg(service->
00052             rawInput().toString()));
00053             result.next();
00054             QSqlRecord record = result.record();
00055             name= record.value(0).toString();
00056             m_priceMin =record.value(1).toDouble();
00057             m_vat =record.value(3).toDouble();
00058             serviceId->setInput(record.value(4).toInt());
00059             serviceName->setInput(name);
00060             price->setOutput(QString("Le prix proposé pour cette prestation est : %1. Son prix
00061             minimum est %1 et son prix maximum %2.\nVeuillez entrer un nouveau prix ou appuyer sur la touche \"CONFIRMER\""
00062             ).arg(record.value(1).toString()).arg(record.value(2).toString()));
00063             vat->setOutput(QString("Cette prestation est associée à une TVA de %1%.\nVeuillez
00064             entrer une autre TVA à appliquer ou appuyer sur la touche \"CONFIRMER\"").arg(record.value(3).toString()));
00065             serviceName->setVisibility(false);
00066         }
00067     });
00068     connect(quantity, &QState::entered, [=] () {
00069         connect(this, &SH_InOutStateMachine::receiveInput, [=] (QString in
00070     ) {
00071         QString newInput;
00072         if(in.right(in.length() - 1).toInt() != 0) {
00073             newInput = in.right(in.length() - 1);
00074             emit receiveInput(newInput);
00075         }
00076     });
00077     connect(price, &QState::entered, [=] () {
00078         connect(this, &SH_InOutStateMachine::confirmInput, [=] () {
00079             price->setInput(m_priceMin);
00080         });
00081     });
00082     connect(vat, &QState::entered, [=] () {
00083         connect(this, &SH_InOutStateMachine::confirmInput, [=] () {
00084             vat->setInput(m_vat);
00085         });
00086     });
00087     this->addState(final);
00088     this->ADDIOState(service, "");
00089     this->ADDIOState(serviceId, "SERVICE_ID");
00090     this->ADDIOState(serviceName, "PRINTEDNAME");

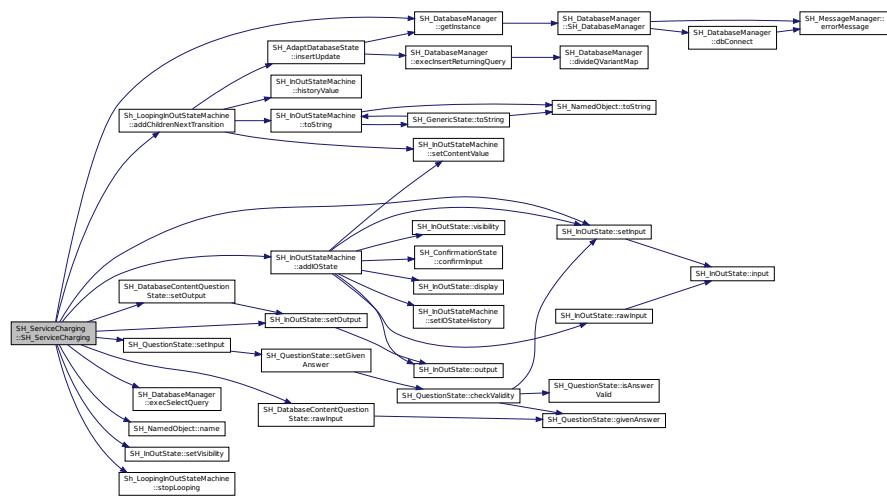
```

```

00078     this->addIOState(price, "CHARGEDUNITPRICE");
00079     this->addIOState(quantity, "QUANTITY");
00080     this->addIOState(vat, "CHARGEDVAT");
00081     this->addChildrenNextTransition(service, serviceId);
00082     this->addChildrenNextTransition(serviceId, serviceName);
00083     this->addChildrenNextTransition(serviceName, quantity);
00084     this->addChildrenNextTransition(quantity, price);
00085     this->addChildrenNextTransition(price, vat);
00086     this->addChildrenNextTransition(vat, final);
00087     this->setInitialState(service);
00088     connect(this, &SH_InOutStateMachine::validateInput, this, &
00089             Sh_LoopingInOutStateMachine::stopLooping);
00089 }

```

Voici le graphe d'appel pour cette fonction :



4.56.3 Documentation des fonctions membres

4.56.3.1 void Sh_LoopingInOutStateMachine ::addChildrenNextTransition (QAbstractState * *previousState*, QAbstractState * *nextState*) [inherited]

Définition à la ligne 86 du fichier `SH_LoopingIOStateMachine.cpp`.

Références `SH_InOutStateMachine ::historyValue()`, `SH_AdaptDatabaseState ::insertUpdate()`, `Sh_LoopingInOutStateMachine ::m_contents`, `Sh_LoopingInOutStateMachine ::m_current`, `SH_InOutStateMachine ::m_ioContent`, `Sh_LoopingInOutStateMachine ::m_limit`, `Sh_LoopingInOutStateMachine ::m_persistentContent`, `SH_InOutStateMachine ::m_tableName`, `SH_InOutStateMachine ::next()`, `SH_InOutStateMachine ::replaceInput()`, `SH_InOutStateMachine ::setContentValue()`, et `SH_InOutStateMachine ::toString()`.

Référencé par `SH_BillingCreationStateMachine ::SH_BillingCreationStateMachine()`, et `SH_ServiceCharging()`.

```

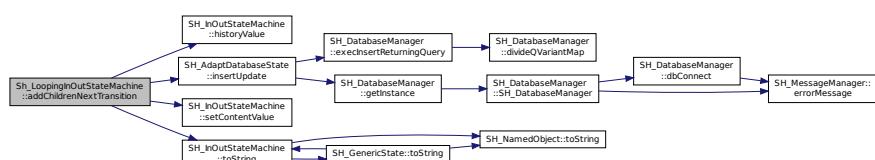
00087 {
00088     SH_GenericState* genPreviousState = qobject_cast<
00089         SH_GenericState*>(previousState);
00090     SH_InOutStateMachine* fsmPreviousState = qobject_cast<
00091         SH_InOutStateMachine*>(previousState);
00092     QFinalState* final = qobject_cast<QFinalState*>(nextState);
00093     if(final) {
00094         /* à faire au moment de l'entrée dans l'état previousState */
00095         connect(previousState, &QAbstractState::entered, [=]() {
00096             m_current++;
00097             m_contents.append(m_ioContent);
00098             m_ioContent.clear();
00099             m_ioContent = m_persistentContent;
00100             if(m_limit == 0 || m_current < m_limit) {
00101                 if(genPreviousState) {
00102                     connect(genPreviousState, &QAbstractState::entered, [=]() {
00103                         genPreviousState->addTransition(genPreviousState, SIGNAL(
00104                             next()), initialState());
00105                     });
00106                 }
00107             }
00108         });
00109     }
00110 }

```

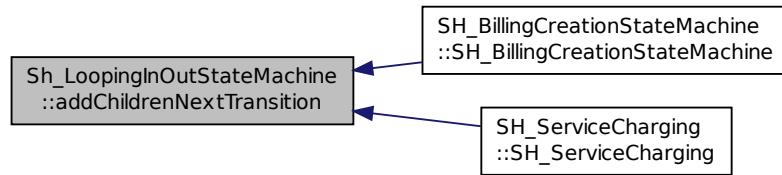
```

00102                     );
00103                 }
00104             if(fsmPreviousState) {
00105                 connect(fsmPreviousState, &QAbstractState::entered, [=]() {
00106                     fsmPreviousState->addTransition(fsmPreviousState, SIGNAL(
00107                         next()), initialState());
00108                 });
00109             } else {
00110                 SH_AdaptDatabaseState* nextSaveState = new
00111                     SH_AdaptDatabaseState("enregistrement 0 de la machine "+
00112                         toString());
00113                 if(genPreviousState) {
00114                     genPreviousState->addTransition(genPreviousState, SIGNAL(
00115                         next()), nextSaveState);
00116                 }
00117                 if(fsmPreviousState) {
00118                     fsmPreviousState->addTransition(fsmPreviousState, SIGNAL(
00119                         next()), nextSaveState);
00120                 }
00121                 if(genPreviousState || fsmPreviousState) {
00122                     for(int i = 1; i < m_limit; i++) {
00123                         SH_AdaptDatabaseState* saveState = nextSaveState;
00124                         nextSaveState = new SH_AdaptDatabaseState(QString(
00125                             "enregistrement %1 de la machine %2").arg(QString::number(i)).arg(toString()));
00126                         saveState->addTransition(saveState, SIGNAL(next()), nextSaveState);
00127                         connect(saveState, &QAbstractState::exited, [=]() {
00128                             connect(nextSaveState, &QAbstractState::entered, [=]() {
00129                                 setContentValue(nextSaveState->
00130                                     insertUpdate(m_tableName, m_contents[i]), "ID");
00131                             });
00132                         });
00133                     }
00134                 }
00135             }
00136             if(fsmPreviousState) {
00137                 fsmPreviousState->addTransition(fsmPreviousState, SIGNAL(next()), nextState);
00138             }
00139         }
00140         if(genPreviousState) {
00141             /*à faire au moment de l'entrée dans l'état previousState*/
00142             connect(genPreviousState, &QAbstractState::entered, [=]() {
00143                 connect(this, &SH_InOutStateMachine::replaceInput, [=]()
00144                     QString field) {
00145                     /*après avoir demandé à revenir sur un état précédent, on attend la fin de l'état actuel
00146                     puis on retourne à l'historique de l'état désiré; celui-ci fini, on passe à l'état qui aurait du suivre celui
00147                     pendant lequel on a demandé à revenir sur un état précédent*/
00148                     QHistoryState* hState = historyValue(field);
00149                     if(hState) { /*si l'historique existe (on a déjà quitté l'état voulu)*/
00150                         hState->parentState()->addTransition(hState->parentState(), SIGNAL(
00151                             next()), nextState);
00152                         genPreviousState->addTransition(genPreviousState, SIGNAL(
00153                             next()), hState);
00154                     };
00155                 });
00156             }
00157         }
00158     }
00159 }
```

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :



4.56.3.2 void SH_InOutStateMachine ::addIOState (SH_InOutState * state, QString field) [slot], [inherited]

Définition à la ligne 110 du fichier [SH_IOStateMachine.cpp](#).

Références [SH_ConfirmationState ::confirmInput\(\)](#), [SH_InOutState ::display\(\)](#), [SH_InOutStateMachine ::displayCalendar\(\)](#), [SH_InOutStateMachine ::displayFileDialog\(\)](#), [SH_InOutState ::output\(\)](#), [SH_InOutState ::rawInput\(\)](#), [SH_InOutStateMachine ::receiveInput\(\)](#), [SH_InOutState ::resendInput\(\)](#), [SH_InOutStateMachine ::resendText\(\)](#), [SH_InOutState ::sendOutput\(\)](#), [SH_InOutStateMachine ::sendText\(\)](#), [SH_InOutStateMachine ::setContentValue\(\)](#), [SH_InOutState ::setInput\(\)](#), [SH_InOutStateMachine ::setIOStateHistory\(\)](#), [SH_InOutStateMachine ::validateInput\(\)](#), et [SH_InOutState ::visibility\(\)](#).

Référencé par [SH_BillingCreationStateMachine ::SH_BillingCreationStateMachine\(\)](#), [SH_ClientCreationStateMachine ::SH_ClientCreationStateMachine\(\)](#), et [SH_ServiceCharging\(\)](#).

```

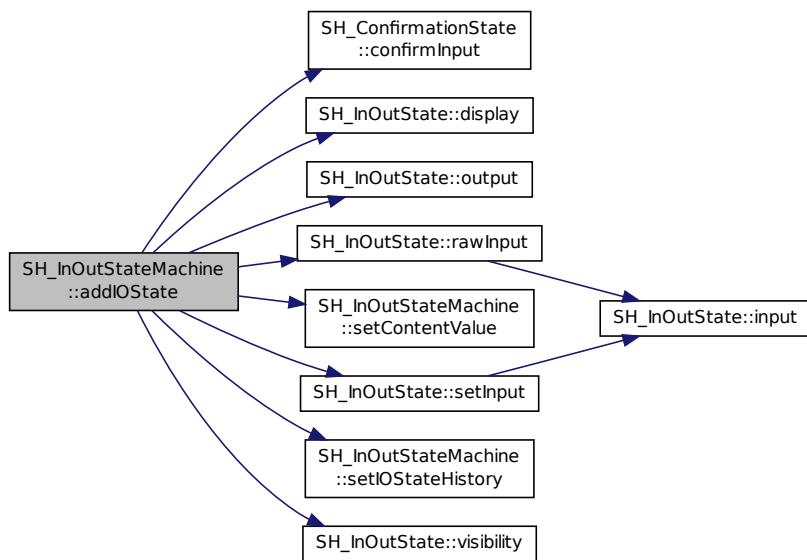
00111 {
00112     /*à faire au moment de l'entrée dans l'état state*/
00113     connect(state, &QState::entered, [=]() {
00114         qDebug() << "entered !";
00115         state->display(true);
00116         connect(this, &SH_InOutStateMachine::receiveInput, state, &
00117             SH_InOutState::setInput); /* la réception d'une valeur entraîne son enregistrement
00118             comme entrée de l'utilisateur auprès de l'état*/
00119         connect(this, &SH_InOutStateMachine::receiveInput, [=](QString in
00120         ) { qDebug() << "Hello world !"; state->setInput(in);}); /* la réception d'une valeur entraîne son
00121             enregistrement comme entrée de l'utilisateur auprès de l'état*/
00122         connect(state, &SH_InOutState::sendOutput, [=](QVariant out) {qDebug() <<
00123             "connected !"; emit this->sendText(out.toString(), false);});
00124         connect(state, &SH_InOutState::resendInput, [=](QVariant in) {emit this->
00125             resendText(in.toString(), true);});
00126         if(state->visibility()) {
00127             state->sendOutput(QVariant(state->output()));
00128         } else {
00129             qDebug() << "invisible";
00130         }
00131     });
00132     SH_ValidationState *validationState = qobject_cast<
00133         SH_ValidationState*>(state);
00134     if(validationState) {
00135         /*à faire au moment de l'entrée dans l'état state*/
00136         connect(validationState, &QState::entered, [=]() {
00137             connect(this, &SH_InOutStateMachine::validateInput,
00138                 validationState, &SH_ValidationState::confirmInput);
00139         });
00140     }
00141     SH_ConfirmationState *confirmationState = qobject_cast<
00142         SH_ConfirmationState*>(state);
00143     if(confirmationState) {
00144         /*à faire au moment de l'entrée dans l'état state*/
00145         connect(confirmationState, &QState::entered, [=]() {
00146             connect(this, &SH_InOutStateMachine::validateInput,
00147                 confirmationState, &SH_ConfirmationState::confirmInput);
00148         });
00149     }
00150     SH_DateQuestionState *dateState = qobject_cast<
00151         SH_DateQuestionState*>(state);
00152     if(dateState) {
00153         /*à faire au moment de l'entrée dans l'état state*/
00154         connect(dateState, &QState::entered, this, &
  
```

```

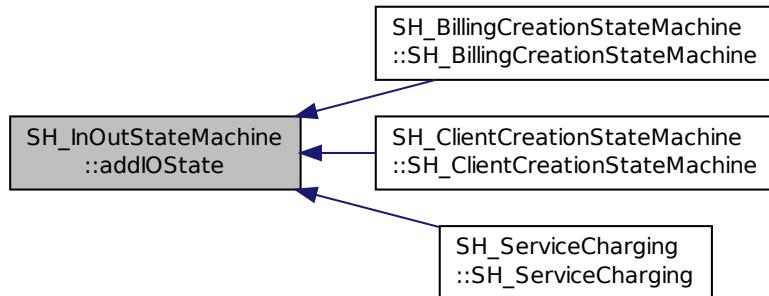
00144     }
00145     SH_FileSelectionState *fileState = qobject_cast<
00146         SH_FileSelectionState*>(state);
00147     if(fileState) {
00148         /*à faire au moment de l'entrée dans l'état state*/
00149         connect(fileState, &QState::entered, this, &
00150             SH_InOutStateMachine::displayFileDialog);
00151     }
00152     /*à faire au moment de la sortie de l'état state*/
00153     connect(state, &QState::exited, [=]() {
00154         qDebug() << "exited !";
00155         if(!field.isEmpty()) {
00156             setContentValue(state->rawInput(), field);
00157             /*gestion de l'historique des états pour pouvoir revenir à l'état state après l'avoir quitté*/
00158             QHistoryState* hState = new QHistoryState(state);
00159             setIOStateHistory(hState, field);
00160         }
00161         state->disconnect(this); /*plus aucune action sur l'état ne pourra être provoquée par la machine*/
00162     });
00163     QAbstractState* astate = qobject_cast<QAbstractState *>(state);
00164     if(astate) {
00165         addState(astate);
00166     }
00167 }

```

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :



4.56.3.3 void SH_InOutStateMachine ::addIOStateMachine (SH_InOutStateMachine * fsm) [slot], [inherited]

Définition à la ligne 175 du fichier [SH_IOSMachine.cpp](#).

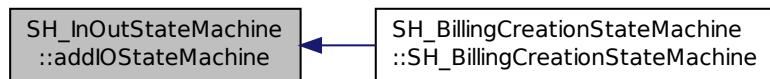
Références [SH_InOutStateMachine ::cancelReplacement\(\)](#), [SH_InOutStateMachine ::confirmInput\(\)](#), [SH_InOutStateMachine ::displayCalendar\(\)](#), [SH_InOutStateMachine ::receiveInput\(\)](#), [SH_InOutStateMachine ::replaceInput\(\)](#), [SH_InOutStateMachine ::resendText\(\)](#), [SH_InOutStateMachine ::sendText\(\)](#), et [SH_InOutStateMachine ::validateInput\(\)](#).

Référencé par [SH_BillingCreationStateMachine ::SH_BillingCreationStateMachine\(\)](#).

```

00176 {
00177     /*à faire au moment de l'entrée dans la machine d'état fsm*/
00178     connect(fsm, &QState::entered, [=]() {
00179         connect(this, &SH_InOutStateMachine::receiveInput, fsm, &
00180             SH_InOutStateMachine::receiveInput);
00181         connect(this, &SH_InOutStateMachine::sendText, fsm, &
00182             SH_InOutStateMachine::sendText);
00183         connect(this, &SH_InOutStateMachine::resendText, fsm, &
00184             SH_InOutStateMachine::resendText);
00185         connect(this, &SH_InOutStateMachine::confirmInput, fsm, &
00186             SH_InOutStateMachine::confirmInput);
00187         connect(this, &SH_InOutStateMachine::validateInput, fsm, &
00188             SH_InOutStateMachine::validateInput);
00189         connect(this, &SH_InOutStateMachine::replaceInput, fsm, &
00190             SH_InOutStateMachine::replaceInput);
00191         connect(this, &SH_InOutStateMachine::cancelReplacement, fsm,
00192             &SH_InOutStateMachine::cancelReplacement);
00193         connect(this, &SH_InOutStateMachine::displayCalendar, fsm, &
00194             SH_InOutStateMachine::displayCalendar);
00195     });
00196     /*à faire au moment de la sortie de la machine d'état fsm*/
00197     connect(fsm, &QState::exited, [=]() {
00198         fsm->disconnect(this); /*plus aucune action sur la machine d'état fille ne pourra être provoquée
00199         par la machine mère*/
00200     });
00201 }
  
```

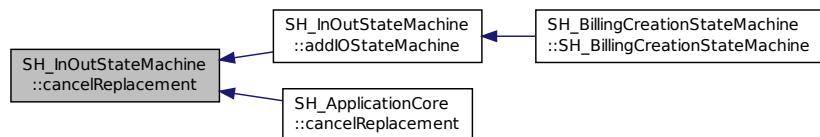
Voici le graphe des appels de cette fonction :



4.56.3.4 void SH_InOutStateMachine ::cancelReplacement() [signal], [inherited]

Référencé par [SH_InOutStateMachine ::addIOStateMachine\(\)](#), et [SH_ApplicationCore ::cancelReplacement\(\)](#).

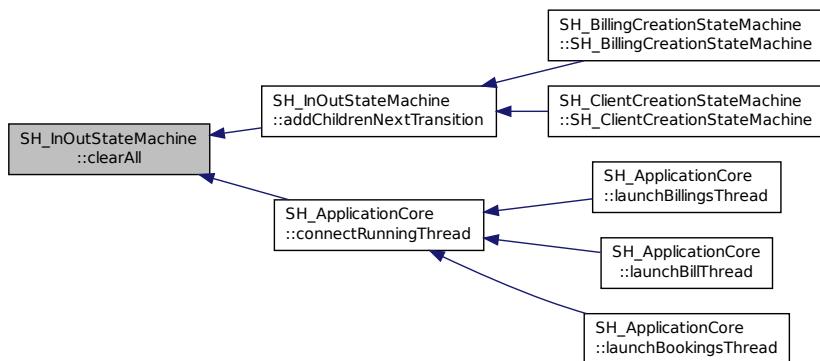
Voici le graphe des appels de cette fonction :



4.56.3.5 void SH_InOutStateMachine ::clearAll() [signal], [inherited]

Référencé par [SH_InOutStateMachine ::addChildsNextTransition\(\)](#), et [SH_ApplicationCore ::connectRunningThread\(\)](#).

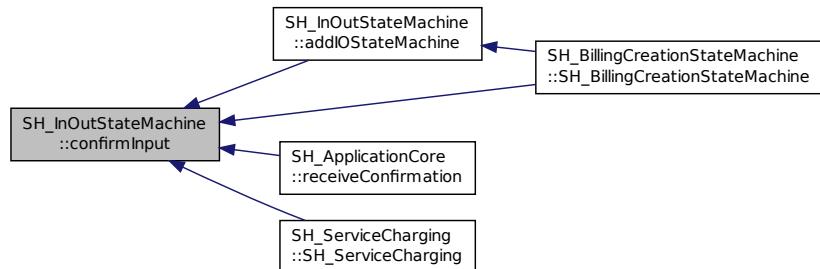
Voici le graphe des appels de cette fonction :



4.56.3.6 void SH_InOutStateMachine ::confirmInput() [signal], [inherited]

Référencé par [SH_InOutStateMachine ::addIOStateMachine\(\)](#), [SH_ApplicationCore ::receiveConfirmation\(\)](#), [SH_BillingCreationStateMachine ::SH_BillingCreationStateMachine\(\)](#), et [SH_ServiceCharging\(\)](#).

Voici le graphe des appelants de cette fonction :



4.56.3.7 int Sh_LoopingInOutStateMachine ::current() const [inherited]

Définition à la ligne 23 du fichier [SH_LoopingIOStateMachine.cpp](#).

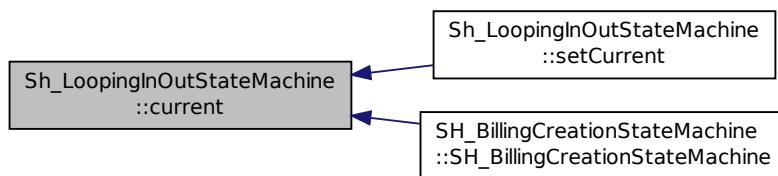
Références [Sh_LoopingInOutStateMachine ::m_current](#).

Référencé par [Sh_LoopingInOutStateMachine ::setCurrent\(\)](#), et [SH_BillingCreationStateMachine ::SH_BillingCreationStateMachine\(\)](#).

```

00024 {
00025     return m_current;
00026 }
```

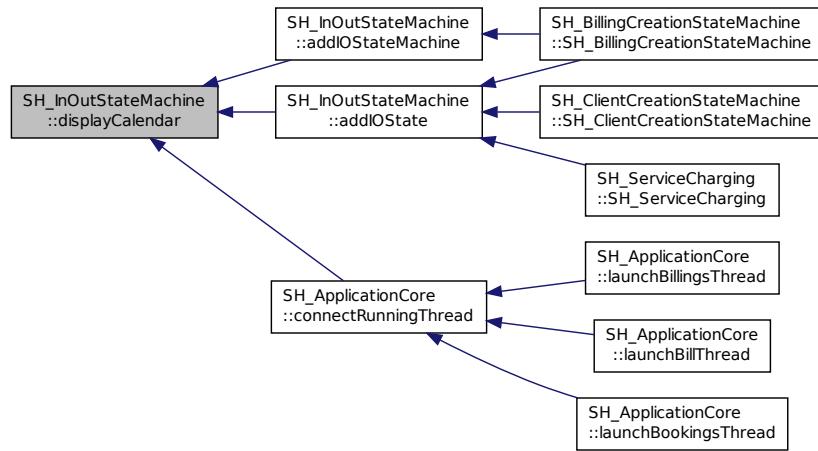
Voici le graphe des appelants de cette fonction :



4.56.3.8 void SH_InOutStateMachine ::displayCalendar() [signal], [inherited]

Référencé par [SH_InOutStateMachine ::addIOState\(\)](#), [SH_InOutStateMachine ::addIOSM\(\)](#), et [SH_ApplicationCore ::connectRunningThread\(\)](#).

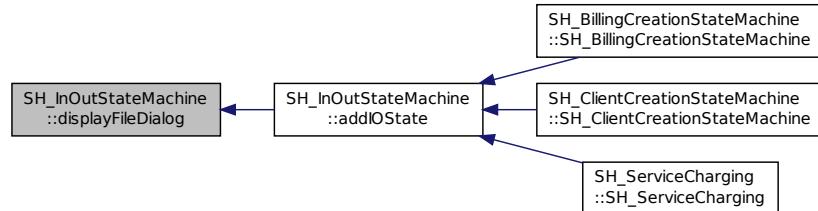
Voici le graphe des appelants de cette fonction :



4.56.3.9 void SH_InOutStateMachine ::displayFileDialog () [signal], [inherited]

Référencé par [SH_InOutStateMachine ::addIOState\(\)](#).

Voici le graphe des appelants de cette fonction :



4.56.3.10 QVariant SH_InOutStateMachine ::getContentValue (QString field) [inherited]

Définition à la ligne 65 du fichier [SH_IOStateMachine.cpp](#).

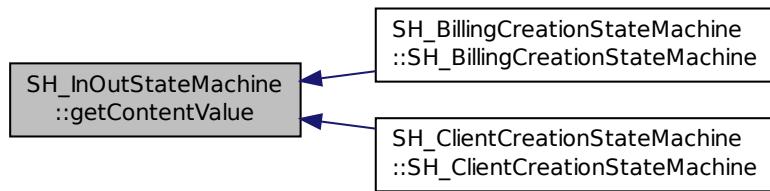
Références [SH_InOutStateMachine ::m_ioContent](#).

Référencé par [SH_BillingCreationStateMachine ::SH_BillingCreationStateMachine\(\)](#), et [SH_ClientCreationStateMachine ::SH_ClientCreationStateMachine\(\)](#).

```

00066 {
00067     return m_ioContent.value(field);
00068 }
  
```

Voici le graphe des appelants de cette fonction :



4.56.3.11 QHistoryState * SH_InOutStateMachine ::historyValue (QString field) [inherited]

Définition à la ligne 238 du fichier [SH_IOStateMachine.cpp](#).

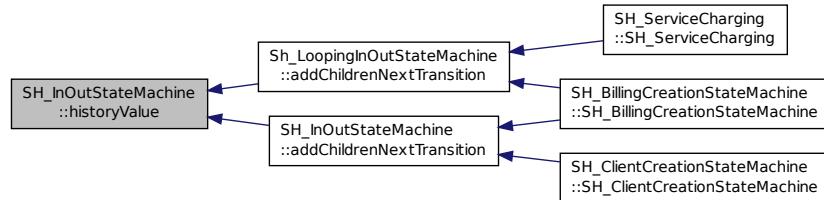
Références [SH_InOutStateMachine ::m_ioStatesHistory](#).

Référencé par [Sh_LoopingInOutStateMachine ::addChildrenNextTransition\(\)](#), et [SH_InOutStateMachine ::addChildrenNextTransition\(\)](#).

```

00239 {
00240     return m_ioStatesHistory.value(field);
00241 }
  
```

Voici le graphe des appelants de cette fonction :



4.56.3.12 QVariantMap SH_InOutStateMachine ::ioContent () const [inherited]

Définition à la ligne 43 du fichier [SH_IOStateMachine.cpp](#).

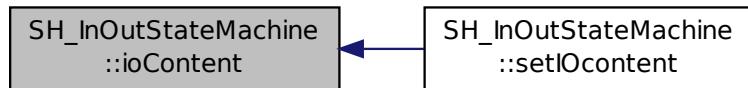
Références [SH_InOutStateMachine ::m_ioContent](#).

Référencé par [SH_InOutStateMachine ::setIOcontent\(\)](#).

```

00044 {
00045     return m_ioContent;
00046 }
  
```

Voici le graphe des appelants de cette fonction :



4.56.3.13 `QMap<QString, QHistoryState *> SH_InOutStateMachine ::ioStatesHistory() const [protected], [inherited]`

Définition à la ligne 202 du fichier [SH_IOStateMachine.cpp](#).

Références [SH_InOutStateMachine ::m_ioStatesHistory](#).

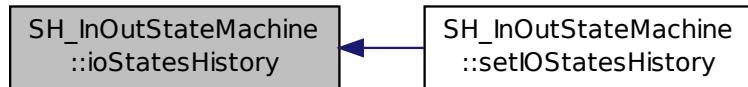
Référencé par [SH_InOutStateMachine ::setIOStatesHistory\(\)](#).

```

00203 {
00204     return m_ioStatesHistory;
00205 }

```

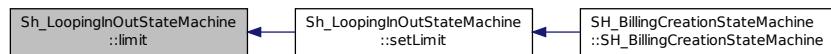
Voici le graphe des appelants de cette fonction :



4.56.3.14 `int Sh_LoopingInOutStateMachine ::limit() const [inherited]`

Référencé par [Sh_LoopingInOutStateMachine ::setLimit\(\)](#).

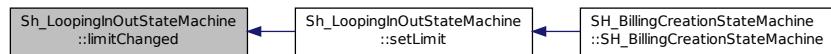
Voici le graphe des appelants de cette fonction :



4.56.3.15 `void Sh_LoopingInOutStateMachine ::limitChanged() [signal], [inherited]`

Référencé par [Sh_LoopingInOutStateMachine ::setLimit\(\)](#).

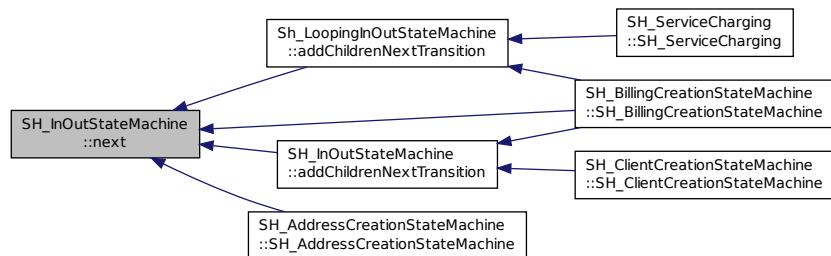
Voici le graphe des appelants de cette fonction :



4.56.3.16 void SH_InOutStateMachine ::next() [signal], [inherited]

Référencé par [Sh_LoopingInOutStateMachine ::addChildsNextTransition\(\)](#), [SH_InOutStateMachine ::addChildsNextTransition\(\)](#), [SH_AddressCreationStateMachine ::SH_AddressCreationStateMachine\(\)](#), et [SH_BillingCreationStateMachine ::SH_BillingCreationStateMachine\(\)](#).

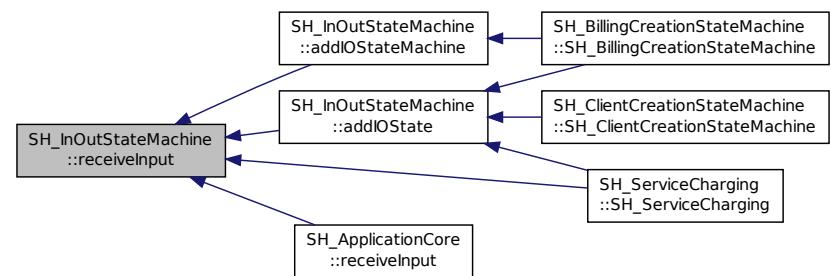
Voici le graphe des appelants de cette fonction :



4.56.3.17 void SH_InOutStateMachine ::receiveInput(QString input) [signal], [inherited]

Référencé par [SH_InOutStateMachine ::addIOState\(\)](#), [SH_InOutStateMachine ::addIOSMachine\(\)](#), [SH_ApplicationCore ::receiveInput\(\)](#), et [SH_ServiceCharging\(\)](#).

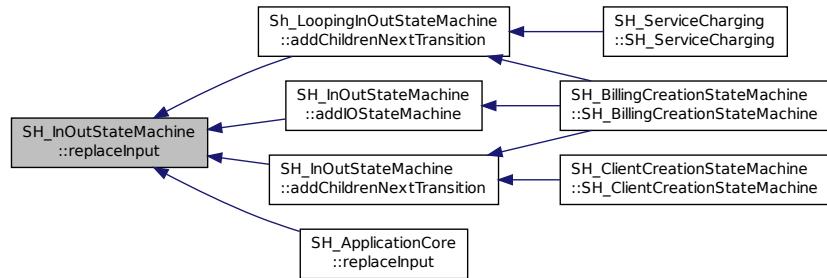
Voici le graphe des appelants de cette fonction :



4.56.3.18 void SH_InOutStateMachine ::replaceInput(QString field) [signal], [inherited]

Référencé par [Sh_LoopingInOutStateMachine ::addChildsNextTransition\(\)](#), [SH_InOutStateMachine ::addChildsNextTransition\(\)](#), [SH_InOutStateMachine ::addIOSMachine\(\)](#), et [SH_ApplicationCore ::replaceInput\(\)](#).

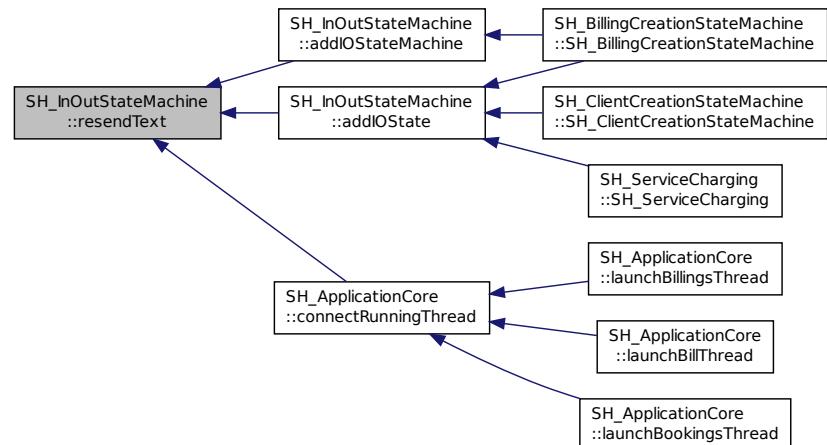
Voici le graphe des appelants de cette fonction :



4.56.3.19 void [SH_InOutStateMachine](#) ::resendText ([QString text](#), [bool editable = false](#)) [signal], [inherited]

Référencé par [SH_InOutStateMachine](#) ::[addIOState\(\)](#), [SH_InOutStateMachine](#) ::[addIOStateMachine\(\)](#), et [SH_ApplicationCore](#) ::[connectRunningThread\(\)](#).

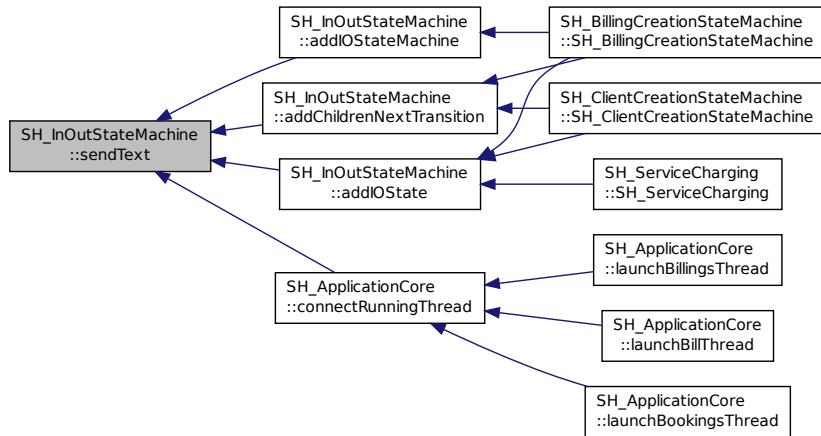
Voici le graphe des appelants de cette fonction :



4.56.3.20 void [SH_InOutStateMachine](#) ::sendText ([QString text](#), [bool editable = false](#)) [signal], [inherited]

Référencé par [SH_InOutStateMachine](#) ::[addChildrenNextTransition\(\)](#), [SH_InOutStateMachine](#) ::[addIOState\(\)](#), [SH_InOutStateMachine](#) ::[addIOStateMachine\(\)](#), et [SH_ApplicationCore](#) ::[connectRunningThread\(\)](#).

Voici le graphe des appelants de cette fonction :



4.56.3.21 void SH_InOutStateMachine ::setContentValue (QVariant content, QString field) [slot], [inherited]

Définition à la ligne 99 du fichier [SH_IStateMachine.cpp](#).

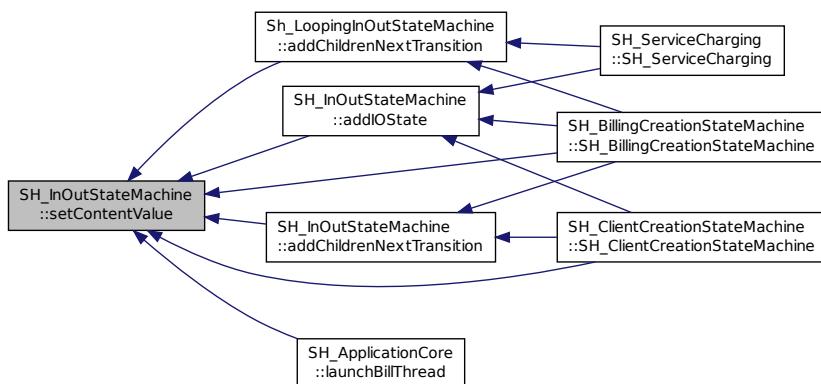
Références [SH_InOutStateMachine ::m_ioContent](#).

Référencé par [Sh_LoopingInOutStateMachine ::addChildrenNextTransition\(\)](#), [SH_InOutStateMachine ::addChildrenNextTransition\(\)](#), [SH_InOutStateMachine ::addIOState\(\)](#), [SH_ApplicationCore ::launchBillThread\(\)](#), [SH_BillingCreationStateMachine ::SH_BillingCreationStateMachine\(\)](#), et [SH_ClientCreationStateMachine ::SH_ClientCreationStateMachine\(\)](#).

```

00100 {
00101     m_ioContent.insert(field, content);
00102 }
  
```

Voici le graphe des appelants de cette fonction :



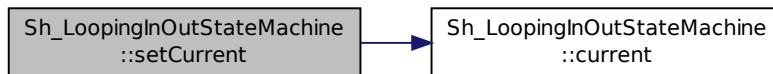
4.56.3.22 void Sh_LoopingInOutStateMachine ::setCurrent(int current) [inherited]

Définition à la ligne 34 du fichier [SH_LoopingIOStateMachine.cpp](#).

Références [Sh_LoopingInOutStateMachine ::current\(\)](#), et [Sh_LoopingInOutStateMachine ::m_current](#).

```
00035 {  
00036     m_current = current;  
00037 }
```

Voici le graphe d'appel pour cette fonction :



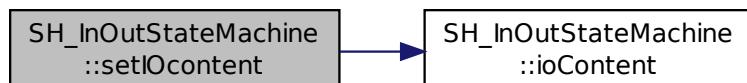
4.56.3.23 void SH_InOutStateMachine ::setIOcontent(const QVariantMap & ioContent) [inherited]

Définition à la ligne 54 du fichier [SH_IOStateMachine.cpp](#).

Références [SH_InOutStateMachine ::ioContent\(\)](#), et [SH_InOutStateMachine ::m_ioContent](#).

```
00055 {  
00056     m_ioContent = ioContent;  
00057 }
```

Voici le graphe d'appel pour cette fonction :



4.56.3.24 void SH_InOutStateMachine ::setIOStateHistory(QHistoryState * state, QString field) [inherited]

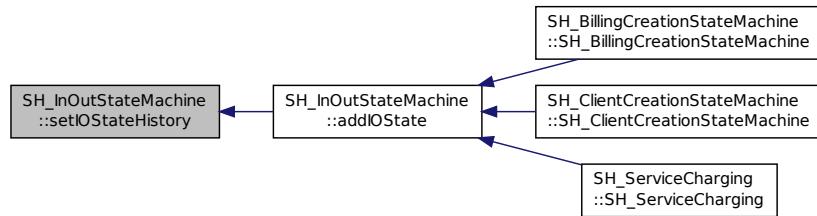
Définition à la ligne 226 du fichier [SH_IOStateMachine.cpp](#).

Références [SH_InOutStateMachine ::m_ioStatesHistory](#).

Référencé par [SH_InOutStateMachine ::addIOState\(\)](#).

```
00227 {  
00228     m_ioStatesHistory.insert(field, state); /*remplacement si plusieurs fois*/  
00229 }
```

Voici le graphe des appelants de cette fonction :



4.56.3.25 void Sh_LoopingInOutStateMachine ::setLimit (int limit) [inherited]

Définition à la ligne 61 du fichier [SH_LoopingIOMachine.cpp](#).

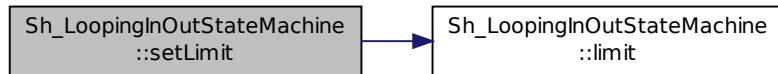
Références [Sh_LoopingInOutStateMachine ::limit\(\)](#), [Sh_LoopingInOutStateMachine ::limitChanged\(\)](#), et [Sh_LoopingInOutStateMachine ::m_limit](#).

Référencé par [SH_BillingCreationStateMachine ::SH_BillingCreationStateMachine\(\)](#).

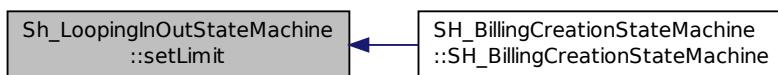
```

00062 {
00063     m_limit = limit;
00064     emit limitChanged();
00065 }
```

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :



4.56.3.26 void Sh_LoopingInOutStateMachine ::setPersistentContentValue (QVariant value, QString field) [inherited]

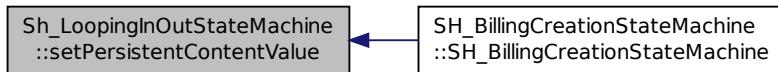
Définition à la ligne 39 du fichier [SH_LoopingIOMachine.cpp](#).

Références [Sh_LoopingInOutStateMachine ::m_persistentContent](#).

Référencé par [SH_BillingCreationStateMachine ::SH_BillingCreationStateMachine\(\)](#).

```
00040 {
00041     m_persistentContent.insert(field, value);
00042 }
```

Voici le graphe des appelants de cette fonction :



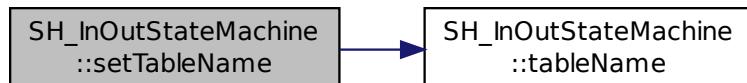
4.56.3.27 void SH_InOutStateMachine ::setTableName (const QString & tableName) [inherited]

Définition à la ligne 87 du fichier [SH_IOStateMachine.cpp](#).

Références [SH_InOutStateMachine ::m_tableName](#), et [SH_InOutStateMachine ::tableName\(\)](#).

```
00088 {
00089     m_tableName = tableName;
00090 }
```

Voici le graphe d'appel pour cette fonction :



4.56.3.28 void Sh_LoopingInOutStateMachine ::stopLooping() [inherited]

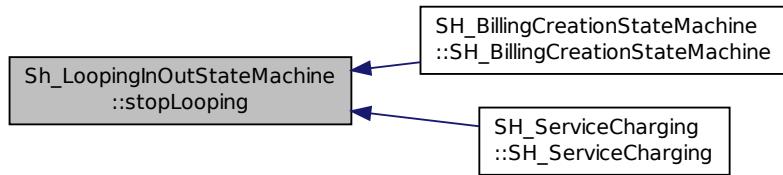
Définition à la ligne 72 du fichier [SH_LoopingIOStateMachine.cpp](#).

Références [Sh_LoopingInOutStateMachine ::m_current](#), et [Sh_LoopingInOutStateMachine ::m_limit](#).

Référencé par [SH_BillingCreationStateMachine ::SH_BillingCreationStateMachine\(\)](#), et [SH_ServiceCharging\(\)](#).

```
00072 {
00073     if(m_limit == 0) {
00074         m_limit = m_current + 1;
00075     } else {
00076         m_current = m_limit - 1;
00077     }
00078 }
```

Voici le graphe des appelants de cette fonction :



4.56.3.29 QString SH_InOutStateMachine ::tableName () const [inherited]

Définition à la ligne 76 du fichier [SH_IOStateMachine.cpp](#).

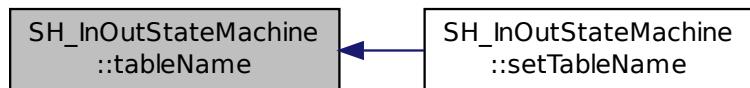
Références [SH_InOutStateMachine ::m_tableName](#).

Référencé par [SH_InOutStateMachine ::setTableName\(\)](#).

```

00077 {
00078     return m_tableName;
00079 }
```

Voici le graphe des appelants de cette fonction :



4.56.3.30 QString SH_InOutStateMachine ::toString () [virtual], [inherited]

Réimplémentée à partir de [SH_NamedObject](#).

Définition à la ligne 26 du fichier [SH_IOStateMachine.cpp](#).

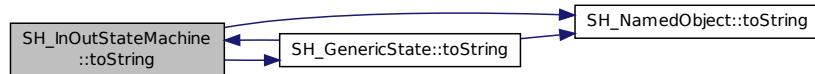
Références [SH_NamedObject ::toString\(\)](#), et [SH_GenericState ::toString\(\)](#).

Référencé par [Sh_LoopingInOutStateMachine ::addChildsNextTransition\(\)](#), [SH_InOutStateMachine ::addChildsNextTransition\(\)](#), [SH_ApplicationCore ::launchBillingsThread\(\)](#), [SH_BillingCreationStateMachine ::SH_BillingCreationStateMachine\(\)](#), et [SH_GenericState ::toString\(\)](#).

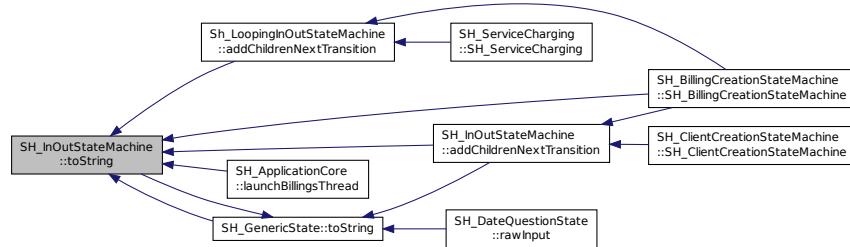
```

00027 {
00028     QObject* parent = this->parent();
00029     SH_GenericState* par = qobject_cast<SH_GenericState *>(parent);
00030     if(par) {
00031         return SH_NamedObject::toString() + " [descending from "+par->
00032             toString()+"] ";
00033     } else {
00034         return SH_NamedObject::toString();
00035     }
  
```

Voici le graphe d'appel pour cette fonction :



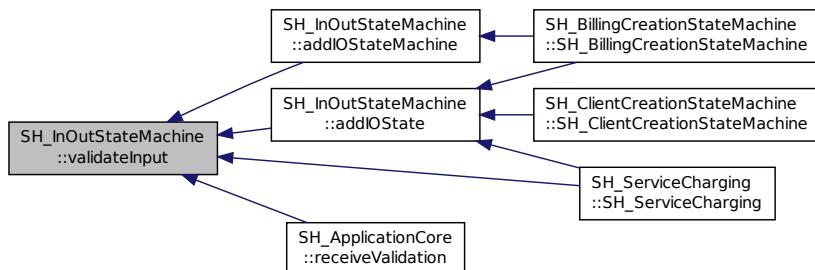
Voici le graphe des appels de cette fonction :



4.56.3.31 void SH_InOutStateMachine ::validateInput() [signal], [inherited]

Référencé par [SH_InOutStateMachine ::addIOState\(\)](#), [SH_InOutStateMachine ::addIOSMachine\(\)](#), [SH_ApplicationCore ::receiveValidation\(\)](#), et [SH_ServiceCharging\(\)](#).

Voici le graphe des appels de cette fonction :



4.56.4 Documentation des données membres

4.56.4.1 QVariantMap SH_InOutStateMachine ::m_ioContent [protected], [inherited]

m_ioContent

Définition à la ligne 209 du fichier [SH_IOSMachine.h](#).

Référencé par [Sh_LoopingInOutStateMachine ::addChildrenNextTransition\(\)](#), [SH_InOutStateMachine ::addChildrenNextTransition\(\)](#), [SH_InOutStateMachine ::getContentValue\(\)](#), [SH_InOutStateMachine ::ioContent\(\)](#), [SH_InOutStateMachine ::setContentValue\(\)](#), [SH_InOutStateMachine ::setIOcontent\(\)](#), et [SH_BillingCreationStateMachine ::SH_BillingCreationStateMachine\(\)](#).

4.56.4.2 `QMap<QString, QHistoryState*> SH_InOutStateMachine ::m_ioStatesHistory` [protected], [inherited]

`m_ioStatesHistory`

Définition à la ligne 217 du fichier [SH_IOStateMachine.h](#).

Référencé par [SH_InOutStateMachine](#) : `:historyValue()`, [SH_InOutStateMachine](#) : `:ioStatesHistory()`, [SH_InOutStateMachine](#) : `:setIOStateHistory()`, et [SH_InOutStateMachine](#) : `:setIOStatesHistory()`.

4.56.4.3 `qreal SH_ServiceCharging ::m_priceMin` [private]

`m_priceMin`

Définition à la ligne 27 du fichier [SH_ServiceCharging.h](#).

Référencé par [SH_ServiceCharging\(\)](#).

4.56.4.4 `QString SH_InOutStateMachine ::m_tableName` [protected], [inherited]

`m_tableName`

Définition à la ligne 213 du fichier [SH_IOStateMachine.h](#).

Référencé par [Sh_LoopingInOutStateMachine](#) : `:addChildsNextTransition()`, [SH_InOutStateMachine](#) : `:addChildsNextTransition()`, [SH_InOutStateMachine](#) : `:setTableName()`, [SH_BillingCreationStateMachine](#) : `:SH_BillingCreationStateMachine()`, et [SH_InOutStateMachine](#) : `:tableName()`.

4.56.4.5 `qreal SH_ServiceCharging ::m_vat` [private]

`m_vat`

Définition à la ligne 31 du fichier [SH_ServiceCharging.h](#).

Référencé par [SH_ServiceCharging\(\)](#).

4.56.5 Documentation des propriétés

4.56.5.1 `int Sh_LoopingInOutStateMachine ::limit` [read], [write], [inherited]

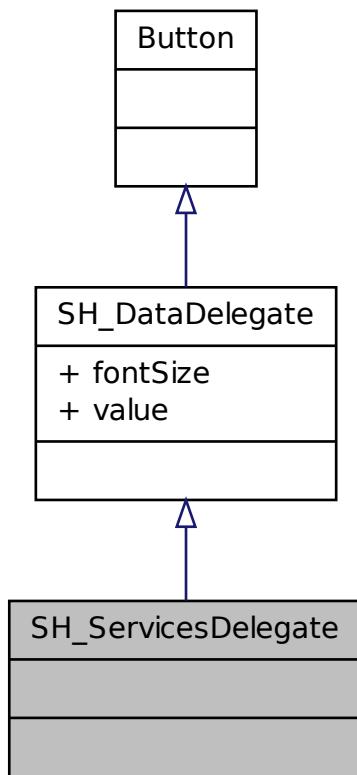
Définition à la ligne 13 du fichier [SH_LoopingIOStateMachine.h](#).

La documentation de cette classe a été générée à partir des fichiers suivants :

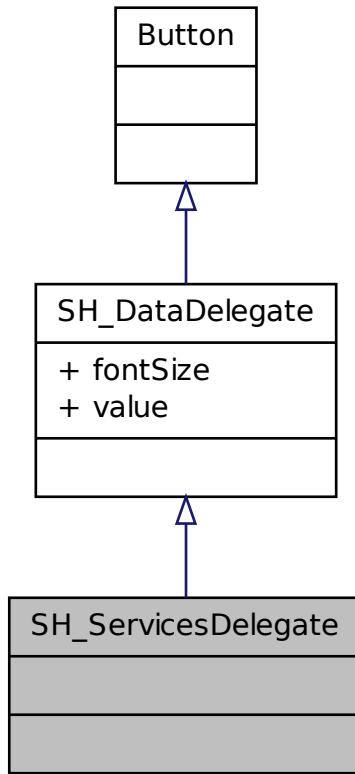
- logic/[SH_ServiceCharging.h](#)
- logic/[SH_ServiceCharging.cpp](#)

4.57 Référence de la classe SH_ServicesDelegate

Graphe d'héritage de SH_ServicesDelegate :



Graphe de collaboration de SH_ServicesDelegate :



Propriétés

- int `fontSize`
- string `value`

4.57.1 Description détaillée

Définition à la ligne 4 du fichier [SH_ServicesDelegate.qml](#).

4.57.2 Documentation des propriétés

4.57.2.1 int SH_DataDelegate ::fontSize [inherited]

Définition à la ligne 9 du fichier [SH_DataDelegate.qml](#).

4.57.2.2 string SH_DataDelegate ::value [inherited]

Définition à la ligne 7 du fichier [SH_DataDelegate.qml](#).

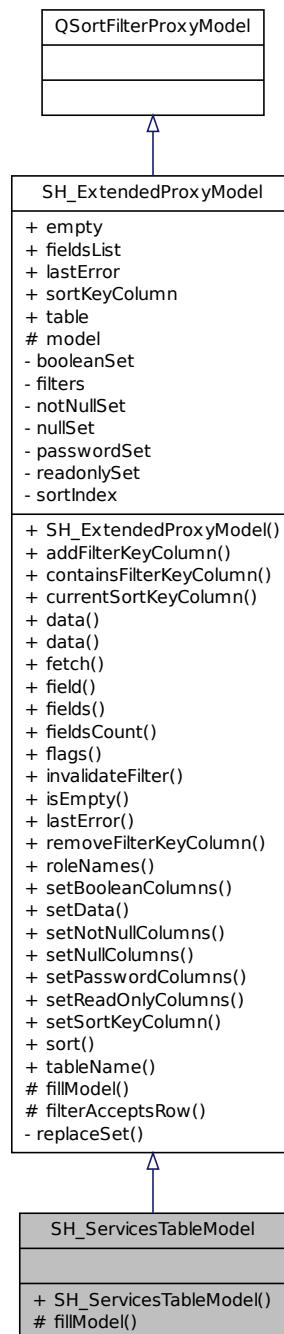
La documentation de cette classe a été générée à partir du fichier suivant :

- [views/qml/SH_ServicesDelegate.qml](#)

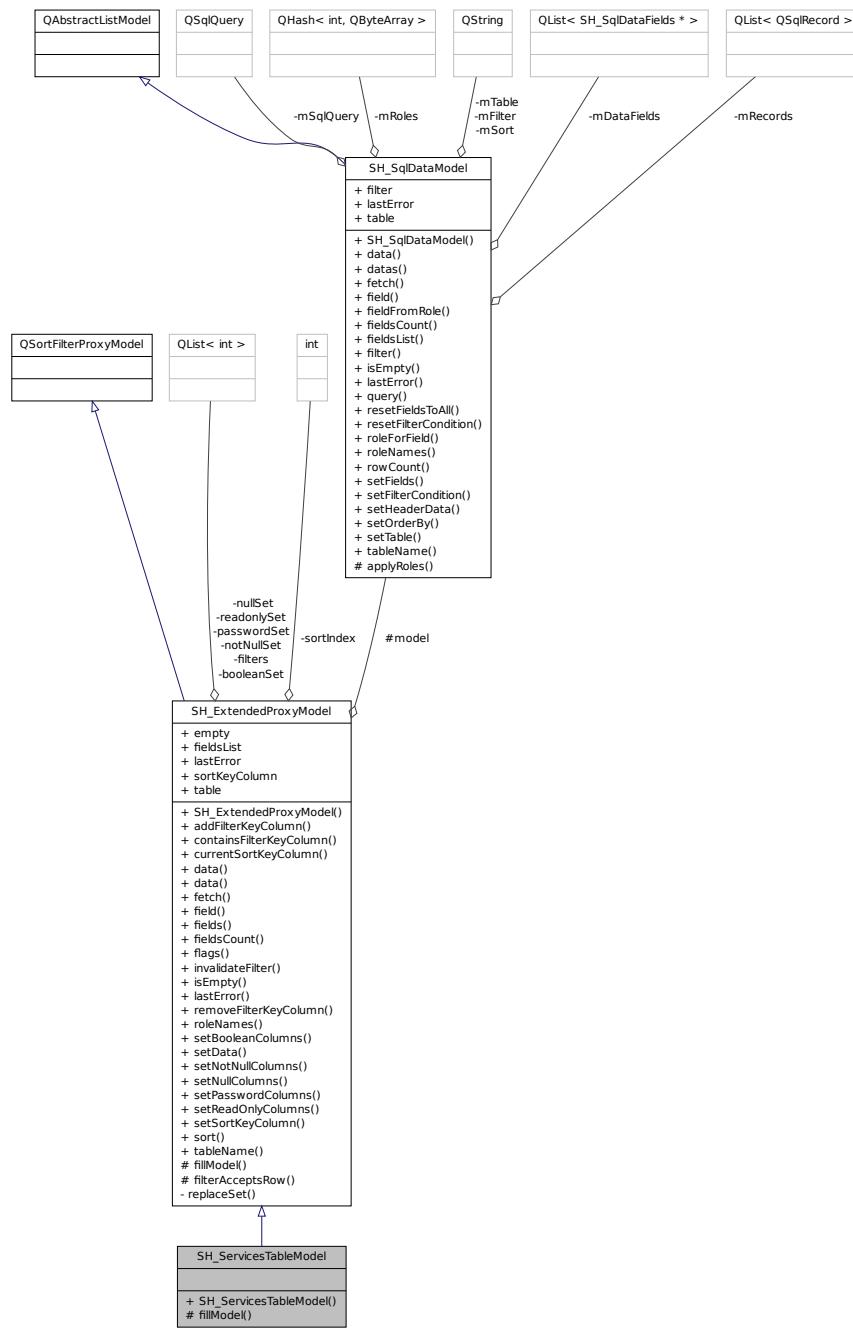
4.58 Référence de la classe SH_ServicesTableModel

```
#include <SH_ServicesTableModel.h>
```

Graphe d'héritage de SH_ServicesTableModel :



Graphe de collaboration de SH_ServicesTableModel :



Signaux

- void `sortChanged ()`

Fonctions membres publiques

- `SH_ServicesTableModel (QObject *parent=0)`
- `Q_INVOKABLE void addFilterKeyColumn (int column)`
- `Q_INVOKABLE bool containsFilterKeyColumn (int column)`
- `const int currentSortKeyColumn () const`

- Q_INVOKABLE QVariant **data** (int row, int column) const
- Q_INVOKABLE QVariant **data** (const QModelIndex &index, int role=Qt::DisplayRole) const
- Q_INVOKABLE bool **fetch** (QString **tableName**="", QString **filter**="", QString **sort**="", QStringList **fields**=QStringList())
- Q_INVOKABLE SH_SqlDataFields * **field** (int i) const
- const QString **fields** () const
- Q_INVOKABLE int **fieldsCount** () const
- Q_INVOKABLE Qt::ItemFlags **flags** (const QModelIndex &index) const
- void **invalidateFilter** ()
- const bool **isEmpty** () const
- const QString **lastError** () const
- Q_INVOKABLE void **removeFilterKeyColumn** (int column)
- virtual Q_INVOKABLE QHash<int, QByteArray > **roleNames** () const
- void **setBooleanColumns** (QList<int> **boolCols**)
- Q_INVOKABLE bool **setData** (const QModelIndex &index, const QVariant &value, int role=Qt::EditRole)
- void **setNotNullColumns** (QList<int> **notNullCols**)
- void **setNullColumns** (QList<int> **nullCols**)
- void **setPasswordColumns** (QList<int> **passwordCols**)
- void **setReadOnlyColumns** (QList<int> **readonlyCols**)
- void **setSortKeyColumn** (int column)
- Q_INVOKABLE void **sort** (int column, Qt::SortOrder newOrder=Qt::AscendingOrder)
- const QString **tableName** () const

Fonctions membres protégées

- void **fillModel** ()
- bool **filterAcceptsRow** (int source_row, const QModelIndex &source_parent) const

Attributs protégés

- SH_SqlDataModel * **model**

Propriétés

- bool **empty**
- QString **fieldsList**
- QString **lastError**
- int **sortKeyColumn**
- QString **table**

4.58.1 Description détaillée

Définition à la ligne 13 du fichier [SH_ServicesTableModel.h](#).

4.58.2 Documentation des constructeurs et destructeur

4.58.2.1 SH_ServicesTableModel : :SH_ServicesTableModel (QObject * parent = 0)

Paramètres

<i>parent</i>

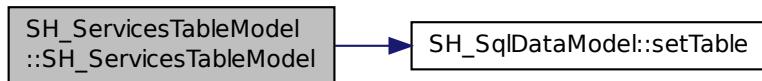
Définition à la ligne 9 du fichier [SH_ServicesTableModel.cpp](#).

Références [SH_ExtendedProxyModel](#) : :[model](#), et [SH_SqlDataModel](#) : :[setTable\(\)](#).

```

00009
00010     SH_ExtendedProxyModel (parent)
00011 {
00012     SH_ExtendedProxyModel::model->setTable ("SERVICESINFOS");
00013 }
```

Voici le graphe d'appel pour cette fonction :



4.58.3 Documentation des fonctions membres

4.58.3.1 void SH_ExtendedProxyModel : :addFilterKeyColumn (int column) [inherited]

Définition à la ligne 259 du fichier [SH_ExtendedProxyModel.cpp](#).

Références [SH_ExtendedProxyModel : :filters](#).

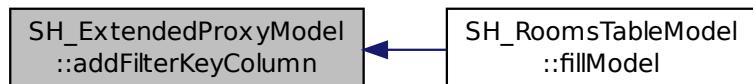
Référencé par [SH_RoomsTableModel : :fillModel\(\)](#).

```

00260 {
00261     this->filters.append\(column\);
00262 }

```

Voici le graphe des appels de cette fonction :



4.58.3.2 bool SH_ExtendedProxyModel : :containsFilterKeyColumn (int column) [inherited]

Définition à la ligne 225 du fichier [SH_ExtendedProxyModel.cpp](#).

Références [SH_ExtendedProxyModel : :filters](#).

```

00226 {
00227     return this->filters.contains\(column\);
00228 }

```

4.58.3.3 const int SH_ExtendedProxyModel : :currentSortKeyColumn () const [inline], [inherited]

Définition à la ligne 38 du fichier [SH_ExtendedSqlProxyModel.h](#).

Références [SH_ExtendedProxyModel : :sortIndex](#).

```
00038 {return this->sortIndex;}
```

4.58.3.4 QVariant SH_ExtendedProxyModel ::data (int row, int column) const [inherited]

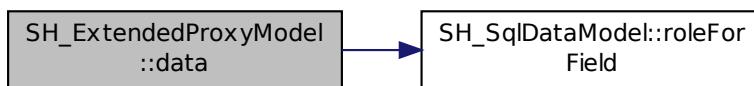
Définition à la ligne 269 du fichier [SH_ExtendedProxyModel.cpp](#).

Références [SH_ExtendedProxyModel ::model](#), et [SH_SqlDataModel ::roleForField\(\)](#).

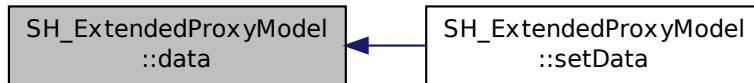
Référencé par [SH_ExtendedProxyModel ::setData\(\)](#).

```
00270 {
00271     QModelIndex modelIndex = this->index(row, 0);
00272     return this->data(modelIndex, this->model->roleForField(column));
00273 }
```

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appels de cette fonction :



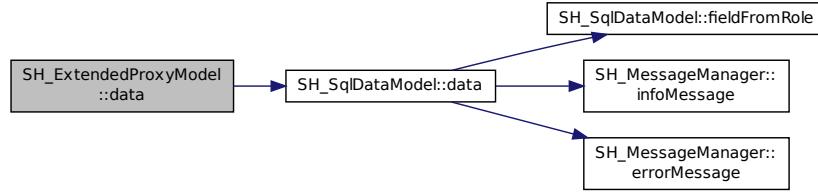
4.58.3.5 QVariant SH_ExtendedProxyModel ::data (const QModelIndex & index, int role = Qt ::DisplayRole) const [inherited]

Définition à la ligne 127 du fichier [SH_ExtendedProxyModel.cpp](#).

Références [SH_ExtendedProxyModel ::booleanSet](#), [SH_SqlDataModel ::data\(\)](#), [SH_ExtendedProxyModel ::filters](#), [SH_ExtendedProxyModel ::model](#), et [SH_ExtendedProxyModel ::passwordSet](#).

```
00128 {
00129     if (index.isValid())
00130     {
00131         if (this->booleanSet.contains(role))
00132         {
00133             return index.data(Qt::EditRole).toBool() ? QVariant(Qt::Checked) : QVariant(Qt::Unchecked);
00134         }
00135         else if (this->passwordSet.contains(role))
00136         {
00137             return QVariant("****");
00138         }
00139         else if (!this->filters.contains(role))
00140         {
00141             QModelIndex source_index = QSortFilterProxyModel::mapToSource(index);
00142             if (source_index.isValid())
00143                 return this->model->data(source_index, role);
00144             }
00145         }
00146     }
00147     return QVariant();
00148 }
```

Voici le graphe d'appel pour cette fonction :



4.58.3.6 bool SH_ExtendedProxyModel::fetch (QString tableName = "", QString filter = "", QString sort = "", QStringList fields = QStringList ()) [inherited]

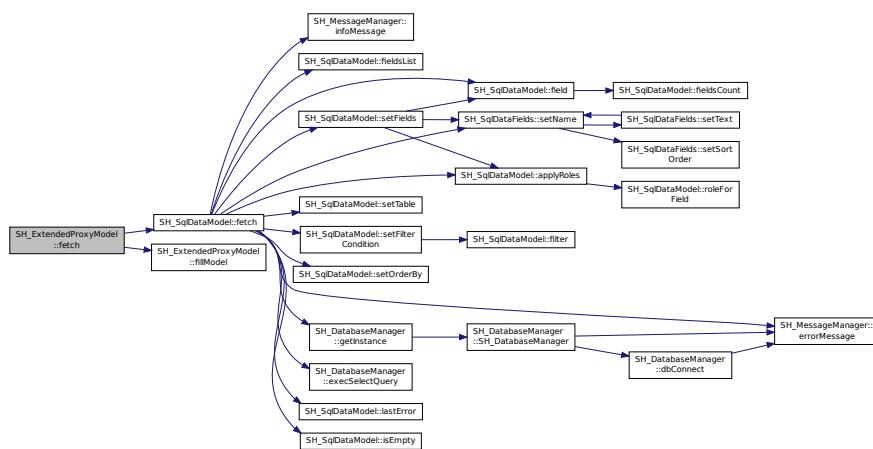
Définition à la ligne 280 du fichier [SH_ExtendedProxyModel.cpp](#).

Références [SH_SqlDataModel::fetch\(\)](#), [SH_ExtendedProxyModel::fillModel\(\)](#), et [SH_ExtendedProxyModel::model](#).

```

00281 {
00282     bool fetched = this->model->fetch(tableName, filter, sort,
00283         fields);
00283     if (fetched)
00284     {
00285         this->fillModel();
00286     }
00287     this->setSourceModel(this->model);
00288     return fetched;
00289 }
  
```

Voici le graphe d'appel pour cette fonction :



4.58.3.7 Q_INVOKABLE SH_SqlDataFields* SH_ExtendedProxyModel::field (int i) const [inline], [inherited]

Définition à la ligne 82 du fichier [SH_ExtendedSqlProxyModel.h](#).

Références [SH_SqlDataModel::field\(\)](#), et [SH_ExtendedProxyModel::model](#).

```
00082 { return this->model->field(i); }
```

Voici le graphe d'appel pour cette fonction :



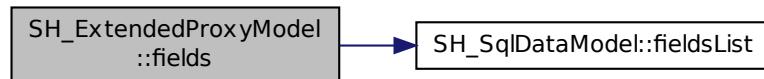
4.58.3.8 const QString SH_ExtendedProxyModel ::fields() const [inline], [inherited]

Définition à la ligne 52 du fichier [SH_ExtendedSqlProxyModel.h](#).

Références [SH_SqlDataModel ::fieldsList\(\)](#), et [SH_ExtendedProxyModel ::model](#).

```
00052 { if(this->model->fieldsList().isEmpty()){ return "*"; } else { return this->model->fieldsList().join(", "); } }
```

Voici le graphe d'appel pour cette fonction :



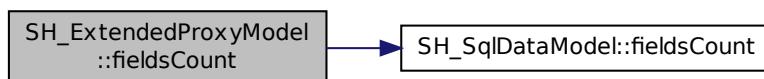
4.58.3.9 Q_INVOKABLE int SH_ExtendedProxyModel ::fieldsCount() const [inline], [inherited]

Définition à la ligne 89 du fichier [SH_ExtendedSqlProxyModel.h](#).

Références [SH_SqlDataModel ::fieldsCount\(\)](#), et [SH_ExtendedProxyModel ::model](#).

```
00089 { return this->model->fieldsCount(); }
```

Voici le graphe d'appel pour cette fonction :



4.58.3.10 void SH_ServicesTableModel ::fillModel() [protected], [virtual]

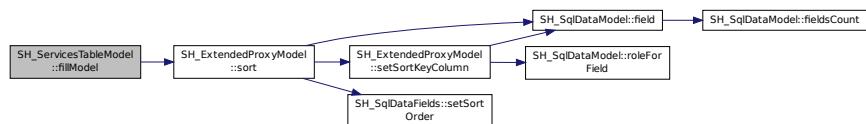
Implémente [SH_ExtendedProxyModel](#).

Définition à la ligne 21 du fichier [SH_ServicesTableModel.cpp](#).

Références [SH_ExtendedProxyModel ::sort\(\)](#).

```
00022 {
00023     SH_ExtendedProxyModel::sort(1,Qt::AscendingOrder);
00024 }
```

Voici le graphe d'appel pour cette fonction :



4.58.3.11 bool SH_ExtendedProxyModel ::filterAcceptsRow (int source_row, const QModelIndex & source_parent) const [protected], [inherited]

Définition à la ligne 92 du fichier [SH_ExtendedProxyModel.cpp](#).

Références [SH_ExtendedProxyModel ::notNullSet](#), et [SH_ExtendedProxyModel ::nullSet](#).

```
00093 {
00094     Q_UNUSED(source_parent);
00095
00096     if (!this->notNullSet.isEmpty())
00097     {
00098         QSqlQueryModel *m = static_cast<QSqlQueryModel *>(sourceModel());
00099         foreach(int column, this->notNullSet)
00100         {
00101             if (m->record(source_row).isNull(column))
00102             {
00103                 return false;
00104             }
00105         }
00106     }
00107
00108     if (!this->nullSet.isEmpty())
00109     {
00110         QSqlQueryModel *m = static_cast<QSqlQueryModel *>(sourceModel());
00111         foreach(int column, this->nullSet)
00112         {
00113             if (!m->record(source_row).isNull(column))
00114             {
00115                 return false;
00116             }
00117         }
00118     }
00119     return true;
00120 }
```

4.58.3.12 Qt ::ItemFlags SH_ExtendedProxyModel ::flags (const QModelIndex & index) const [inherited]

Définition à la ligne 179 du fichier [SH_ExtendedProxyModel.cpp](#).

Références [SH_ExtendedProxyModel ::booleanSet](#), et [SH_ExtendedProxyModel ::readonlySet](#).

```
00180 {
00181     if (!index.isValid())
00182     {
00183         return Qt::ItemIsEnabled;
```

```

00184     }
00185     if (!this->booleanSet.isEmpty())
00186     {
00187         return Qt::ItemIsUserCheckable | Qt::ItemIsSelectable | Qt::ItemIsEnabled;
00188     }
00189     else if (!this->readonlySet.isEmpty())
00190     {
00191         return Qt::ItemIsSelectable;
00192     }
00193     else
00194     {
00195         return QSortFilterProxyModel::flags(index);
00196     }
00197 }
00198 }
```

4.58.3.13 void SH_ExtendedProxyModel : :invalidateFilter () [inherited]

Définition à la ligne 205 du fichier [SH_ExtendedProxyModel.cpp](#).

Références [SH_ExtendedProxyModel : :filters](#).

```

00206 {
00207     this->filters.clear();
00208 }
```

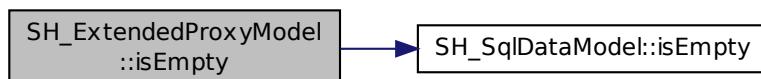
4.58.3.14 const bool SH_ExtendedProxyModel : :isEmpty () const [inline], [inherited]

Définition à la ligne 66 du fichier [SH_ExtendedSqlProxyModel.h](#).

Références [SH_SqlDataManager : :isEmpty\(\)](#), et [SH_ExtendedProxyModel : :model](#).

```
00066 { return this->model->isEmpty(); }
```

Voici le graphe d'appel pour cette fonction :



4.58.3.15 const QString SH_ExtendedProxyModel : :lastError () const [inline], [inherited]

Définition à la ligne 59 du fichier [SH_ExtendedSqlProxyModel.h](#).

Références [SH_SqlDataManager : :lastError](#), et [SH_ExtendedProxyModel : :model](#).

```
00059 { return this->model->lastError(); }
```

4.58.3.16 void SH_ExtendedProxyModel : :removeFilterKeyColumn (int column) [inherited]

Définition à la ligne 215 du fichier [SH_ExtendedProxyModel.cpp](#).

Références [SH_ExtendedProxyModel : :filters](#).

```

00216 {
00217     this->filters.removeAt(this->filters.indexOf(column));
00218 }
```

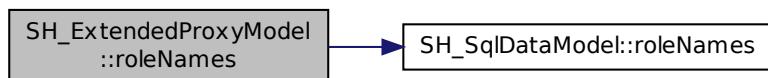
```
4.58.3.17 virtual Q_INVOKABLE QHash<int, QByteArray> SH_ExtendedProxyModel::roleNames() const [inline],  
[virtual], [inherited]
```

Définition à la ligne 165 du fichier [SH_ExtendedSqlProxyModel.h](#).

Références [SH_ExtendedProxyModel::model](#), et [SH_SqlDataModel::roleNames\(\)](#).

```
00165 { return this->model->roleNames(); }
```

Voici le graphe d'appel pour cette fonction :



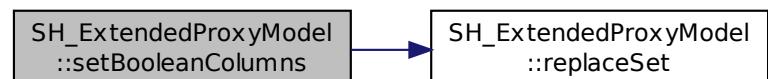
```
4.58.3.18 void SH_ExtendedProxyModel::setBooleanColumns( QList<int> &boolCols ) [inherited]
```

Définition à la ligne 41 du fichier [SH_ExtendedProxyModel.cpp](#).

Références [SH_ExtendedProxyModel::booleanSet](#), et [SH_ExtendedProxyModel::replaceSet\(\)](#).

```
00041 {  
00042     replaceSet(this->booleanSet, boolCols);  
00043 }
```

Voici le graphe d'appel pour cette fonction :



```
4.58.3.19 bool SH_ExtendedProxyModel::setData( const QModelIndex & index, const QVariant & value, int role = Qt::EditRole ) [inherited]
```

Définition à la ligne 156 du fichier [SH_ExtendedProxyModel.cpp](#).

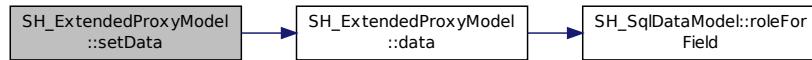
Références [SH_ExtendedProxyModel::booleanSet](#), et [SH_ExtendedProxyModel::data\(\)](#).

```
00157 {  
00158     if (!index.isValid())  
00159         return false;  
00160  
00161     if (this->booleanSet.contains(role))  
00162     {  
00163         QVariant data = (value.toInt() == Qt::Checked) ? QVariant(1) : QVariant(0);  
00164         return QSortFilterProxyModel::setData(index, data, role);  
00165     }  
00166     else  
00167     {
```

```

00168     return QSortFilterProxyModel::setData(index, value, role);
00169 }
00170
00171 }
```

Voici le graphe d'appel pour cette fonction :



4.58.3.20 void SH_ExtendedProxyModel ::setNotNullColumns (QList< int > notNullCols) [inherited]

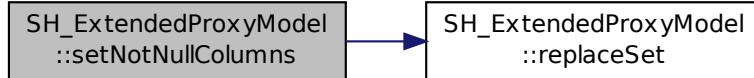
Définition à la ligne 80 du fichier [SH_ExtendedProxyModel.cpp](#).

Références [SH_ExtendedProxyModel ::notNullSet](#), et [SH_ExtendedProxyModel ::replaceSet\(\)](#).

```

00080
00081     if (sourceModel()->inherits("QSqlQueryModel")) {
00082         replaceSet(this->notNullSet, notNullCols);
00083     }
00084 }
```

Voici le graphe d'appel pour cette fonction :



4.58.3.21 void SH_ExtendedProxyModel ::setNullColumns (QList< int > nullCols) [inherited]

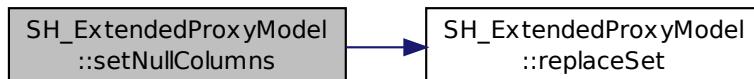
Définition à la ligne 68 du fichier [SH_ExtendedProxyModel.cpp](#).

Références [SH_ExtendedProxyModel ::nullSet](#), et [SH_ExtendedProxyModel ::replaceSet\(\)](#).

```

00068
00069     if (sourceModel()->inherits("QSqlQueryModel")) {
00070         replaceSet(this->nullSet, nullCols);
00071     }
00072 }
```

Voici le graphe d'appel pour cette fonction :



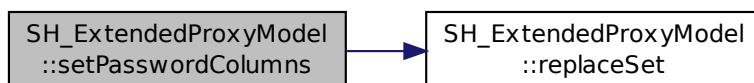
4.58.3.22 void SH_ExtendedProxyModel : :setPasswordColumns (QList< int > *passwordCols*) [inherited]

Définition à la ligne 59 du fichier [SH_ExtendedProxyModel.cpp](#).

Références [SH_ExtendedProxyModel : :passwordSet](#), et [SH_ExtendedProxyModel : :replaceSet\(\)](#).

```
00059     replaceSet(this->passwordSet, passwordCols);
00060 }
00061 }
```

Voici le graphe d'appel pour cette fonction :



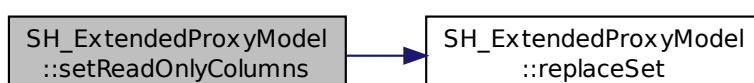
4.58.3.23 void SH_ExtendedProxyModel : :setReadOnlyColumns (QList< int > *readonlyCols*) [inherited]

Définition à la ligne 50 du fichier [SH_ExtendedProxyModel.cpp](#).

Références [SH_ExtendedProxyModel : :readonlySet](#), et [SH_ExtendedProxyModel : :replaceSet\(\)](#).

```
00050     replaceSet(this->readonlySet, readonlyCols);
00051 }
00052 }
```

Voici le graphe d'appel pour cette fonction :



4.58.3.24 void SH_ExtendedProxyModel : :setSortKeyColumn (int *column*) [inherited]

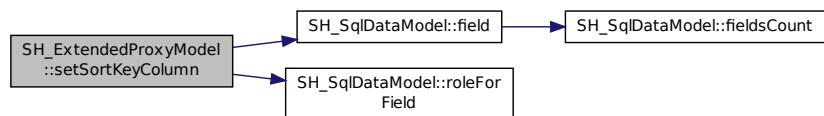
Définition à la ligne 246 du fichier [SH_ExtendedProxyModel.cpp](#).

Références [SH_SqlDataModel : :field\(\)](#), [SH_ExtendedProxyModel : :model](#), [SH_SqlDataModel : :roleForField\(\)](#), [SH_ExtendedProxyModel : :sortChanged\(\)](#), [SH_ExtendedProxyModel : :sortIndex](#), et [SH_SqlDataFields : :sortOrder](#).

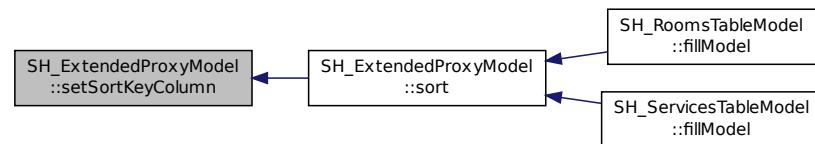
Référencé par [SH_ExtendedProxyModel : :sort\(\)](#).

```
00247 {
00248     this->sortIndex = column;
00249     QSortFilterProxyModel::setSortRole(this->model->roleForField(column));
00250     QSortFilterProxyModel::sort(0, this->model->field(column)->
00251         sortOrder());
00251     emit sortChanged();
00252 }
```

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appels de cette fonction :



4.58.3.25 void SH_ExtendedProxyModel ::sort (int column, Qt ::SortOrder newOrder = Qt ::AscendingOrder) [inherited]

Définition à la ligne 235 du fichier [SH_ExtendedProxyModel.cpp](#).

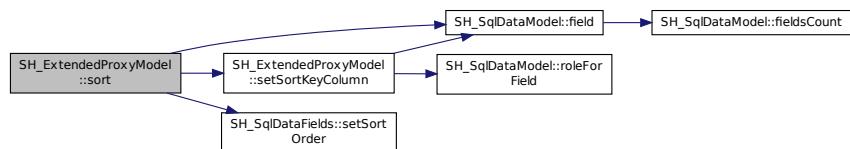
Références [SH_SqlDataModel ::field\(\)](#), [SH_ExtendedProxyModel ::model\(\)](#), [SH_ExtendedProxyModel ::setSortKeyColumn\(\)](#), et [SH_SqlDataFields ::setSortOrder\(\)](#).

Référencé par [SH_RoomsTableModel ::fillModel\(\)](#), et [fillModel\(\)](#).

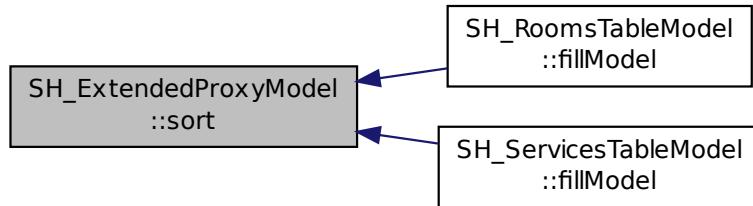
```

00236 {
00237     this->model->field(column)->setSortOrder(newOrder);
00238     SH_ExtendedProxyModel::setSortKeyColumn(column);
00239 }
  
```

Voici le graphe d'appel pour cette fonction :



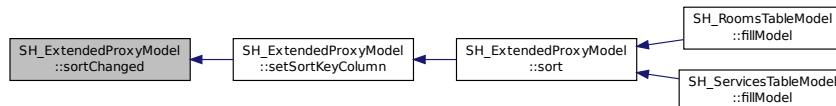
Voici le graphe des appelants de cette fonction :



4.58.3.26 void SH_ExtendedProxyModel ::sortChanged() [signal], [inherited]

Référencé par [SH_ExtendedProxyModel ::setSortKeyColumn\(\)](#).

Voici le graphe des appelants de cette fonction :



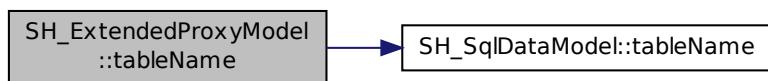
4.58.3.27 const QString SH_ExtendedProxyModel ::tableName() const [inline], [inherited]

Définition à la ligne 45 du fichier [SH_ExtendedSqlProxyModel.h](#).

Références [SH_ExtendedProxyModel ::model](#), et [SH_SqlDataModel ::tableName\(\)](#).

```
00045 { return this->model->tableName(); }
```

Voici le graphe d'appel pour cette fonction :



4.58.4 Documentation des données membres

4.58.4.1 SH_SqlDataModel* SH_ExtendedProxyModel ::model [protected], [inherited]

Définition à la ligne 241 du fichier [SH_ExtendedSqlProxyModel.h](#).

Référencé par [SH_ExtendedProxyModel](#) : `:data()`, [SH_ExtendedProxyModel](#) : `:fetch()`, [SH_ExtendedProxyModel](#) : `:field()`, [SH_ExtendedProxyModel](#) : `:fields()`, [SH_ExtendedProxyModel](#) : `:fieldsCount()`, [SH_BillingsTableModel](#) : `:fillModel()`, [SH_RoomsTableModel](#) : `:fillModel()`, [SH_BookingsTableModel](#) : `:fillModel()`, [SH_ExtendedProxyModel](#) : `:isEmpty()`, [SH_ExtendedProxyModel](#) : `:lastError()`, [SH_ExtendedProxyModel](#) : `:roleNames()`, [SH_ExtendedProxyModel](#) : `:setSortKeyColumn()`, [SH_BillingsTableModel](#) : `:SH_BillingsTableModel()`, [SH_BillsTableModel](#) : `:SH_BillsTableModel()`, [SH_BookingsTableModel](#) : `:SH_BookingsTableModel()`, [SH_ClientsTableModel](#) : `:SH_ClientsTableModel()`, [SH_ExtendedProxyModel](#) : `:SH_ExtendedProxyModel()`, [SH_GroupsTableModel](#) : `:SH_GroupsTableModel()`, [SH_RoomsTableModel](#) : `:SH_RoomsTableModel()`, [SH_ServicesTableModel](#) : `:SH_ServicesTableModel()`, [SH_ExtendedProxyModel](#) : `:sort()`, et [SH_ExtendedProxyModel](#) : `:tableName()`.

4.58.5 Documentation des propriétés

4.58.5.1 bool SH_ExtendedProxyModel :`:empty` [read], [inherited]

Définition à la ligne 21 du fichier [SH_ExtendedSqlProxyModel.h](#).

4.58.5.2 QString SH_ExtendedProxyModel :`:fieldsList` [read], [inherited]

Définition à la ligne 18 du fichier [SH_ExtendedSqlProxyModel.h](#).

4.58.5.3 QString SH_ExtendedProxyModel :`:lastError` [read], [inherited]

Définition à la ligne 19 du fichier [SH_ExtendedSqlProxyModel.h](#).

4.58.5.4 int SH_ExtendedProxyModel :`:sortKeyColumn` [read], [write], [inherited]

Définition à la ligne 20 du fichier [SH_ExtendedSqlProxyModel.h](#).

4.58.5.5 QString SH_ExtendedProxyModel :`:table` [read], [inherited]

Définition à la ligne 17 du fichier [SH_ExtendedSqlProxyModel.h](#).

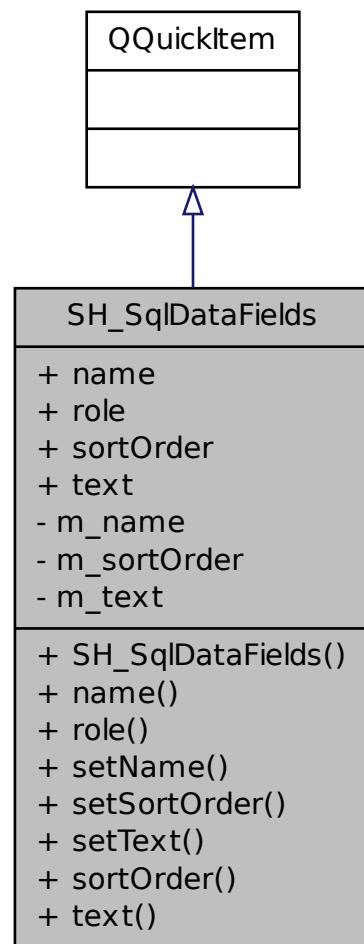
La documentation de cette classe a été générée à partir des fichiers suivants :

- [models/SH_ServicesTableModel.h](#)
- [models/SH_ServicesTableModel.cpp](#)

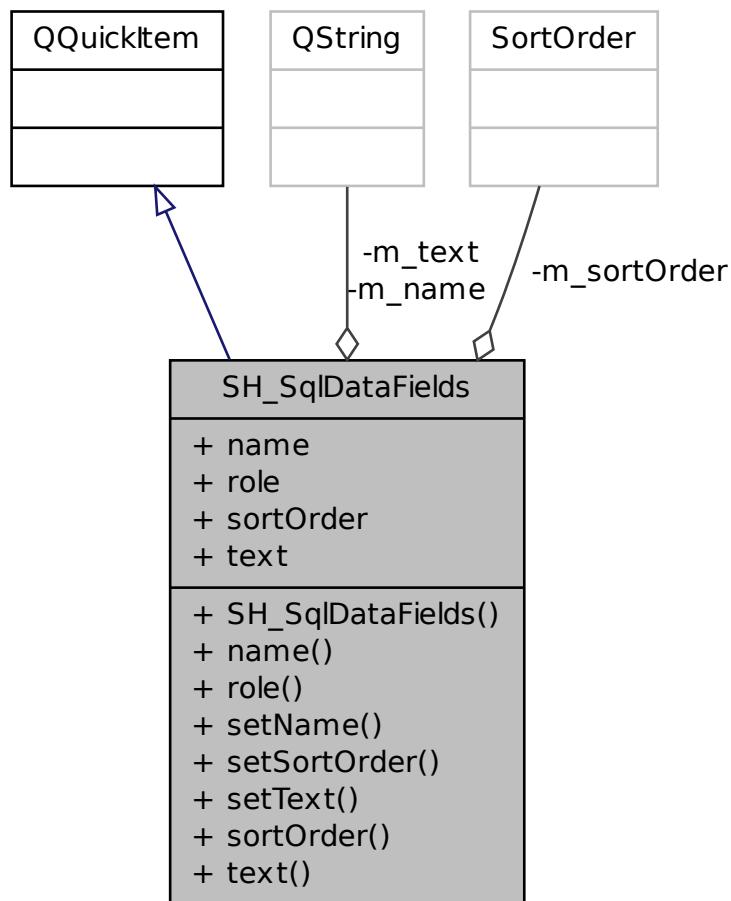
4.59 Référence de la classe SH_SqlDataFields

```
#include <SH_SqlDataField.h>
```

Graphe d'héritage de SH_SqlDataFields :



Graphe de collaboration de SH_SqlDataFields :



Signaux

- void `nameChanged ()`
- void `roleChanged ()`
- void `sortOrderChanged ()`
- void `textChanged ()`

Fonctions membres publiques

- `SH_SqlDataFields (QQuickItem *parent=0)`
- `QString name () const`
- `QByteArray role () const`
- `void setName (QString newName)`
- `void setSortOrder (Qt::SortOrder newSortOrder)`
- `void setText (QString newText)`
- `Qt::SortOrder sortOrder () const`
- `QString text () const`

Propriétés

- `QString name`

- QByteArray **role**
- Qt::SortOrder **sortOrder**
- QString **text**

Attributs privés

- QString **m_name**
m_name
- Qt::SortOrder **m_sortOrder**
m_sortOrder
- QString **m_text**
m_text

4.59.1 Description détaillée

Définition à la ligne 12 du fichier [SH_SqlDataField.h](#).

4.59.2 Documentation des constructeurs et destructeur

4.59.2.1 SH_SqlDataFields ::SH_SqlDataFields (QQuickItem * *parent* = 0) [explicit]

Paramètres

<i>parent</i>

Définition à la ligne 8 du fichier [SH_SqlDataField.cpp](#).

```
00008
00009     QQuickItem(parent)
00010 {
00011     this->setHeight(15);
00012 }
```

4.59.3 Documentation des fonctions membres

4.59.3.1 QString SH_SqlDataFields ::name () const [inline]

Définition à la ligne 42 du fichier [SH_SqlDataField.h](#).

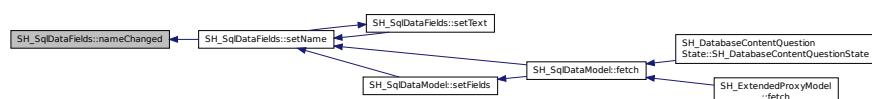
Références [m_name](#).

```
00042 { return m_name; }
```

4.59.3.2 void SH_SqlDataFields ::nameChanged () [signal]

Référencé par [setName\(\)](#).

Voici le graphe des appels de cette fonction :



4.59.3.3 QByteArray SH_SqlDataFields ::role() const [inline]

Définition à la ligne 49 du fichier [SH_SqlDataField.h](#).

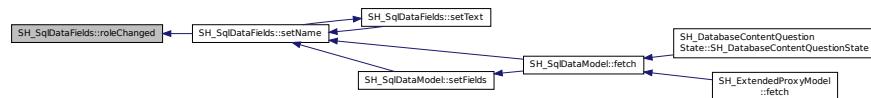
Références [m_name](#).

```
00049 { return QByteArray(m_name.toUpper().toStdString().c_str()); }
```

4.59.3.4 void SH_SqlDataFields ::roleChanged() [signal]

Référencé par [setName\(\)](#).

Voici le graphe des appels de cette fonction :



4.59.3.5 void SH_SqlDataFields ::setName(QString newName)

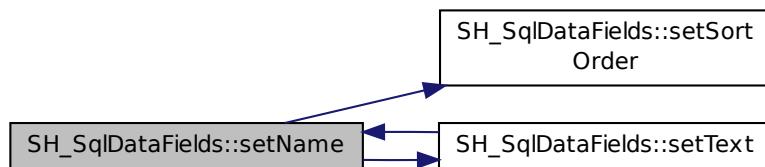
Définition à la ligne 35 du fichier [SH_SqlDataField.cpp](#).

Références [m_name](#), [m_text](#), [nameChanged\(\)](#), [roleChanged\(\)](#), [setSortOrder\(\)](#), et [setText\(\)](#).

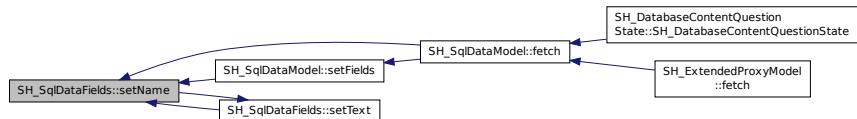
Référencé par [SH_SqlDataModel ::fetch\(\)](#), [SH_SqlDataModel ::setFields\(\)](#), et [setText\(\)](#).

```
00036 {
00037     m_name = newName;
00038     this->setSortOrder(Qt::AscendingOrder);
00039     if (m_text == "")
00040     {
00041         this->setText(m_name);
00042     }
00043     emit nameChanged();
00044     emit roleChanged();
00045 }
```

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :



4.59.3.6 void SH_SqlDataFields ::setSortOrder (Qt ::SortOrder newSortOrder)

Définition à la ligne 54 du fichier [SH_SqlDataField.cpp](#).

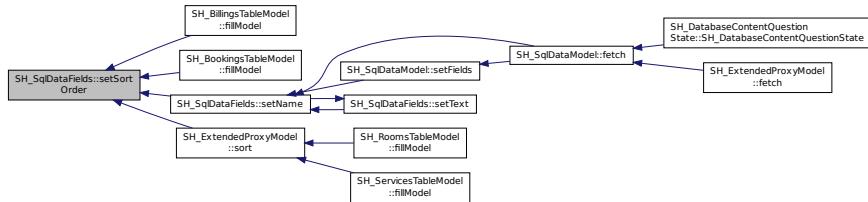
Références [m_sortOrder](#), et [sortOrderChanged\(\)](#).

Référencé par [SH_BillingsTableModel ::fillModel\(\)](#), [SH_BookingsTableModel ::fillModel\(\)](#), [setName\(\)](#), et [SH_ExtendedProxyModel ::sort\(\)](#).

```

00055 {
00056     m_sortOrder = newSortOrder;
00057     emit sortOrderChanged();
00058 }
  
```

Voici le graphe des appelants de cette fonction :



4.59.3.7 void SH_SqlDataFields ::setText (QString newText)

Définition à la ligne 19 du fichier [SH_SqlDataField.cpp](#).

Références [m_name](#), [m_text](#), [setName\(\)](#), et [textChanged\(\)](#).

Référencé par [setName\(\)](#).

```

00020 {
00021     m_text = newText;
00022     if (m_name == "")
00023     {
00024         this->setName(m_text.toUpper());
00025     }
00026     emit textChanged();
00027 }
  
```

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :



4.59.3.8 Qt::SortOrder SH_SqlDataFields::sortOrder() const [inline]

Définition à la ligne 56 du fichier [SH_SqlDataField.h](#).

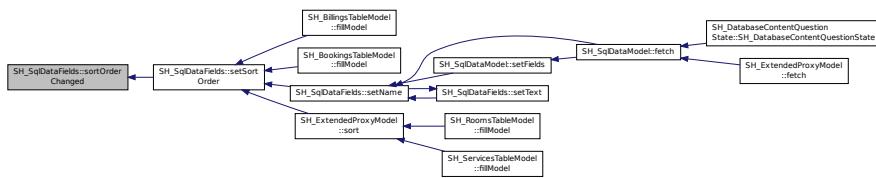
Références [m_sortOrder](#).

```
00056 { return m_sortOrder; }
```

4.59.3.9 void SH_SqlDataFields::sortOrderChanged() [signal]

Référencé par [setSortOrder\(\)](#).

Voici le graphe des appelants de cette fonction :



4.59.3.10 QString SH_SqlDataFields::text() const [inline]

Définition à la ligne 35 du fichier [SH_SqlDataField.h](#).

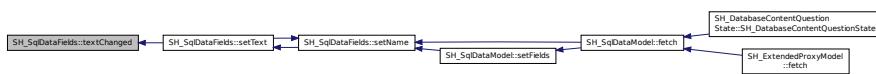
Références [m_text](#).

```
00035 { return m_text; }
```

4.59.3.11 void SH_SqlDataFields::textChanged() [signal]

Référencé par [setText\(\)](#).

Voici le graphe des appelants de cette fonction :



4.59.4 Documentation des données membres

4.59.4.1 `QString SH_SqlDataFields ::m_name [private]`

`m_name`

Définition à la ligne 88 du fichier [SH_SqlDataField.h](#).

Référencé par [name\(\)](#), [role\(\)](#), [setName\(\)](#), et [setText\(\)](#).

4.59.4.2 `Qt ::SortOrder SH_SqlDataFields ::m_sortOrder [private]`

`m_sortOrder`

Définition à la ligne 92 du fichier [SH_SqlDataField.h](#).

Référencé par [setSortOrder\(\)](#), et [sortOrder\(\)](#).

4.59.4.3 `QString SH_SqlDataFields ::m_text [private]`

`m_text`

Définition à la ligne 84 du fichier [SH_SqlDataField.h](#).

Référencé par [setName\(\)](#), [setText\(\)](#), et [text\(\)](#).

4.59.5 Documentation des propriétés

4.59.5.1 `QString SH_SqlDataFields ::name [read], [write]`

Définition à la ligne 15 du fichier [SH_SqlDataField.h](#).

Référencé par [SH_SqlDataModel ::fetch\(\)](#).

4.59.5.2 `QByteArray SH_SqlDataFields ::role [read]`

Définition à la ligne 17 du fichier [SH_SqlDataField.h](#).

4.59.5.3 `Qt ::SortOrder SH_SqlDataFields ::sortOrder [read], [write]`

Définition à la ligne 18 du fichier [SH_SqlDataField.h](#).

Référencé par [SH_ExtendedProxyModel ::setSortKeyColumn\(\)](#).

4.59.5.4 `QString SH_SqlDataFields ::text [read], [write]`

Définition à la ligne 16 du fichier [SH_SqlDataField.h](#).

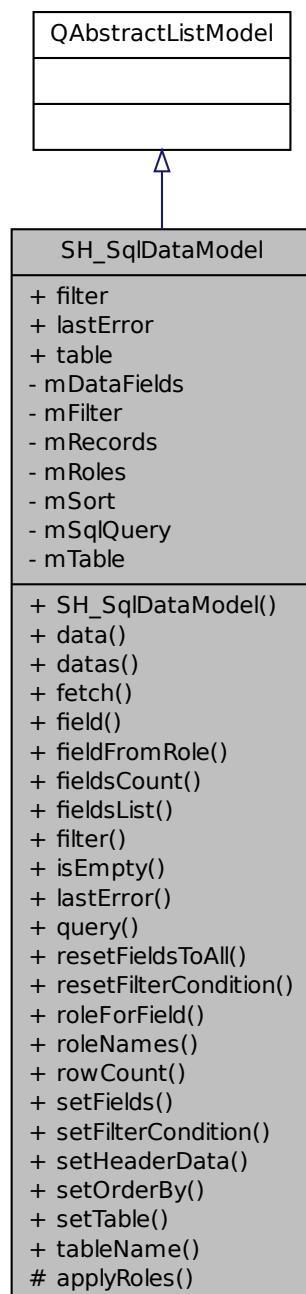
La documentation de cette classe a été générée à partir des fichiers suivants :

- models/[SH_SqlDataField.h](#)
- models/[SH_SqlDataField.cpp](#)

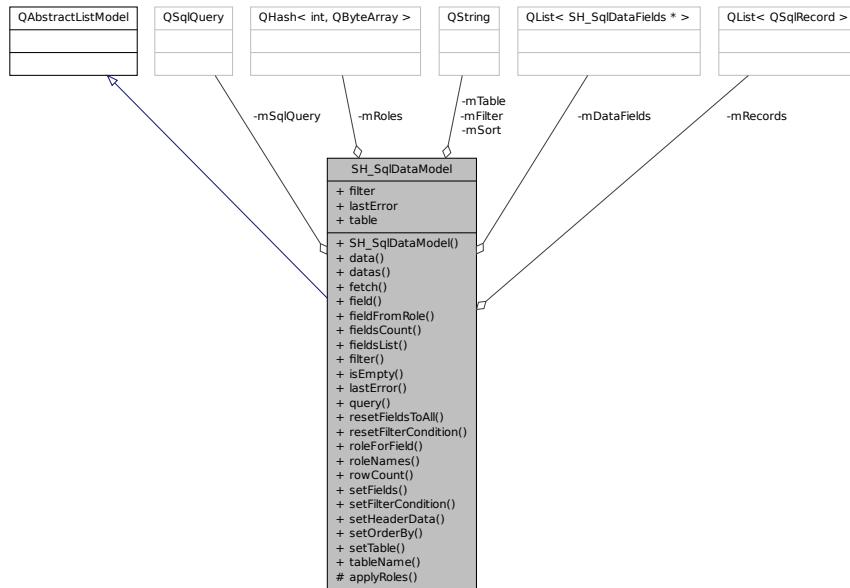
4.60 Référence de la classe SH_SqlDataModel

```
#include <SH_SqlDataModel.h>
```

Graphe d'héritage de SH_SqlDataModel :



Graphe de collaboration de SH_SqlDataModel :



Signaux

- void `fieldsChanged ()`
- void `filterChanged ()`
- void `lastErrorChanged ()`
- void `rolesChanged ()`
- void `tableChanged ()`

Fonctions membres publiques

- `SH_SqlDataModel (QObject *parent=0)`
- `QVariant data (const QModelIndex &index, int role) const`
- `QVariantMap datas () const`
- `bool fetch (QString tableName="", QString filter="", QString sort="", QStringList fields=QStringList())`
- `SH_SqlDataFields * field (int i) const`
- `int fieldFromRole (int role) const`
- `int fieldsCount () const`
- `const QStringList fieldsList () const`
- `const QString & filter () const`
- `bool isEmpty () const`
- `const QString & lastError ()`
- `const QString & query () const`
- `void resetFieldsToAll ()`
- `void resetFilterCondition ()`
- `int roleForField (int fieldIndex) const`
- `virtual QHash<int, QByteArray> roleNames () const`
- `int rowCount (const QModelIndex &parent) const`
- `void setFields (QStringList fieldList)`
- `void setFilterCondition (const QString &filter)`
- `bool setHeaderData (int section, Qt::Orientation orientation, const QVariant &value, int role=Qt::EditRole)`
- `void setOrderBy (QString sort)`
- `void setTable (const QString &tableName)`
- `const QString & tableName () const`

Fonctions membres protégées

- void `applyRoles ()`

Propriétés

- QString **filter**
- QString **lastError**
- QString **table**

Attributs privés

- QList< **SH_SqlDataFields** * > **mDataFields**
mDataFields
- QString **mFilter**
mFilter
- QList< QSqlRecord > **mRecords**
mRecords
- QHash< int, QByteArray > **mRoles**
mRoles
- QString **mSort**
mSort
- QSqlQuery **mSqlQuery**
mSqlQuery
- QString **mTable**
mTable

4.60.1 Description détaillée

Définition à la ligne 14 du fichier [SH_SqlDataModel.h](#).

4.60.2 Documentation des constructeurs et destructeur

4.60.2.1 SH_SqlDataModel : :SH_SqlDataModel (**QObject** * *parent* = 0) [explicit]

Paramètres

<i>parent</i>	
---------------	--

Définition à la ligne 16 du fichier [SH_SqlDataModel.cpp](#).

```
00016
00017     QAbstractListModel(parent)
00018 {
00019 }
```

:

4.60.3 Documentation des fonctions membres

4.60.3.1 void SH_SqlDataModel : :applyRoles () [protected]

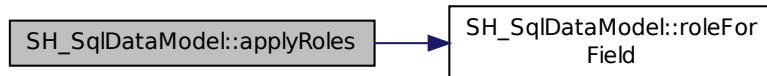
Définition à la ligne 339 du fichier [SH_SqlDataModel.cpp](#).

Références [mDataFields](#), [mRoles](#), [roleForField\(\)](#), et [rolesChanged\(\)](#).

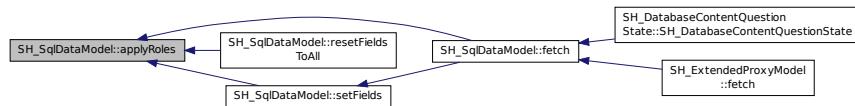
Référencé par [fetch\(\)](#), [resetFieldsToAll\(\)](#), et [setFields\(\)](#).

```
00340 {
00341     this->mRoles.clear();
00342     int nbFields = this->mDataFields.count();
00343     for (int i = 0; i < nbFields; i++)
00344     {
00345         /*MessageManager::infoMessage(QString("nouveau rôle :
00346             %1").arg(QString(this->mDataFields.at(i)->role())));*/
00347         this->mRoles.insert(this->roleForField(i), this-
00348             mDataFields.at(i)->role());
00349     }
00349     emit rolesChanged();
00349 }
```

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appels de cette fonction :



4.60.3.2 QVariant SH_SqlDataModel ::data (const QModelIndex & index, int role) const

Définition à la ligne 39 du fichier [SH_SqlDataModel.cpp](#).

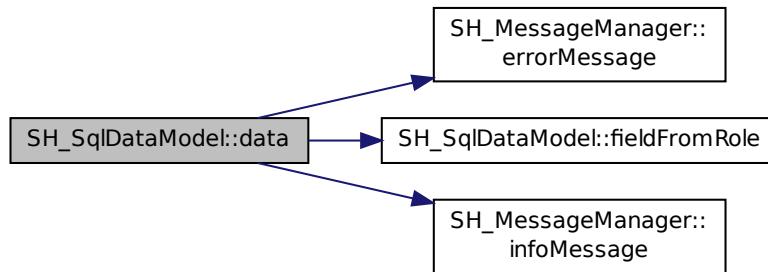
Références [SH_MessageManager ::errorMessage\(\)](#), [fieldFromRole\(\)](#), [SH_MessageManager ::infoMessage\(\)](#), [m-DataFields](#), [mRecords](#), et [mRoles](#).

Référencé par [SH_ExtendedProxyModel ::data\(\)](#).

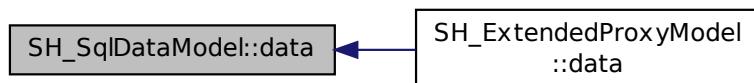
```

00040 {
00041     if (this->mRecords.count() > 0)
00042     {
00043         int row = index.row();
00044         int column = this->fieldFromRole(role);
00045         int nbCols = this->mRoles.count();
00046         if(column >= 0 && column < nbCols) {
00047             SH_MessageManager::infoMessage(QString("row : %1, column : %2,
00048 field: %3 (%4), value : %5\n").arg(index.row()).arg(index.column()).arg(column).arg(QString(this->
00049 mDataFields.at(column)->role())).arg(this->mRecords.at(row).value(column).toString()));
00050             } else{
00051                 SH_MessageManager::errorMessage(QString("rien à retourner pour
00052 %1x%2x%3 (%4>=%5)").arg(index.row()).arg(index.column()).arg(role).arg(column).arg(nbCols));
00053             }
00054             SH_MessageManager::errorMessage("modèle vide");
00055         return QVariant();
00056     }
  
```

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appels de cette fonction :



4.60.3.3 QVariantMap SH_SqlDataModel ::datas() const

Définition à la ligne 62 du fichier [SH_SqlDataModel.cpp](#).

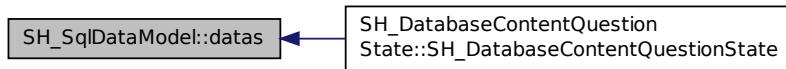
Références [mRecords](#), et [mRoles](#).

Référencé par [SH_DatabaseContentQuestionState ::SH_DatabaseContentQuestionState\(\)](#).

```

00063 {
00064     qDebug() << "datas";
00065     QVariantMap result;
00066     if (this->mRecords.count() > 0)
00067     {
00068         qDebug() << "datas ok";
00069         for(int column = 0; column < this->mRoles.count(); column++) {
00070             for(int row = 0; row < this->mRecords.count();row++) {
00071                 qDebug() << "data inserted";
00072                 result.insertMulti(this->mRoles.value(column),this->
00073                         mRecords.at(row).value(column));
00074             }
00075         }
00076     }
00077 }
```

Voici le graphe des appelants de cette fonction :



4.60.3.4 bool SH_SqlDataModel ::fetch (QString tableName = "", QString filter = "", QString sort = "", QStringList fields = QStringList ())

Définition à la ligne 194 du fichier [SH_SqlDataModel.cpp](#).

Références [applyRoles\(\)](#), [SH_MessageManager ::errorMessage\(\)](#), [SH_DatabaseManager ::execSelectQuery\(\)](#), [field\(\)](#), [fieldsChanged\(\)](#), [fieldsList\(\)](#), [SH_DatabaseManager ::getInstance\(\)](#), [SH_MessageManager ::infoMessage\(\)](#), [isEmpty\(\)](#), [lastError\(\)](#), [mDataFields](#), [mFilter](#), [mRecords](#), [mSort](#), [mSqlQuery](#), [mTable](#), [SH_SqlDataFields ::name](#), [setFields\(\)](#), [setFilterCondition\(\)](#), [SH_SqlDataFields ::setName\(\)](#), [setOrderBy\(\)](#), et [setTable\(\)](#).

Référencé par [SH_ExtendedProxyModel ::fetch\(\)](#), et [SH_DatabaseContentQuestionState ::SH_DatabaseContentQuestionState\(\)](#).

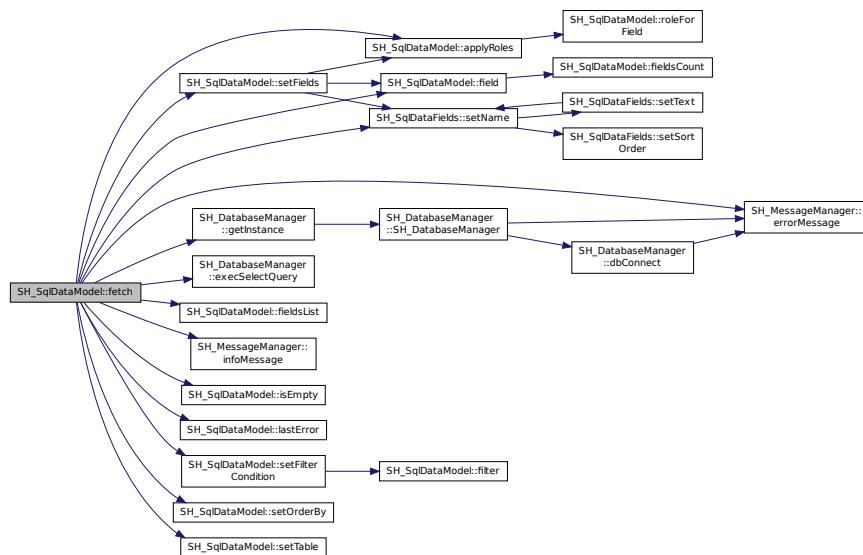
```

00195 {
00196     if(!mTable.isEmpty() || !tableName.isEmpty()) {
00197         SH_MessageManager::infoMessage("Bienvenue dans fetch");
00198         qDebug() << mTable << " " << this->fieldsList().join(", ") << " " <<
00199         mFilter << " " << mSort;
00200         this->setFields(fieldsList);
00201         this->setTable(tableName);
00202         this->setFilterCondition(filter);
00203         this->setOrderBy(sort);
00204         qDebug() << tableName << " " << filter << " " << sort << " " <<
00205         fieldsList.join(", ");
00206         try
00207         {
00208             beginResetModel();
00209             mRecords.clear();
00210             endResetModel();
00211             qDebug() << mTable << " " << this->fieldsList() << " " <<
00212             mFilter << " " << mSort;
00213             mSqlQuery = SH_DatabaseManager::getInstance()->
00214             execSelectQuery(mTable, this->fieldsList(),
00215             mFilter, mSort);
00216             qDebug() << mSqlQuery.executedQuery();
00217             bool next = mSqlQuery.next();
00218             if(next)
00219             {
00220                 qDebug() << "next ok";
00221             }
00222             while (next) /* && mSqlQuery.isActive() */
00223             {
00224                 QSqlRecord record = mSqlQuery.record();
00225                 qDebug() << "\n\n";
00226                 SH_MessageManager::infoMessage("Nouvelle ligne récupérée");
00227                 SH_MessageManager::infoMessage(QString("%1 champs").arg(
00228                     record.count()));
00229                 if (mSqlQuery.isValid() && (!record.isEmpty()) && (record.count() > 0))
00230                 {
00231                     beginInsertRows(QModelIndex(), 0, 0);
00232                     mRecords.append(record);
00233                     int nbFields = record.count();
00234                     for (int i = 0; i < nbFields; i++)
00235                     {
00236                         SH_MessageManager::infoMessage(QString("%1 : %2").arg(
00237                             record.fieldName(i)).arg(record.value(i).toString()));
00238                     }
00239                     if (mDataFields.empty())
00240                     {
00241                         int nbFields = record.count();
00242                         for (int i = 0; i < nbFields; i++)
00243                         {
00244                             SH_SqlDataFields *field = new
00245                             SH_SqlDataFields();
00246                             field->setName(record.fieldName(i));
00247                         }
00248                     }
00249                 }
00250             }
00251         }
00252     }
00253 }
  
```

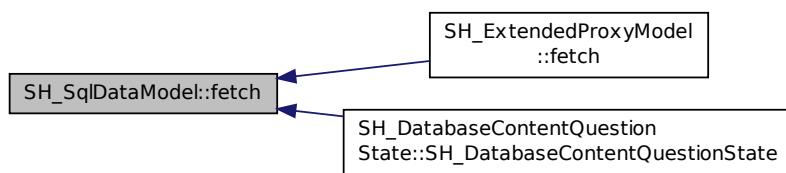
```

00238     SH_MessageManager::infoMessage(QString("nouveau
00239     champ (le n°%l): %2").arg(i).arg(field->name()));
00240             mDataFields.append(field);
00241         }
00242         this->applyRoles();
00243         emit fieldsChanged();
00244     }
00245     endInsertRows();
00246     next = mSqlQuery.next();
00247 }
00248 }
00249 catch (const std::exception &e)
00250 {
00251     SH_MessageManager::errorMessage(e.what(), "exception");
00252     if (this->lastError().isEmpty())
00253     {
00254         SH_MessageManager::errorMessage(this->
00255         lastError(), "erreur SQL");
00256     }
00257     if (this->lastError().isEmpty())
00258     {
00259         SH_MessageManager::errorMessage(this->
00260         lastError(), "erreur SQL");
00261     }
00262 return (!this->isEmpty());
00263 }
```

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appels de cette fonction :



4.60.3.5 SH_SqlDataFields * SH_SqlDataModel ::field (int *i*) const

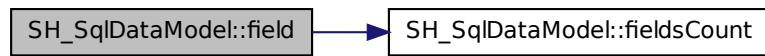
Définition à la ligne 271 du fichier [SH_SqlDataModel.cpp](#).

Références [fieldsCount\(\)](#), et [mDataFields](#).

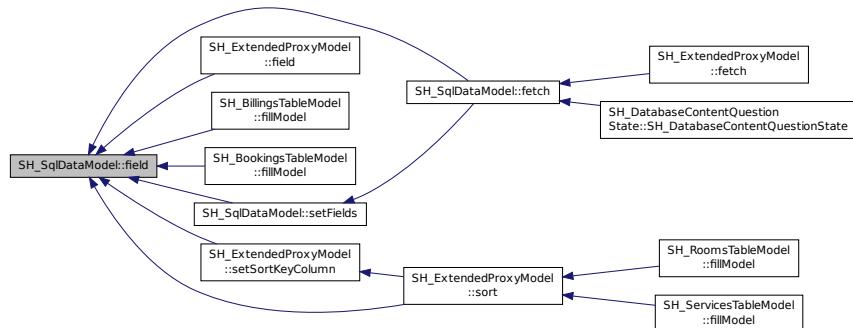
Référencé par [fetch\(\)](#), [SH_ExtendedProxyModel ::field\(\)](#), [SH_BillingsTableModel ::fillModel\(\)](#), [SH_BookingsTableModel ::fillModel\(\)](#), [setFields\(\)](#), [SH_ExtendedProxyModel ::setSortKeyColumn\(\)](#), et [SH_ExtendedProxyModel ::sort\(\)](#).

```
00272 {
00273     i = qMin(i, this->fieldsCount () -1);
00274     i = qMax(i, 0);
00275     return this->mDataFields.at (i);
00276 }
```

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :

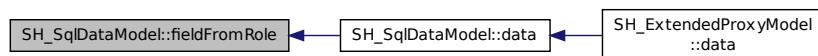
4.60.3.6 int SH_SqlDataModel ::fieldFromRole (int *role*) const [inline]

Définition à la ligne 81 du fichier [SH_SqlDataModel.h](#).

Référencé par [data\(\)](#).

```
00081 { return role - Qt::UserRole; }
```

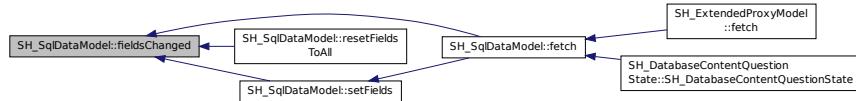
Voici le graphe des appelants de cette fonction :



4.60.3.7 void SH_SqlDataModel : :fieldsChanged () [signal]

Référencé par [fetch\(\)](#), [resetFieldsToAll\(\)](#), et [setFields\(\)](#).

Voici le graphe des appelants de cette fonction :



4.60.3.8 int SH_SqlDataModel : :fieldsCount () const

Définition à la ligne 358 du fichier [SH_SqlDataModel.cpp](#).

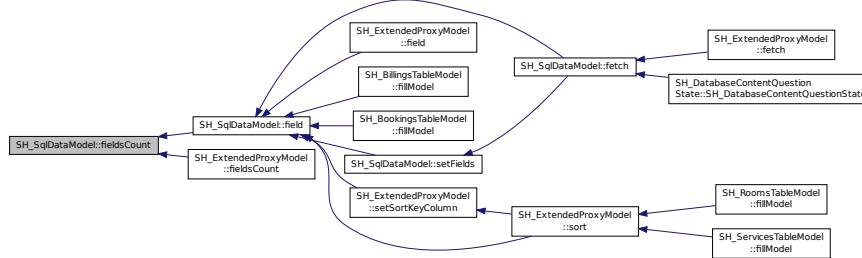
Références [mDataFields](#).

Référencé par [field\(\)](#), et [SH_ExtendedProxyModel : :fieldsCount\(\)](#).

```

00359 {
00360     return mDataFields.count();
00361 }
  
```

Voici le graphe des appelants de cette fonction :



4.60.3.9 const QStringList SH_SqlDataModel : :fieldsList () const

Définition à la ligne 134 du fichier [SH_SqlDataModel.cpp](#).

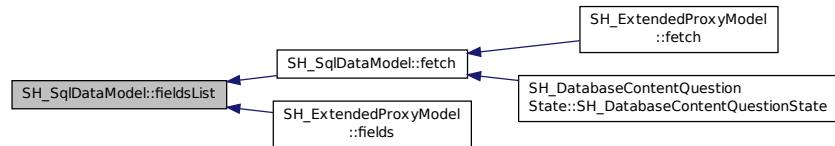
Références [mDataFields](#).

Référencé par [fetch\(\)](#), et [SH_ExtendedProxyModel : :fields\(\)](#).

```

00135 {
00136     QStringList fields;
00137     if(!this->mDataFields.isEmpty()) {
00138         int c = mDataFields.count();
00139         for (int i = 0; i < c; i++)
00140         {
00141             fields.append(this->mDataFields.at(i)->name());
00142         }
00143     }
00144     return fields;
00145 }
  
```

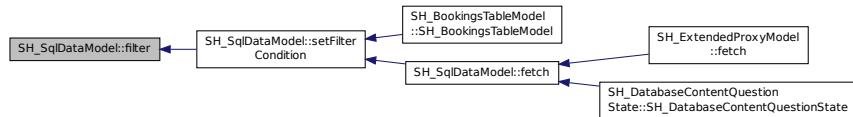
Voici le graphe des appelants de cette fonction :



4.60.3.10 const QString& SH_SqlDataModel ::filter() const

Référencé par [setFilterCondition\(\)](#).

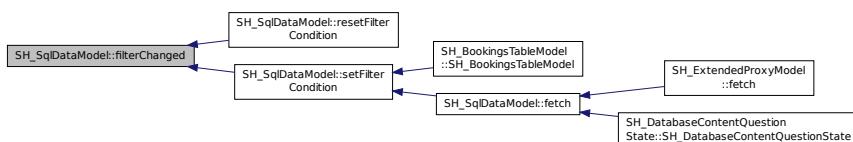
Voici le graphe des appelants de cette fonction :



4.60.3.11 void SH_SqlDataModel ::filterChanged() [signal]

Référencé par [resetFilterCondition\(\)](#), et [setFilterCondition\(\)](#).

Voici le graphe des appelants de cette fonction :



4.60.3.12 bool SH_SqlDataModel ::isEmpty() const

Définition à la ligne 380 du fichier [SH_SqlDataModel.cpp](#).

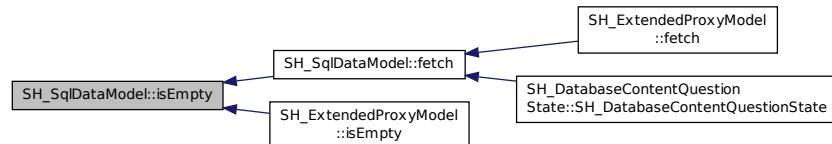
Références [mRecords](#).

Référencé par [fetch\(\)](#), et [SH_ExtendedProxyModel ::isEmpty\(\)](#).

```

00381 {
00382     return mRecords.empty();
00383 }
```

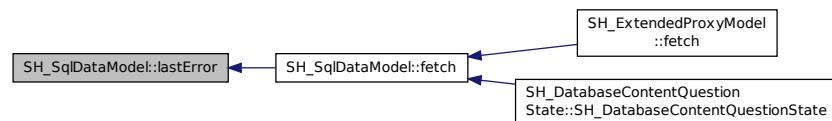
Voici le graphe des appelants de cette fonction :



4.60.3.13 const QString& SH_SqlDataModel::lastError()

Référencé par [fetch\(\)](#).

Voici le graphe des appelants de cette fonction :



4.60.3.14 void SH_SqlDataModel::lastErrorChanged() [signal]

4.60.3.15 const QString & SH_SqlDataModel::query() const

Définition à la ligne 101 du fichier [SH_SqlDataModel.cpp](#).

Références [mSqlQuery](#).

```

00102 {
00103     return mSqlQuery.lastQuery();
00104 }
```

4.60.3.16 void SH_SqlDataModel::resetFieldsToAll()

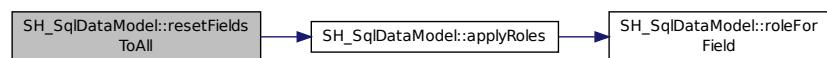
Définition à la ligne 306 du fichier [SH_SqlDataModel.cpp](#).

Références [applyRoles\(\)](#), [fieldsChanged\(\)](#), et [mDataFields](#).

```

00307 {
00308     mDataFields.clear();
00309     this->applyRoles();
00310     emit fieldsChanged();
00311 }
```

Voici le graphe d'appel pour cette fonction :



4.60.3.17 void SH_SqlDataModel ::resetFilterCondition ()

Définition à la ligne 182 du fichier [SH_SqlDataModel.cpp](#).

Références [filterChanged\(\)](#), et [mFilter](#).

```
00183 {
00184     mFilter = "";
00185     emit filterChanged();
00186 }
```

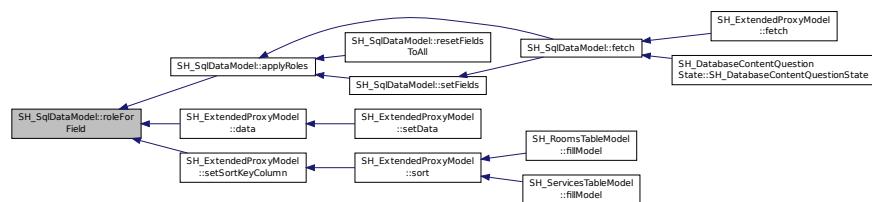
4.60.3.18 int SH_SqlDataModel ::roleForField (int fieldIndex) const [inline]

Définition à la ligne 73 du fichier [SH_SqlDataModel.h](#).

Référencé par [applyRoles\(\)](#), [SH_ExtendedProxyModel ::data\(\)](#), et [SH_ExtendedProxyModel ::setSortKeyColumn\(\)](#).

```
00073 { return UserRole + fieldIndex; }
```

Voici le graphe des appelants de cette fonction :



4.60.3.19 virtual QHash<int, QByteArray> SH_SqlDataModel ::roleNames () const [inline], [virtual]

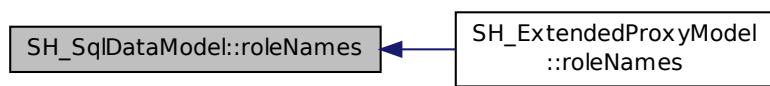
Définition à la ligne 179 du fichier [SH_SqlDataModel.h](#).

Références [mRoles](#).

Référencé par [SH_ExtendedProxyModel ::roleNames\(\)](#).

```
00179 { return this->mRoles; }
```

Voici le graphe des appelants de cette fonction :



4.60.3.20 void SH_SqlDataModel ::rolesChanged() [signal]

Référencé par [applyRoles\(\)](#).

Voici le graphe des appelants de cette fonction :



4.60.3.21 int SH_SqlDataModel ::rowCount (const QModelIndex & parent) const

Définition à la ligne [27](#) du fichier [SH_SqlDataModel.cpp](#).

Références [mRecords](#).

```

00028 {
00029     return mRecords.count();
00030 }
  
```

4.60.3.22 void SH_SqlDataModel ::setFields (QStringList fieldList)

Définition à la ligne [284](#) du fichier [SH_SqlDataModel.cpp](#).

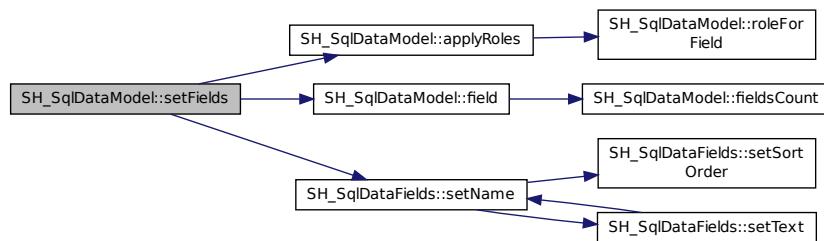
Références [applyRoles\(\)](#), [field\(\)](#), [fieldsChanged\(\)](#), [mDataFields](#), et [SH_SqlDataFields ::setName\(\)](#).

Référencé par [fetch\(\)](#).

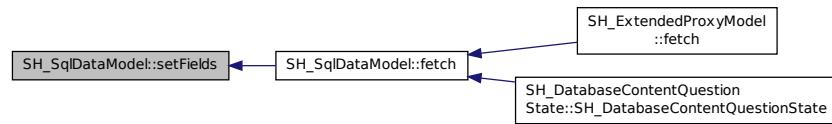
```

00285 {
00286     fields.removeDuplicates();
00287     int nbFields = fields.count();
00288     if (nbFields > 0)
00289     {
00290         for (int i = 0; i < nbFields; i++)
00291         {
00292             SH_SqlDataFields *field = new SH_SqlDataFields();
00293             field->setName(fields.at(i));
00294             mDataFields.append(field);
00295         }
00296         this->applyRoles();
00297         emit fieldsChanged();
00298     }
00299 }
  
```

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :



4.60.3.23 void SH_SqlDataModel ::setFilterCondition (const QString & filter)

Définition à la ligne 168 du fichier [SH_SqlDataModel.cpp](#).

Références [filter\(\)](#), [filterChanged\(\)](#), et [mFilter](#).

Référencé par [fetch\(\)](#), et [SH_BookingsTableModel ::SH_BookingsTableModel\(\)](#).

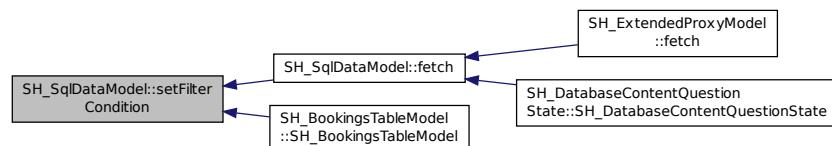
```

00169 {
00170     if (mFilter != filter && filter != "")
00171     {
00172         mFilter = filter;
00173         emit filterChanged();
00174     }
00175 }
  
```

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :



4.60.3.24 bool SH_SqlDataModel ::setHeaderData (int section, Qt ::Orientation orientation, const QVariant & value, int role = Qt ::EditRole)

Définition à la ligne 85 du fichier [SH_SqlDataModel.cpp](#).

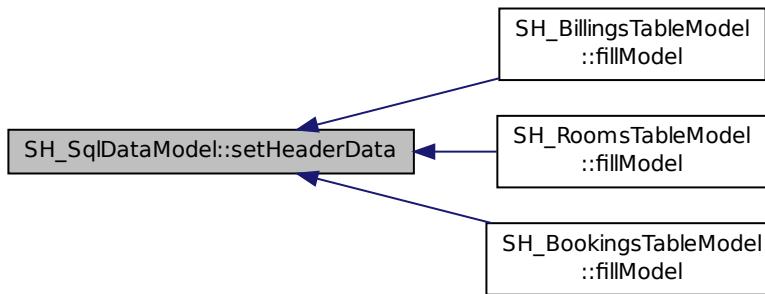
Références [mDataFields](#).

Référencé par [SH_BillingsTableModel ::fillModel\(\)](#), [SH_RoomsTableModel ::fillModel\(\)](#), et [SH_BookingsTableModel ::fillModel\(\)](#).

```

00086 {
00087     Q_UNUSED(role);
00088     if (orientation == Qt::Horizontal)
00089     {
00090         this->mDataFields.at(section)->setText(value.toString());
00091         return (this->mDataFields.at(section)->text() == value.toString());
00092     }
00093     return false;
00094 }
```

Voici le graphe des appelants de cette fonction :



4.60.3.25 void SH_SqlDataModel ::setOrderBy (QString sort)

Définition à la ligne 369 du fichier [SH_SqlDataModel.cpp](#).

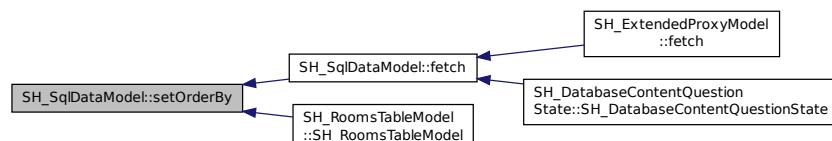
Références [mSort](#).

Référencé par [fetch\(\)](#), et [SH_RoomsTableModel ::SH_RoomsTableModel\(\)](#).

```

00370 {
00371     this->mSort = sort;
00372 }
```

Voici le graphe des appelants de cette fonction :



4.60.3.26 void SH_SqlDataModel ::setTable (const QString & tableName)

Définition à la ligne 153 du fichier [SH_SqlDataModel.cpp](#).

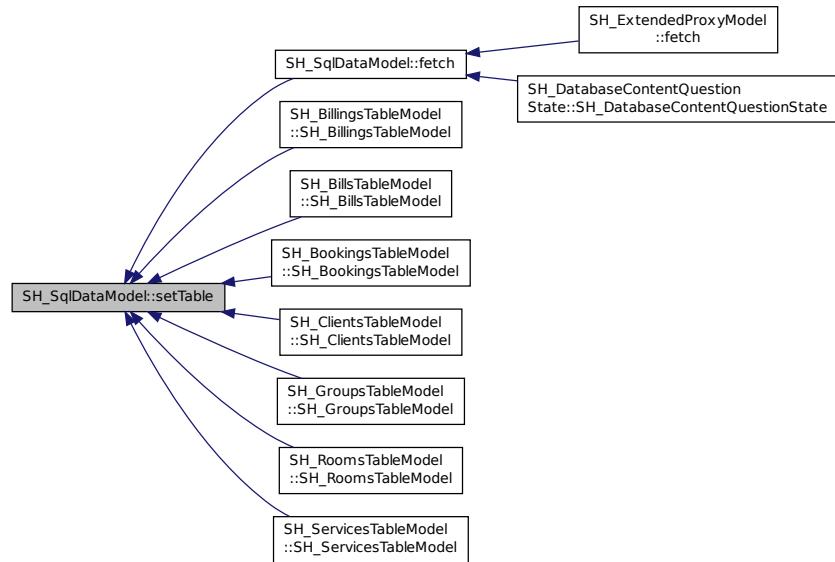
Références [mTable](#), et [tableChanged\(\)](#).

Référencé par [fetch\(\)](#), [SH_BillingsTableModel ::SH_BillingsTableModel\(\)](#), [SH_BillsTableModel ::SH_BillsTableModel\(\)](#), [SH_BookingsTableModel ::SH_BookingsTableModel\(\)](#), [SH_ClientsTableModel ::SH_ClientsTableModel\(\)](#),

`SH_GroupsTableModel` : `:SH_GroupsTableModel()`, `SH_RoomsTableModel` : `:SH_RoomsTableModel()`, et `SH_ServicesTableModel` : `:SH_ServicesTableModel()`.

```
00154 {
00155     if (mTable.toUpper() != tableName.toUpper() && tableName != "") {
00156     {
00157         mTable = tableName.toUpper();
00158         emit tableChanged();
00159     }
00160 }
```

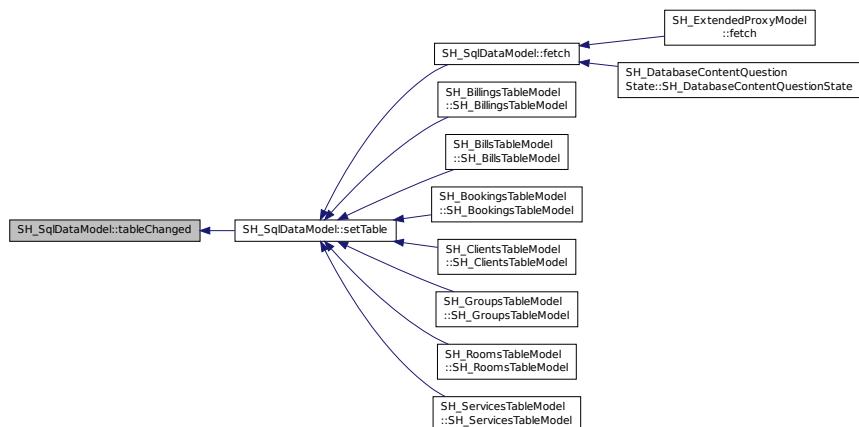
Voici le graphe des appelants de cette fonction :



4.60.3.27 void SH_SqlDataModel::tableChanged() [signal]

Référencé par `setTable()`.

Voici le graphe des appelants de cette fonction :



4.60.3.28 const QString & SH_SqlDataModel::tableName() const

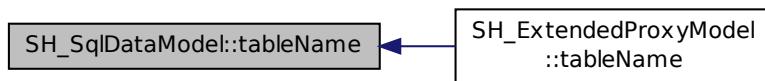
Définition à la ligne 112 du fichier [SH_SqlDataModel.cpp](#).

Références [mTable](#).

Référencé par [SH_ExtendedProxyModel::tableName\(\)](#).

```
00113 {
00114     return mTable;
00115 }
```

Voici le graphe des appelants de cette fonction :



4.60.4 Documentation des données membres

4.60.4.1 QList<SH_SqlDataFields *> SH_SqlDataModel::mDataFields [private]

`mDataFields`

Définition à la ligne 258 du fichier [SH_SqlDataModel.h](#).

Référencé par [applyRoles\(\)](#), [data\(\)](#), [fetch\(\)](#), [field\(\)](#), [fieldsCount\(\)](#), [fieldsList\(\)](#), [resetFieldsToAll\(\)](#), [setFields\(\)](#), et [setHeaderData\(\)](#).

4.60.4.2 QString SH_SqlDataModel::mFilter [private]

`mFilter`

Définition à la ligne 250 du fichier [SH_SqlDataModel.h](#).

Référencé par [fetch\(\)](#), [resetFilterCondition\(\)](#), et [setFilterCondition\(\)](#).

4.60.4.3 QList<QSqlRecord> SH_SqlDataModel::mRecords [private]

`mRecords`

Définition à la ligne 270 du fichier [SH_SqlDataModel.h](#).

Référencé par [data\(\)](#), [datas\(\)](#), [fetch\(\)](#), [isEmpty\(\)](#), et [rowCount\(\)](#).

4.60.4.4 QHash<int, QByteArray> SH_SqlDataModel::mRoles [private]

`mRoles`

Définition à la ligne 262 du fichier [SH_SqlDataModel.h](#).

Référencé par [applyRoles\(\)](#), [data\(\)](#), [datas\(\)](#), et [roleNames\(\)](#).

4.60.4.5 `QString SH_SqlDataModel::mSort [private]`

mSort

Définition à la ligne 254 du fichier [SH_SqlDataModel.h](#).

Référencé par [fetch\(\)](#), et [setOrderBy\(\)](#).

4.60.4.6 `QSqQuery SH_SqlDataModel::mSqlQuery [private]`

mSqlQuery

Définition à la ligne 266 du fichier [SH_SqlDataModel.h](#).

Référencé par [fetch\(\)](#), et [query\(\)](#).

4.60.4.7 `QString SH_SqlDataModel::mTable [private]`

mTable

Définition à la ligne 246 du fichier [SH_SqlDataModel.h](#).

Référencé par [fetch\(\)](#), [setTable\(\)](#), et [tableName\(\)](#).

4.60.5 Documentation des propriétés

4.60.5.1 `const QString & SH_SqlDataModel::filter [read], [write]`

Définition à la ligne 18 du fichier [SH_SqlDataModel.h](#).

4.60.5.2 `const QString & SH_SqlDataModel::lastError [read]`

Définition à la ligne 19 du fichier [SH_SqlDataModel.h](#).

Référencé par [SH_ExtendedProxyModel::lastError\(\)](#).

4.60.5.3 `QString SH_SqlDataModel::table [read], [write]`

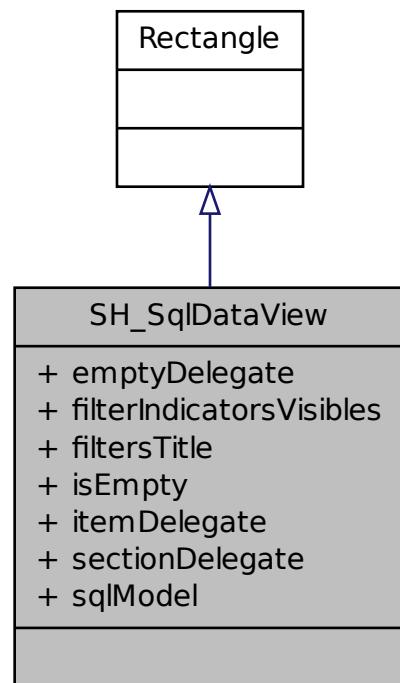
Définition à la ligne 17 du fichier [SH_SqlDataModel.h](#).

La documentation de cette classe a été générée à partir des fichiers suivants :

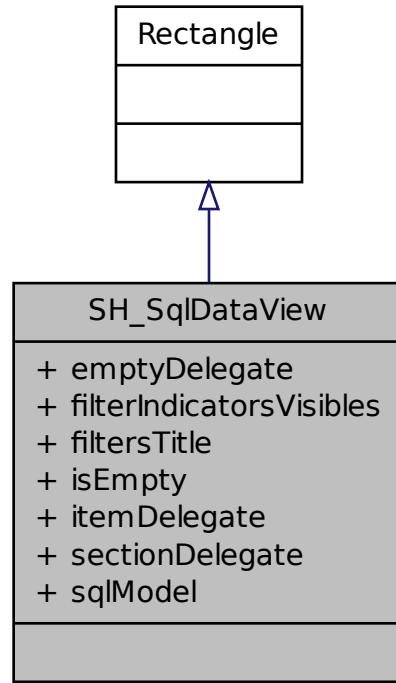
- models/[SH_SqlDataModel.h](#)
- models/[SH_SqlDataModel.cpp](#)

4.61 Référence de la classe SH_SqlTableView

Graphe d'héritage de SH_SqlTableView :



Graphe de collaboration de SH_Sql DataView :



Signaux

- void `newItem()`
- void `selected(string selectedItem)`

Propriétés

- string `emptyDelegate`
- bool `filterIndicatorsVisibles`
- string `filtersTitle`
- bool `isEmpty`
- string `itemDelegate`
- string `sectionDelegate`
- var `sqlModel`

4.61.1 Description détaillée

Définition à la ligne 4 du fichier [SH_Sql DataView.qml](#).

4.61.2 Documentation des fonctions membres

4.61.2.1 void `SH_Sql DataView::newItem()` [signal]

4.61.2.2 void `SH_Sql DataView::selected(string selectedItem)` [signal]

4.61.3 Documentation des propriétés

4.61.3.1 **string SH_Sql DataView : :emptyDelegate**

Définition à la ligne 11 du fichier [SH_Sql DataView.qml](#).

4.61.3.2 **bool SH_Sql DataView : :filterIndicatorsVisible**

type : bool visibilité des cases à cocher permettant le tri du modèle

Définition à la ligne 17 du fichier [SH_Sql DataView.qml](#).

4.61.3.3 **string SH_Sql DataView : :filtersTitle**

Définition à la ligne 19 du fichier [SH_Sql DataView.qml](#).

4.61.3.4 **bool SH_Sql DataView : :isEmpty**

Définition à la ligne 21 du fichier [SH_Sql DataView.qml](#).

4.61.3.5 **string SH_Sql DataView : :itemDelegate**

Définition à la ligne 9 du fichier [SH_Sql DataView.qml](#).

4.61.3.6 **string SH_Sql DataView : :sectionDelegate**

Définition à la ligne 13 du fichier [SH_Sql DataView.qml](#).

4.61.3.7 **var SH_Sql DataView : :sqlModel**

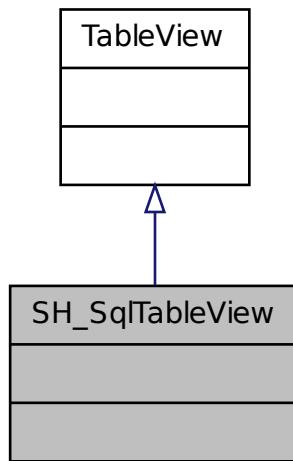
Définition à la ligne 7 du fichier [SH_Sql DataView.qml](#).

La documentation de cette classe a été générée à partir du fichier suivant :

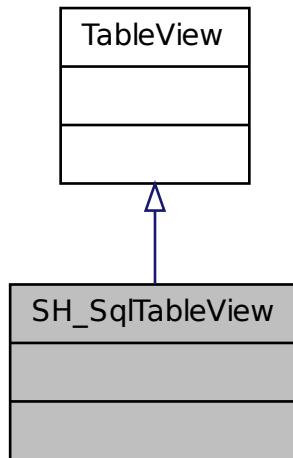
– views/qml/[SH_Sql DataView.qml](#)

4.62 Référence de la classe SH_SqlTableView

Graphe d'héritage de SH_SqlTableView :



Graphe de collaboration de SH_SqlTableView :



Signaux

- void **selected** (string selectedItem)

4.62.1 Description détaillée

Définition à la ligne 4 du fichier [SH_SqlTableView.qml](#).

4.62.2 Documentation des fonctions membres

4.62.2.1 void SH_SqlTableView : :selected (string *selectedItem*) [signal]

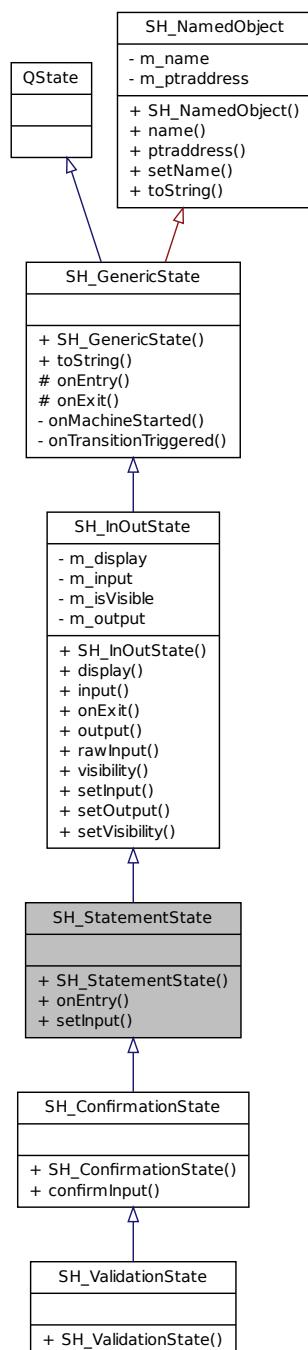
La documentation de cette classe a été générée à partir du fichier suivant :

– views/qml/[SH_SqlTableView.qml](#)

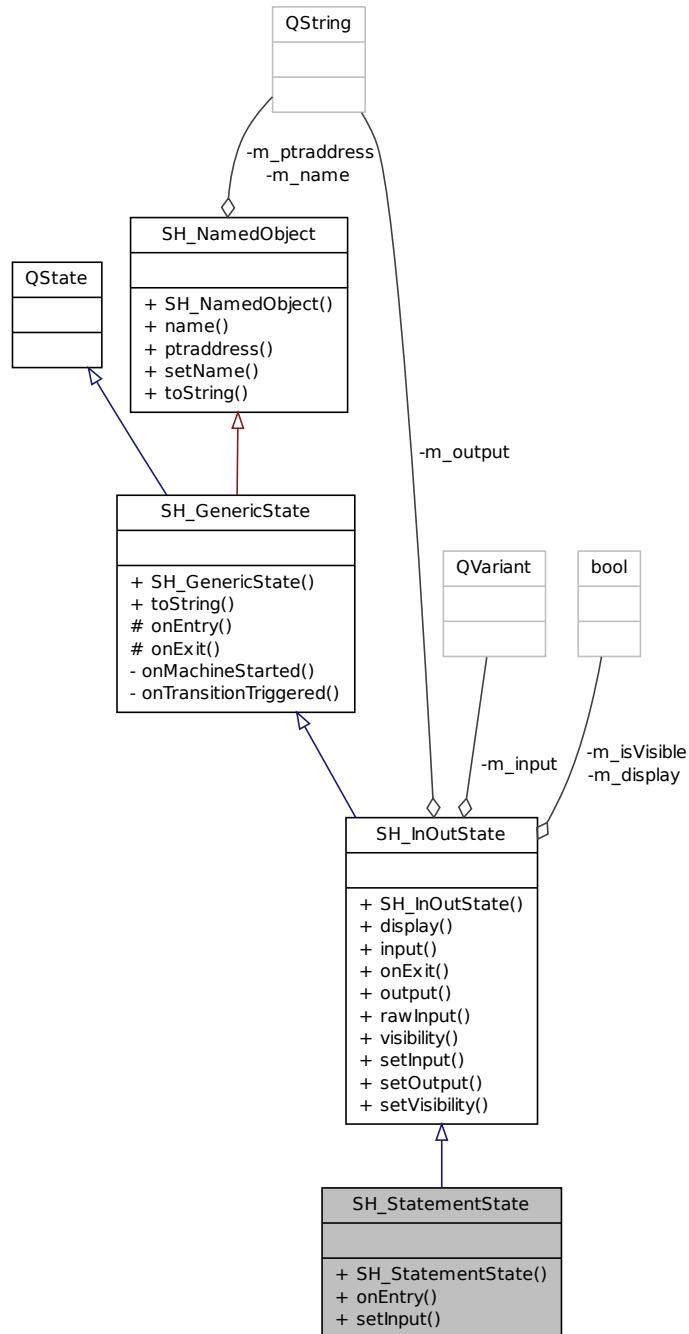
4.63 Référence de la classe SH_StatementState

```
#include <SH_StatementState.h>
```

Graphe d'héritage de SH_StatementState :



Graphe de collaboration de SH_StatementState :



Connecteurs publics

- virtual void `setOutput` (const `QString` &`output`)
- virtual void `setVisibility` (`bool` `isVisible`)

Signaux

- void `next` ()

- void [resendInput](#) (QVariant *input*)
- void [sendOutput](#) (QVariant *output*)

Fonctions membres publiques

- [SH_StatementState](#) (QString *output*, QString *name*, QState **parent*=0)
- void [display](#) (bool *canDisplay*)
- virtual QVariant [input](#) () const
- void [onEntry](#) (QEvent **event*)
- void [onExit](#) (QEvent **event*)
- virtual QString [output](#) () const
- virtual QVariant [rawInput](#) () const
- void [setInput](#) (const QVariant &*input*)
- QString [toString](#) ()
- bool [visibility](#) ()

4.63.1 Description détaillée

Définition à la ligne 10 du fichier [SH_StatementState.h](#).

4.63.2 Documentation des constructeurs et destructeur

4.63.2.1 SH_StatementState : :SH_StatementState (QString *output*, QString *name*, QState * *parent* = 0)

Paramètres

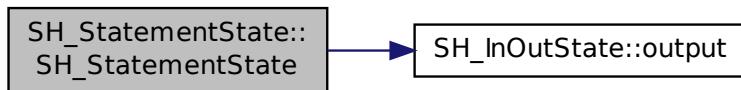
<i>output</i>	
<i>name</i>	
<i>parent</i>	

Définition à la ligne 9 du fichier [SH_StatementState.cpp](#).

Références [SH_InOutState](#) : :[output\(\)](#).

```
00009
00010     SH_InOutState(output, name, parent)
00011 {
00012     qDebug() << "salut ! " << output;
00013 }
```

Voici le graphe d'appel pour cette fonction :



4.63.3 Documentation des fonctions membres

4.63.3.1 void SH_InOutState : :display (bool *canDisplay*) [inherited]

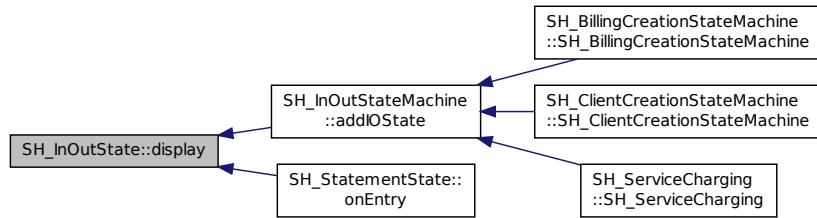
Définition à la ligne 95 du fichier [SH_IOState.cpp](#).

Références [SH_InOutState](#) : :[m_display](#), [SH_InOutState](#) : :[m_isVisible](#), [SH_InOutState](#) : :[m_output](#), et [SH_InOutState](#) : :[sendOutput\(\)](#).

Référencé par [SH_InOutStateMachine ::addIOState\(\)](#), et [onEntry\(\)](#).

```
00096 {
00097     m_display=canDisplay;
00098     if(m_display && !m_output.isEmpty() && m_isVisible) {
00099         qDebug() << "resalut !" << QVariant(m_output);
00100         emit sendOutput(QVariant(m_output));
00101     }
00102 }
```

Voici le graphe des appels de cette fonction :



4.63.3.2 QVariant SH_InOutState ::input() const [virtual], [inherited]

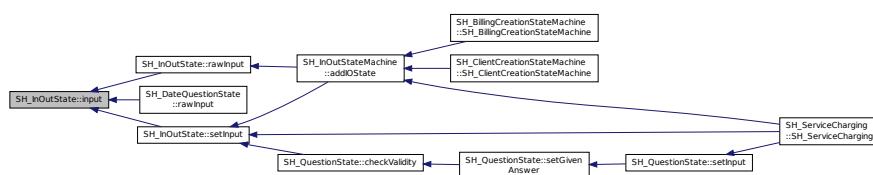
Définition à la ligne 20 du fichier [SH_IOState.cpp](#).

Références [SH_InOutState ::m_input](#).

Référencé par [SH_InOutState ::rawInput\(\)](#), [SH_DateQuestionState ::rawInput\(\)](#), et [SH_InOutState ::setInput\(\)](#).

```
00021 {
00022     return m_input;
00023 }
```

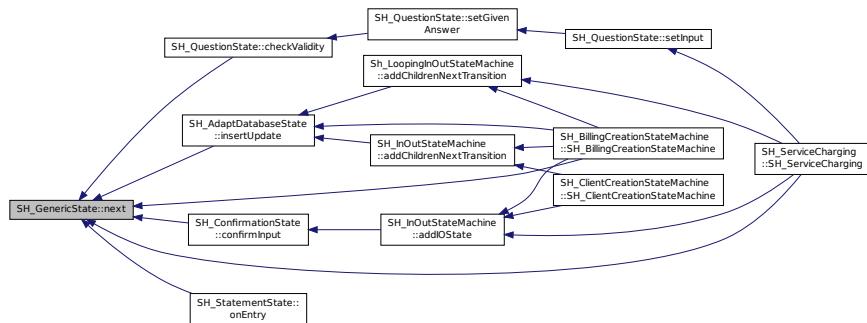
Voici le graphe des appels de cette fonction :



4.63.3.3 void SH_GenericState ::next() [signal], [inherited]

Référencé par [SH_QuestionState ::checkValidity\(\)](#), [SH_ConfirmationState ::confirmInput\(\)](#), [SH_AdaptDatabaseState ::insertUpdate\(\)](#), [onEntry\(\)](#), [SH_BillingCreationStateMachine ::SH_BillingCreationStateMachine\(\)](#), et [SH_ServiceCharging ::SH_ServiceCharging\(\)](#).

Voici le graphe des appels de cette fonction :



4.63.3.4 void SH_StatementState ::onEntry (QEvent * event)

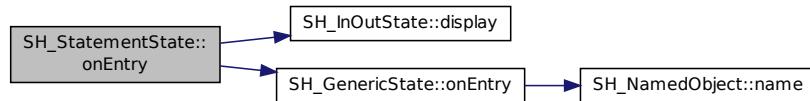
Définition à la ligne 33 du fichier [SH_StatementState.cpp](#).

Références [SH_InOutState ::display\(\)](#), [SH_GenericState ::next\(\)](#), et [SH_GenericState ::onEntry\(\)](#).

```

00034 {
00035     SH_GenericState::onEntry(event);
00036     display(true);
00037     emit next();
00038 }
  
```

Voici le graphe d'appel pour cette fonction :



4.63.3.5 void SH_InOutState ::onExit (QEvent * event) [inherited]

Définition à la ligne 110 du fichier [SH_IOState.cpp](#).

Références [SH_InOutState ::m_input](#), [SH_InOutState ::m_isVisible](#), [SH_GenericState ::onExit\(\)](#), et [SH_InOutState ::resendInput\(\)](#).

```

00111 {
00112     SH_GenericState::onExit(event);
00113     if(m_isVisible) {
00114         emit resendInput(m_input);
00115     }
00116 }
  
```

Voici le graphe d'appel pour cette fonction :



4.63.3.6 QString SH_InOutState ::output() const [virtual], [inherited]

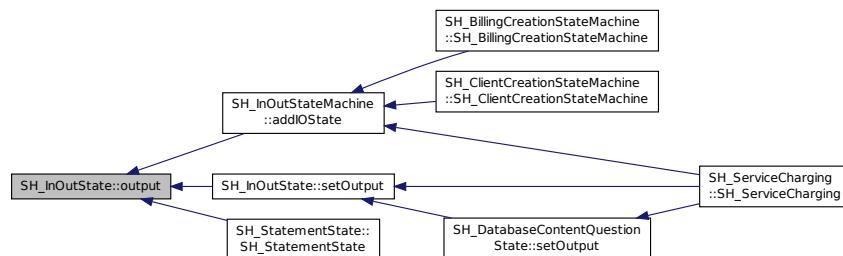
Définition à la ligne 56 du fichier [SH_IOState.cpp](#).

Références [SH_InOutState ::m_output](#).

Référencé par [SH_InOutStateMachine ::addIOState\(\)](#), [SH_InOutState ::setOutput\(\)](#), et [SH_StatementState\(\)](#).

```
00057 {
00058     return m_output;
00059 }
```

Voici le graphe des appels de cette fonction :



4.63.3.7 QVariant SH_InOutState ::rawInput() const [virtual], [inherited]

Réimplémentée dans [SH_DateQuestionState](#), et [SH_DatabaseContentQuestionState](#).

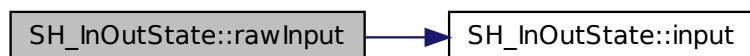
Définition à la ligne 30 du fichier [SH_IOState.cpp](#).

Références [SH_InOutState ::input\(\)](#).

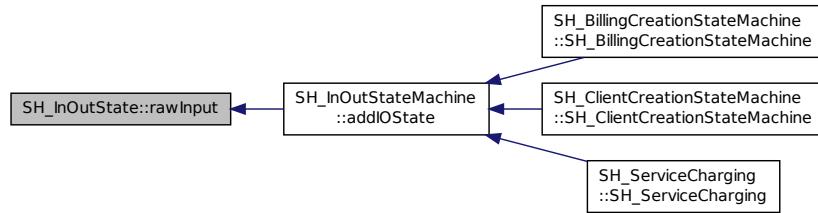
Référencé par [SH_InOutStateMachine ::addIOState\(\)](#).

```
00031 {
00032     return input();
00033 }
```

Voici le graphe d'appel pour cette fonction :



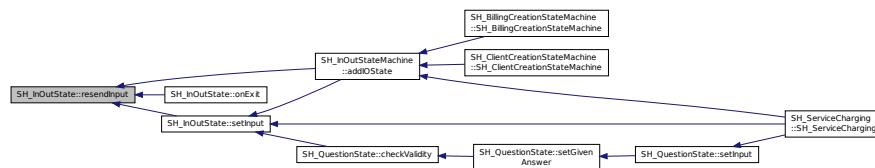
Voici le graphe des appelants de cette fonction :



4.63.3.8 void SH_InOutState ::resendInput (QVariant input) [signal], [inherited]

Référencé par [SH_InOutStateMachine ::addIOState\(\)](#), [SH_InOutState ::onExit\(\)](#), et [SH_InOutState ::setInput\(\)](#).

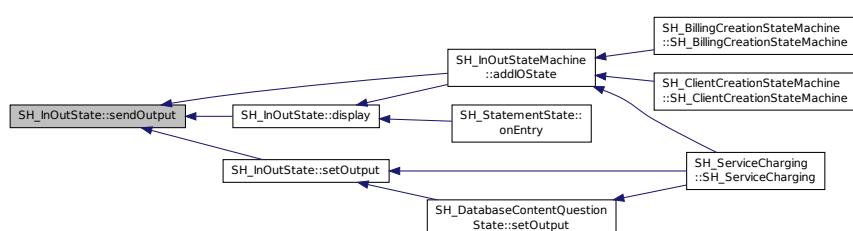
Voici le graphe des appelants de cette fonction :



4.63.3.9 void SH_InOutState ::sendOutput (QVariant output) [signal], [inherited]

Référencé par [SH_InOutStateMachine ::addIOState\(\)](#), [SH_InOutState ::display\(\)](#), et [SH_InOutState ::setOutput\(\)](#).

Voici le graphe des appelants de cette fonction :



4.63.3.10 void SH_StatementState ::setInput (const QVariant & input) [virtual]

Réimplémentée à partir de [SH_InOutState](#).

Définition à la ligne 21 du fichier [SH_StatementState.cpp](#).

```

00022 {
00023     Q_UNUSED(input);
00024     /*DO NOTHING*/
00025 }
  
```

4.63.3.11 void SH_InOutState ::setOutput(const QString & output) [virtual], [slot], [inherited]

Réimplémentée dans [SH_DatabaseContentQuestionState](#).

Définition à la ligne 68 du fichier [SH_IOState.cpp](#).

Références [SH_InOutState ::m_isVisible](#), [SH_InOutState ::m_output](#), [SH_InOutState ::output\(\)](#), et [SH_InOutState ::sendOutput\(\)](#).

Référencé par [SH_DatabaseContentQuestionState ::setOutput\(\)](#), et [SH_ServiceCharging ::SH_ServiceCharging\(\)](#).

```
00069 {
00070     m_output = output;
00071     if(m_isVisible) {
00072         emit sendOutput(QVariant(m_output));
00073     }
00074 }
```

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :



4.63.3.12 void SH_InOutState ::setVisibility(bool isVisible) [virtual], [slot], [inherited]

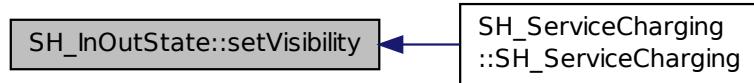
Définition à la ligne 81 du fichier [SH_IOState.cpp](#).

Références [SH_InOutState ::m_isVisible](#).

Référencé par [SH_ServiceCharging ::SH_ServiceCharging\(\)](#).

```
00082 {
00083     m_isVisible = isVisible;
00084 }
```

Voici le graphe des appelants de cette fonction :



4.63.3.13 QString SH_GenericState : :toString() [virtual], [inherited]

Réimplémentée à partir de [SH_NamedObject](#).

Définition à la ligne 21 du fichier [SH_GenericDebugableState.cpp](#).

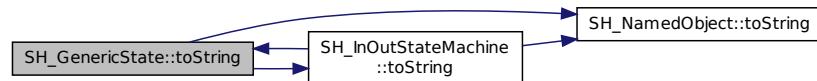
Références [SH_NamedObject : :toString\(\)](#), et [SH_InOutStateMachine : :toString\(\)](#).

Référencé par [SH_InOutStateMachine : :addChildsNextTransition\(\)](#), [SH_DateQuestionState : :rawInput\(\)](#), et [SH_InOutStateMachine : :toString\(\)](#).

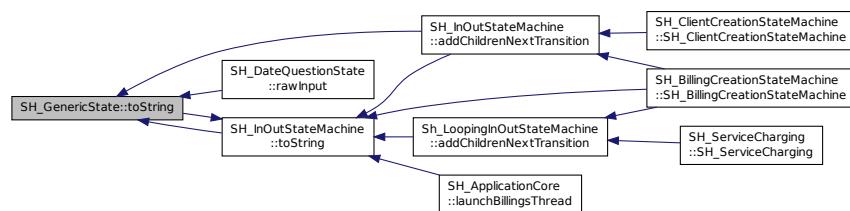
```

00022 {
00023     QStateMachine* machine = this->machine();
00024     SH_InOutStateMachine* mach = qobject_cast<
00025         SH_InOutStateMachine * >(machine);
00026     if(mach) {
00027         return SH_NamedObject::toString() + " [in "+mach->
00028             toString()+"] ";
00029     } else {
00030         return SH_NamedObject::toString();
00031     }
00030 }
  
```

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :



4.63.3.14 bool SH_InOutState ::visibility() [inherited]

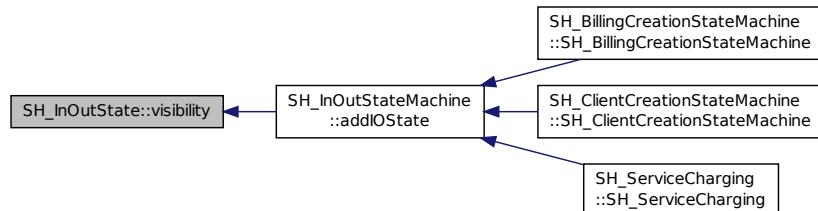
Définition à la ligne 91 du fichier [SH_IOSState.cpp](#).

Références [SH_InOutState ::m_isVisible](#).

Référencé par [SH_InOutStateMachine ::addIOSState\(\)](#).

```
00091     {
00092         return m_isVisible;
00093     }
```

Voici le graphe des appels de cette fonction :



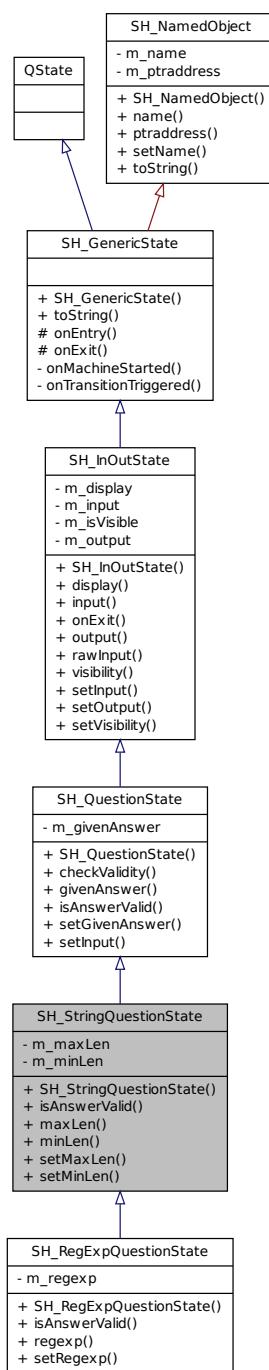
La documentation de cette classe a été générée à partir des fichiers suivants :

- logic/[SH_StatementState.h](#)
- logic/[SH_StatementState.cpp](#)

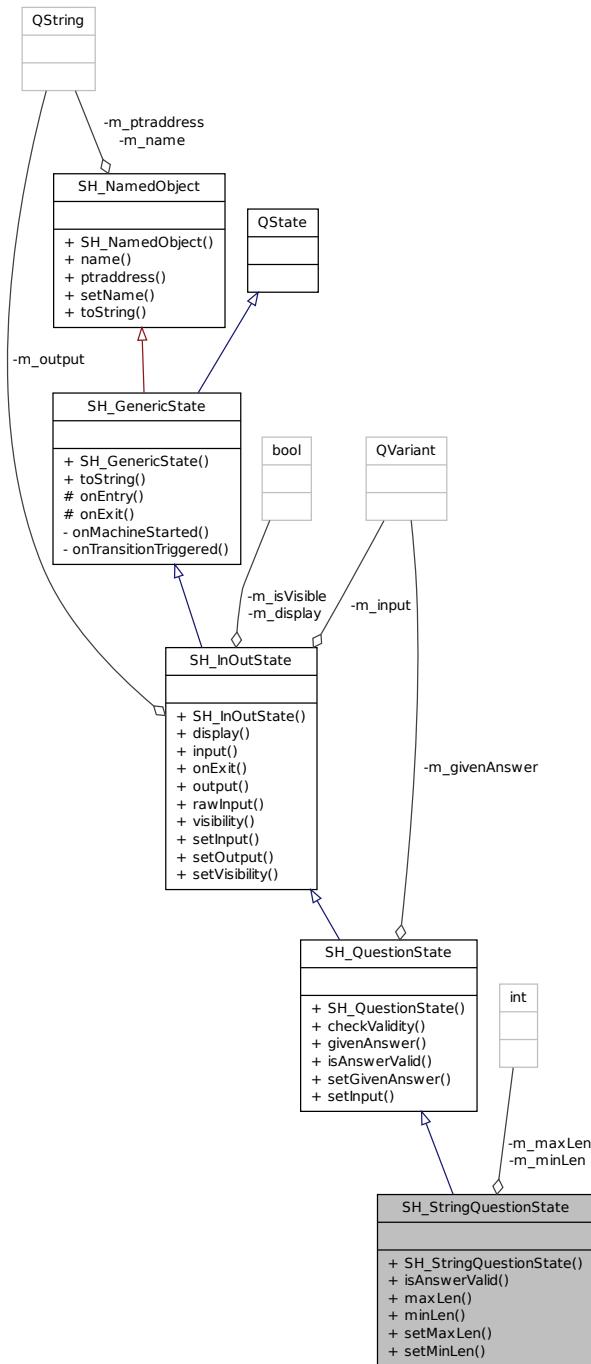
4.64 Référence de la classe SH_StringQuestionState

```
#include <SH_StringQuestionState.h>
```

Graphe d'héritage de SH_StringQuestionState :



Graphe de collaboration de SH_StringQuestionState :



Connecteurs publics

- virtual void `setOutput (const QString &output)`
- virtual void `setVisibility (bool isVisible)`

Signaux

- void `answerInvalid ()`

```

    answerInvalid
- void answerValid ()
    answerValid
- void next ()
- void resendInput (QVariant input)
- void sendOutput (QVariant output)

```

Fonctions membres publiques

```

- SH_StringQuestionState (QString question, QString name, int minLength=0, int maxLength=-1, QState *parent=0)
- bool checkValidity ()
- void display (bool canDisplay)
- virtual QVariant givenAnswer () const
- virtual QVariant input () const
- virtual bool isValid (const QVariant &givenAnswer)
- int maxLen () const
- int minLen () const
- void onExit (QEvent *event)
- virtual QString output () const
- virtual QVariant rawInput () const
- virtual void setGivenAnswer (const QVariant &givenAnswer)
- virtual void setInput (const QVariant &input)
- void setMaxLen (int maxLen)
- void setMinLen (int minLen)
- QString toString ()
- bool visibility ()

```

Fonctions membres protégées

- void **onEntry** (QEvent *event)

Attributs privés

```

- int m_maxLen
    m_maxLen
- int m_minLen
    m_minLen

```

4.64.1 Description détaillée

Définition à la ligne 10 du fichier [SH_StringQuestionState.h](#).

4.64.2 Documentation des constructeurs et destructeur

4.64.2.1 SH_StringQuestionState : :SH_StringQuestionState (QString question, QString name, int minLength = 0, int maxLength = -1, QState * parent = 0)

Paramètres

<i>question</i>	
<i>name</i>	
<i>minLength</i>	
<i>maxLength</i>	
<i>parent</i>	

Définition à la ligne 10 du fichier [SH_StringQuestionState.cpp](#).

```

00010
:
00011      SH_QuestionState(question, name, parent), m_minLen(minLength),
        m_maxLen(maxLength)

```

```
00012 {
00013
00014 }
```

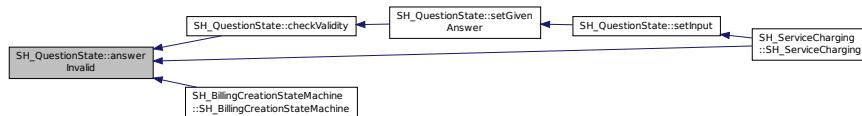
4.64.3 Documentation des fonctions membres

4.64.3.1 void SH_QuestionState ::answerInvalid() [signal], [inherited]

answerInvalid

Référencé par [SH_QuestionState ::checkValidity\(\)](#), [SH_BillingCreationStateMachine ::SH_BillingCreationStateMachine\(\)](#), et [SH_ServiceCharging ::SH_ServiceCharging\(\)](#).

Voici le graphe des appelants de cette fonction :

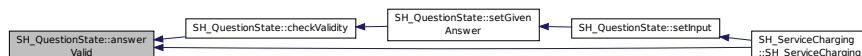


4.64.3.2 void SH_QuestionState ::answerValid() [signal], [inherited]

answerValid

Référencé par [SH_QuestionState ::checkValidity\(\)](#), et [SH_ServiceCharging ::SH_ServiceCharging\(\)](#).

Voici le graphe des appelants de cette fonction :



4.64.3.3 bool SH_QuestionState ::checkValidity() [inherited]

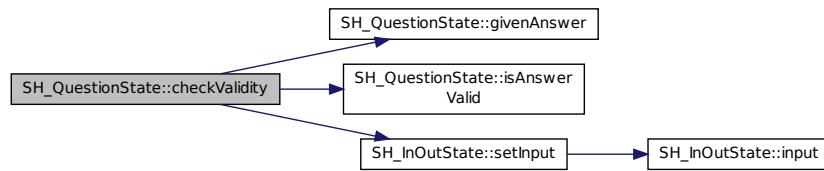
Définition à la ligne 20 du fichier [SH_QuestionState.cpp](#).

Références [SH_QuestionState ::answerInvalid\(\)](#), [SH_QuestionState ::answerValid\(\)](#), [SH_QuestionState ::givenAnswer\(\)](#), [SH_QuestionState ::isValid\(\)](#), [SH_GenericState ::next\(\)](#), et [SH_InOutState ::setInput\(\)](#).

Référencé par [SH_QuestionState ::setGivenAnswer\(\)](#).

```
00021 {
00022     bool ok = this->isValid(this->givenAnswer());
00023     if(ok) {
00024         qDebug() << this->givenAnswer() << " answer valid !";
00025         qDebug() << this->givenAnswer();
00026         SH_InOutState::setInput(this->givenAnswer());
00027         emit answerValid();
00028         emit next();
00029     } else {
00030         qDebug() << this->givenAnswer() << " answer invalid :-(";
00031         qDebug() << this->givenAnswer();
00032         emit answerInvalid();
00033     }
00034     return ok;
00035 }
```

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appels de cette fonction :



4.64.3.4 void SH_InOutState ::display (bool canDisplay) [inherited]

Définition à la ligne 95 du fichier [SH_IOState.cpp](#).

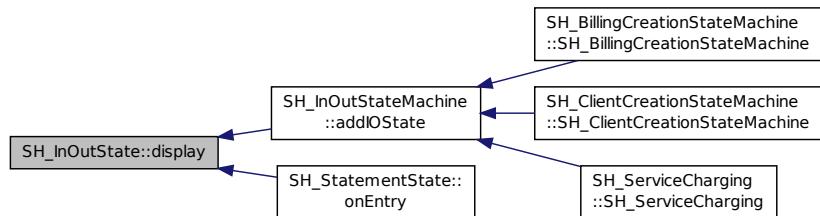
Références [SH_InOutState ::m_display](#), [SH_InOutState ::m_isVisible](#), [SH_InOutState ::m_output](#), et [SH_InOutState ::sendOutput\(\)](#).

Référencé par [SH_InOutStateMachine ::addIOState\(\)](#), et [SH_StatementState ::onEntry\(\)](#).

```

00096 {
00097     m_display=canDisplay;
00098     if(m_display && !m_output.isEmpty() && m_isVisible) {
00099         qDebug() << "resalut !" << QVariant(m_output);
00100         emit sendOutput(QVariant(m_output));
00101     }
00102 }
```

Voici le graphe des appels de cette fonction :



4.64.3.5 QVariant SH_QuestionState ::givenAnswer () const [virtual], [inherited]

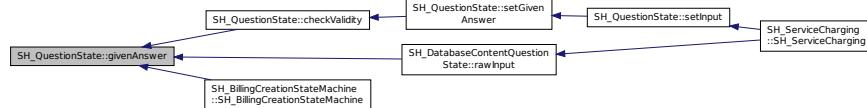
Définition à la ligne 55 du fichier [SH_QuestionState.cpp](#).

Références [SH_QuestionState ::m_givenAnswer](#).

Référencé par [SH_QuestionState ::checkValidity\(\)](#), [SH_DatabaseContentQuestionState ::rawInput\(\)](#), et [SH_BillingCreationStateMachine ::SH_BillingCreationStateMachine\(\)](#).

```
00056 {
00057     return this->m_givenAnswer;
00058 }
```

Voici le graphe des appelants de cette fonction :



4.64.3.6 QVariant SH_InOutState ::input() const [virtual], [inherited]

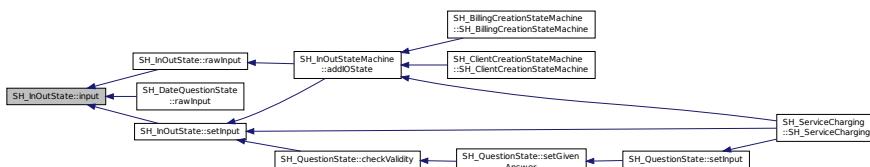
Définition à la ligne 20 du fichier [SH_IOState.cpp](#).

Références [SH_InOutState ::m_input](#).

Référencé par [SH_InOutState ::rawInput\(\)](#), [SH_DateQuestionState ::rawInput\(\)](#), et [SH_InOutState ::setInput\(\)](#).

```
00021 {
00022     return m_input;
00023 }
```

Voici le graphe des appelants de cette fonction :



4.64.3.7 bool SH_StringQuestionState ::isValid(const QVariant & givenAnswer) [virtual]

Implémente [SH_QuestionState](#).

Réimplémentée dans [SH_RegExpQuestionState](#).

Définition à la ligne 22 du fichier [SH_StringQuestionState.cpp](#).

Références [m_maxLen](#), et [m_minLen](#).

```
00023 {
00024     QString answer = givenAnswer.toString();
00025     if(!answer.isEmpty()) {
00026         int answerLength= answer.length();
00027         return ((m_maxLen <= m_minLen || answerLength <=
00028             m_maxLen) && answerLength >= m_minLen);
00029     } else {
00030         return false;
00031     }
  
```

4.64.3.8 int SH_StringQuestionState ::maxLen() const

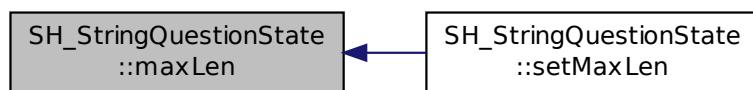
Définition à la ligne 39 du fichier [SH_StringQuestionState.cpp](#).

Références [m_maxLen](#).

Référencé par [setMaxLen\(\)](#).

```
00040 {  
00041     return m_maxLen;  
00042 }
```

Voici le graphe des appelants de cette fonction :



4.64.3.9 int SH_StringQuestionState ::minLen() const

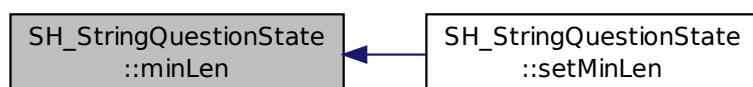
Définition à la ligne 61 du fichier [SH_StringQuestionState.cpp](#).

Références [m_minLen](#).

Référencé par [setMaxLen\(\)](#).

```
00062 {  
00063     return m_minLen;  
00064 }
```

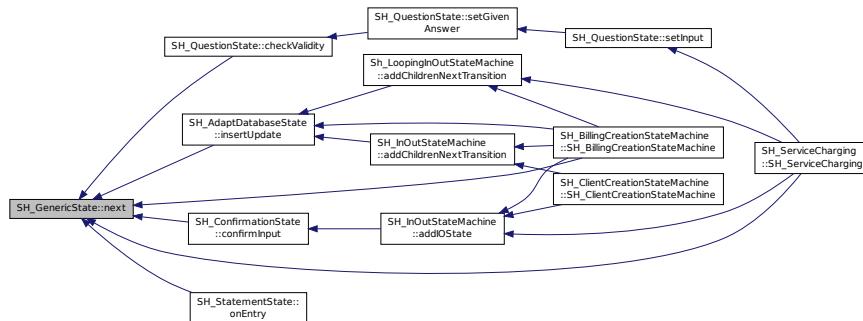
Voici le graphe des appelants de cette fonction :



4.64.3.10 void SH_GenericState ::next() [signal], [inherited]

Référencé par [SH_QuestionState ::checkValidity\(\)](#), [SH_ConfirmationState ::confirmInput\(\)](#), [SH_AdaptDatabaseState ::insertUpdate\(\)](#), [SH_StatementState ::onEntry\(\)](#), [SH_BillingCreationStateMachine ::SH_BillingCreationStateMachine\(\)](#), et [SH_ServiceCharging ::SH_ServiceCharging\(\)](#).

Voici le graphe des appelants de cette fonction :



4.64.3.11 void SH_GenericState ::onEntry (QEvent * event) [protected], [inherited]

Définition à la ligne 62 du fichier [SH_GenericDebugableState.cpp](#).

Références [SH_NamedObject ::name\(\)](#).

Référencé par [SH_StatementState ::onEntry\(\)](#).

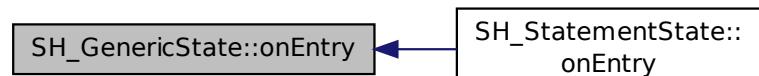
```

00063 {
00064     Q_UNUSED(event);
00065     qDebug() << "Machine: " << machine()->objectName() << " entered " << name();
00066 }
  
```

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :



4.64.3.12 void SH_InOutState ::onExit (QEvent * event) [inherited]

Définition à la ligne 110 du fichier [SH_IOState.cpp](#).

Références [SH_InOutState ::m_input](#), [SH_InOutState ::m_isVisible](#), [SH_GenericState ::onExit\(\)](#), et [SH_InOutState ::resendInput\(\)](#).

```

00111 {
00112     SH_GenericState::onExit(event);
00113     if(m_isVisible) {
00114         emit resendInput(m_input);
00115     }
00116 }
```

Voici le graphe d'appel pour cette fonction :



4.64.3.13 QString SH_InOutState::output() const [virtual], [inherited]

Définition à la ligne 56 du fichier [SH_IOState.cpp](#).

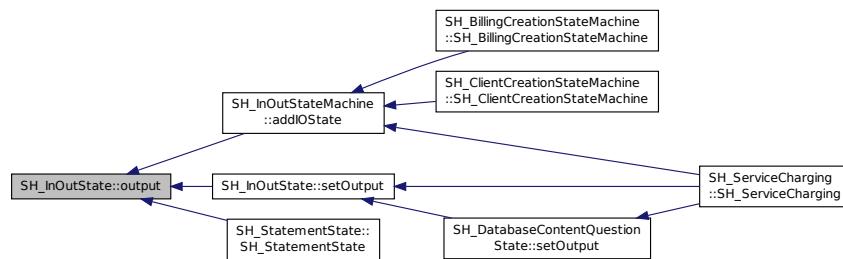
Références [SH_InOutState::m_output](#).

Référencé par [SH_InOutStateMachine::addIOState\(\)](#), [SH_InOutState::setOutput\(\)](#), et [SH_StatementState::SH_StatementState\(\)](#).

```

00057 {
00058     return m_output;
00059 }
```

Voici le graphe des appels de cette fonction :



4.64.3.14 QVariant SH_InOutState::rawInput() const [virtual], [inherited]

Réimplémentée dans [SH_DateQuestionState](#), et [SH_DatabaseContentQuestionState](#).

Définition à la ligne 30 du fichier [SH_IOState.cpp](#).

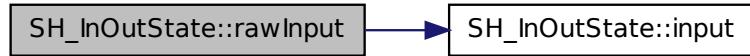
Références [SH_InOutState::input\(\)](#).

Référencé par [SH_InOutStateMachine::addIOState\(\)](#).

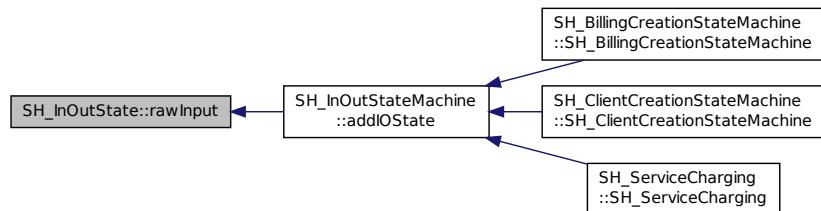
```

00031 {
00032     return input();
00033 }
```

Voici le graphe d'appel pour cette fonction :



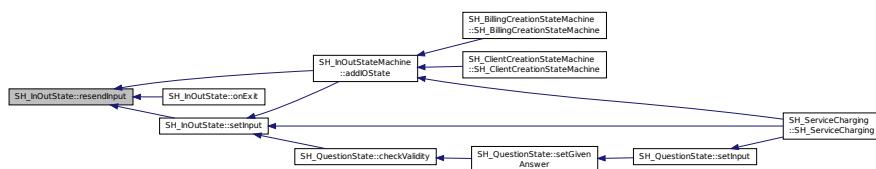
Voici le graphe des appels de cette fonction :



4.64.3.15 void SH_InOutState ::resendInput (QVariant *input*) [signal], [inherited]

Référencé par [SH_InOutStateMachine ::addIOState\(\)](#), [SH_InOutState ::onExit\(\)](#), et [SH_InOutState ::setInput\(\)](#).

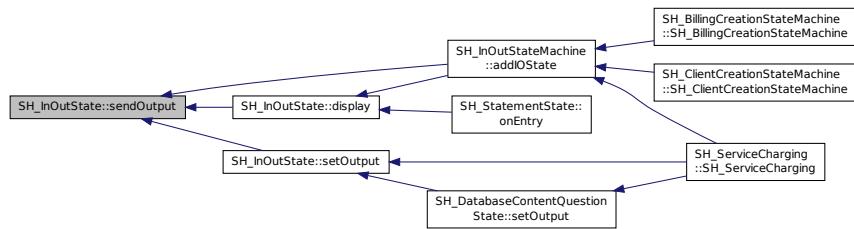
Voici le graphe des appels de cette fonction :



4.64.3.16 void SH_InOutState ::sendOutput (QVariant *output*) [signal], [inherited]

Référencé par [SH_InOutStateMachine ::addIOState\(\)](#), [SH_InOutState ::display\(\)](#), et [SH_InOutState ::setOutput\(\)](#).

Voici le graphe des appelants de cette fonction :



4.64.3.17 void SH_QuestionState ::setGivenAnswer (const QVariant & givenAnswer) [virtual], [inherited]

Définition à la ligne 66 du fichier [SH_QuestionState.cpp](#).

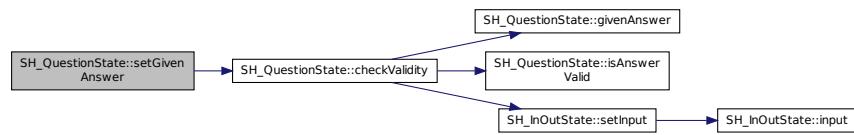
Références [SH_QuestionState ::checkValidity\(\)](#), et [SH_QuestionState ::m_givenAnswer](#).

Référencé par [SH_QuestionState ::setInput\(\)](#).

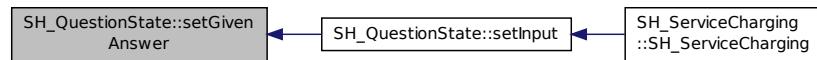
```

00067 {
00068     this->m_givenAnswer = givenAsnwer;
00069     this->checkValidity();
00070 }
  
```

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :



4.64.3.18 void SH_QuestionState ::setInput (const QVariant & input) [virtual], [inherited]

Réimplémentée à partir de [SH_InOutState](#).

Définition à la ligne 43 du fichier [SH_QuestionState.cpp](#).

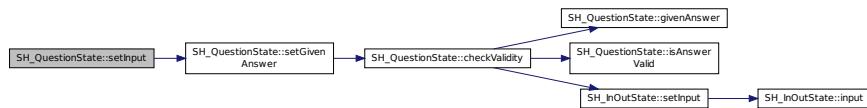
Références [SH_QuestionState ::setGivenAnswer\(\)](#).

Référencé par [SH_ServiceCharging ::SH_ServiceCharging\(\)](#).

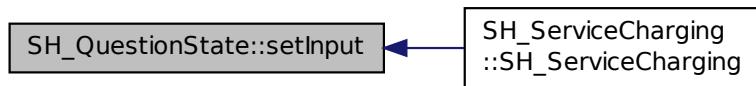
```

00044 {
00045     qDebug() << "new answer " << input.toString();
00046     this->setGivenAnswer(input);
00047 }
  
```

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appels de cette fonction :



4.64.3.19 void SH_StringQuestionState ::setMaxLen (int maxLen)

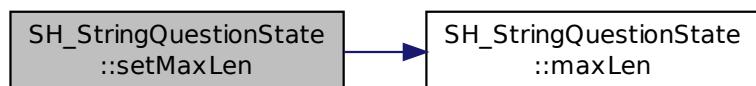
Définition à la ligne 50 du fichier [SH_StringQuestionState.cpp](#).

Références [m_maxLen](#), et [maxLen\(\)](#).

```

00051 {
00052     m_maxLen = maxLen;
00053 }
```

Voici le graphe d'appel pour cette fonction :



4.64.3.20 void SH_StringQuestionState ::setMinLen (int minLen)

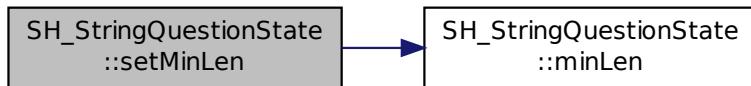
Définition à la ligne 72 du fichier [SH_StringQuestionState.cpp](#).

Références [m_minLen](#), et [minLen\(\)](#).

```

00073 {
00074     m_minLen = minLen;
00075 }
```

Voici le graphe d'appel pour cette fonction :



4.64.3.21 void SH_InOutState ::setOutput(const QString & output) [virtual], [slot], [inherited]

Réimplémentée dans [SH_DatabaseContentQuestionState](#).

Définition à la ligne 68 du fichier [SH_IOState.cpp](#).

Références [SH_InOutState ::m_isVisible](#), [SH_InOutState ::m_output](#), [SH_InOutState ::output\(\)](#), et [SH_InOutState ::sendOutput\(\)](#).

Référencé par [SH_DatabaseContentQuestionState ::setOutput\(\)](#), et [SH_ServiceCharging ::SH_ServiceCharging\(\)](#).

```

00069 {
00070     m_output = output;
00071     if(m_isVisible) {
00072         emit sendOutput(QVariant(m_output));
00073     }
00074 }
```

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :



4.64.3.22 void SH_InOutState ::setVisibility(bool isVisible) [virtual], [slot], [inherited]

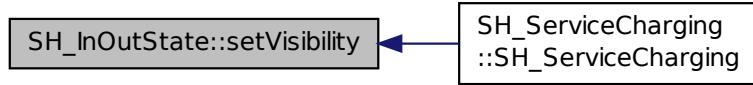
Définition à la ligne 81 du fichier [SH_IOState.cpp](#).

Références [SH_InOutState ::m_isVisible](#).

Référencé par [SH_ServiceCharging ::SH_ServiceCharging\(\)](#).

```
00082 {
00083     m_isVisible = isVisible;
00084 }
```

Voici le graphe des appels de cette fonction :



4.64.3.23 QString SH_GenericState : toString() [virtual], [inherited]

Réimplémentée à partir de [SH_NamedObject](#).

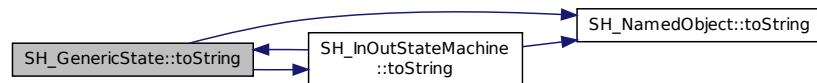
Définition à la ligne 21 du fichier [SH_GenericDebugableState.cpp](#).

Références [SH_NamedObject : toString\(\)](#), et [SH_InOutStateMachine : toString\(\)](#).

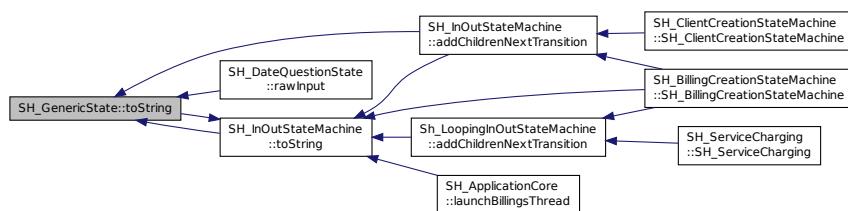
Référencé par [SH_InOutStateMachine : addChildrenNextTransition\(\)](#), [SH_DateQuestionState : rawInput\(\)](#), et [SH_InOutStateMachine : toString\(\)](#).

```
00022 {
00023     QStateMachine* machine = this->machine();
00024     SH_InOutStateMachine* mach = qobject_cast<
00025         SH_InOutStateMachine *>(machine);
00026     if(mach) {
00027         return SH_NamedObject::toString() + " [in "+mach->
00028             toString()+"] ";
00029     } else {
00030         return SH_NamedObject::toString();
00031     }
00030 }
```

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appels de cette fonction :



4.64.3.24 bool SH_InOutState::visibility() [inherited]

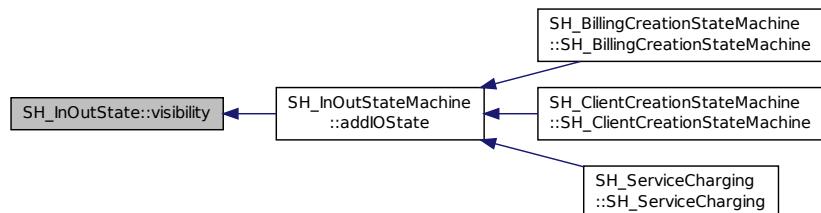
Définition à la ligne 91 du fichier [SH_IOState.cpp](#).

Références [SH_InOutState::m_isVisible](#).

Référencé par [SH_InOutStateMachine::addIOState\(\)](#).

```
00091     {
00092         return m_isVisible;
00093     }
```

Voici le graphe des appels de cette fonction :



4.64.4 Documentation des données membres

4.64.4.1 int SH_StringQuestionState::m_maxLen [private]

m_maxLen

Définition à la ligne 75 du fichier [SH_StringQuestionState.h](#).

Référencé par [isAnswerValid\(\)](#), [maxLen\(\)](#), et [setMaxLen\(\)](#).

4.64.4.2 int SH_StringQuestionState::m_minLen [private]

m_minLen

Définition à la ligne 71 du fichier [SH_StringQuestionState.h](#).

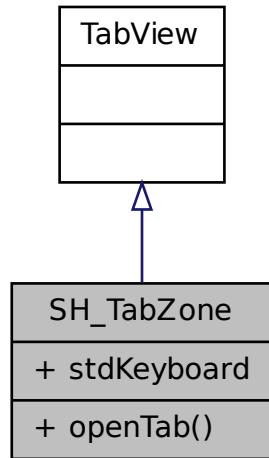
Référencé par [isAnswerValid\(\)](#), [minLen\(\)](#), et [setMinLen\(\)](#).

La documentation de cette classe a été générée à partir des fichiers suivants :

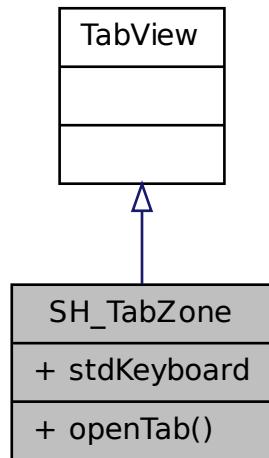
- logic/[SH_StringQuestionState.h](#)
- logic/[SH_StringQuestionState.cpp](#)

4.65 Référence de la classe SH_TabZone

Graphe d'héritage de SH_TabZone :



Graphe de collaboration de SH_TabZone :



Signaux

- void `newBilling ()`
- void `newBooking ()`
- void `newSelling ()`
- void `reload ()`

- void `selected` (string `selectedItem`)
- void `selectedForDetail` (var `data`)

Fonctions membres publiques

- void `openTab` (tabIndex)

Propriétés

- var `stdKeyboard`

4.65.1 Description détaillée

Définition à la ligne 4 du fichier [SH_TabZone.qml](#).

4.65.2 Documentation des fonctions membres

4.65.2.1 void `SH_TabZone::newBilling()` [signal]

4.65.2.2 void `SH_TabZone::newBooking()` [signal]

4.65.2.3 void `SH_TabZone::newSelling()` [signal]

4.65.2.4 void `SH_TabZone::openTab(tabIndex)`

4.65.2.5 void `SH_TabZone::reload()` [signal]

4.65.2.6 void `SH_TabZone::selected(string selectedItem)` [signal]

4.65.2.7 void `SH_TabZone::selectedForDetail(var data)` [signal]

4.65.3 Documentation des propriétés

4.65.3.1 var `SH_TabZone::stdKeyboard`

Définition à la ligne 7 du fichier [SH_TabZone.qml](#).

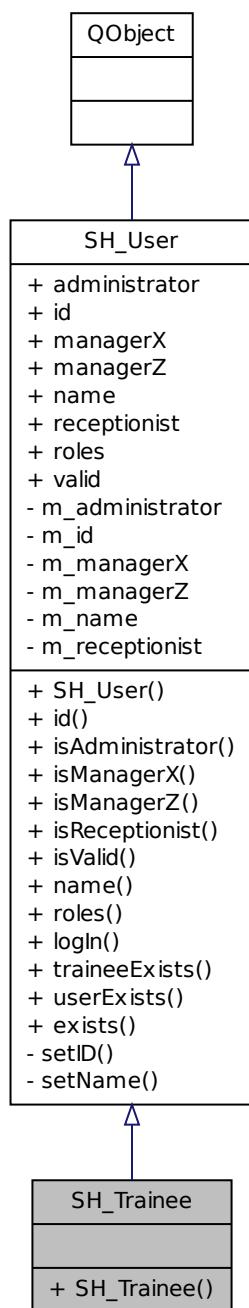
La documentation de cette classe a été générée à partir du fichier suivant :

- [views/qml/SH_TabZone.qml](#)

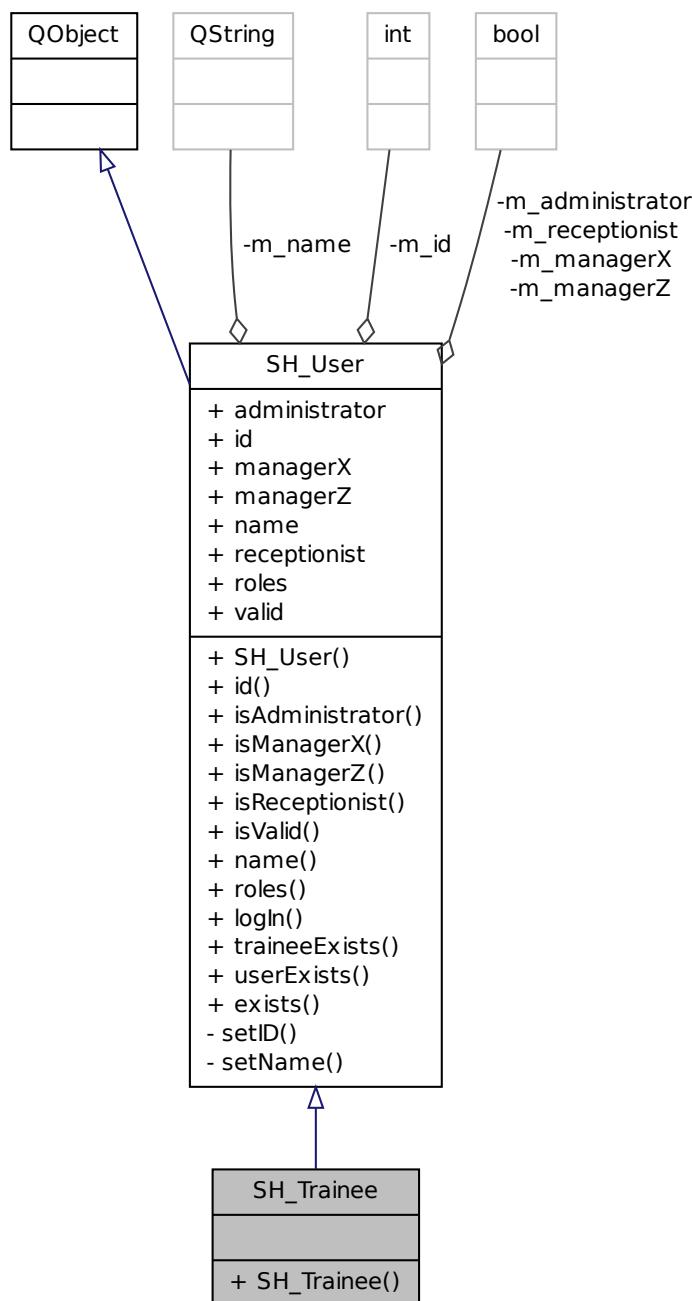
4.66 Référence de la classe SH_Trainee

```
#include <SH_Trainee.h>
```

Graphe d'héritage de SH_Trainee :



Graphe de collaboration de SH_Trainee :



Connecteurs publics

- static QVariant `exists` (QVariant login)

Signaux

- void `nameChanged` ()
- void `rolesChanged` ()

- void `validityChanged ()`

Fonctions membres publiques

- `SH_Trainee (QString name, int id, QObject *parent=0)`
- int `id () const`
- bool `isAdministrator () const`
- bool `isManagerX () const`
- bool `isManagerZ () const`
- bool `isReceptionist () const`
- bool `isValid () const`
- QString `name () const`
- int `roles () const`

Fonctions membres publiques statiques

- static `SH_User * login (QString login, QString pass)`
- static bool `traineeExists (QString login)`
- static bool `userExists (QString login)`

Propriétés

- bool `administrator`
- int `id`
- bool `managerX`
- bool `managerZ`
- QString `name`
- bool `receptionist`
- int `roles`
- bool `valid`

4.66.1 Description détaillée

Définition à la ligne 14 du fichier `SH_Trainee.h`.

4.66.2 Documentation des constructeurs et destructeur

4.66.2.1 `SH_Trainee : :SH_Trainee (QString name, int id, QObject * parent = 0)`

Paramètres

<code>name</code>	
<code>id</code>	
<code>parent</code>	

Définition à la ligne 9 du fichier `SH_Trainee.cpp`.

```
00009 : SH_User(
00010     name, id, true, false, false, parent)
00011 }
00012 }
```

4.66.3 Documentation des fonctions membres

4.66.3.1 `static QVariant SH_User : :exists (QVariant login) [inline], [static], [slot], [inherited]`

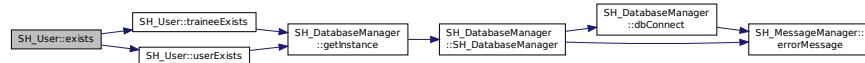
Définition à la ligne 132 du fichier `SH_User.h`.

Références `SH_User : :traineeExists()`, et `SH_User : :userExists()`.

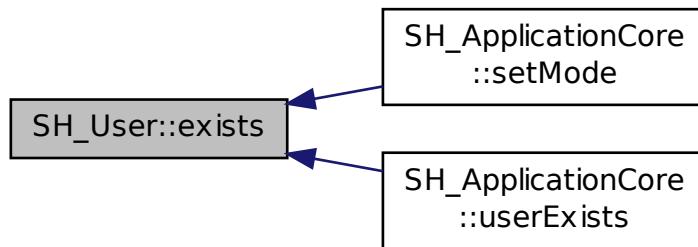
Référencé par `SH_ApplicationCore : :setMode()`, et `SH_ApplicationCore : :userExists()`.

```
00132 {return QVariant(SH_User::userExists(login.toString()) ||
SH_User::traineeExists(login.toString()));}
```

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :



4.66.3.2 int SH_User::id() const [inline], [inherited]

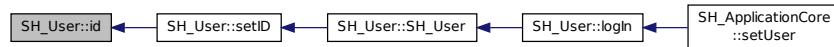
Définition à la ligne 54 du fichier [SH_User.h](#).

Références [SH_User::m_id](#).

Référencé par [SH_User::setId\(\)](#).

```
00054 { return this->m_id; }
```

Voici le graphe des appelants de cette fonction :



4.66.3.3 bool SH_User::isAdministrator() const [inline], [inherited]

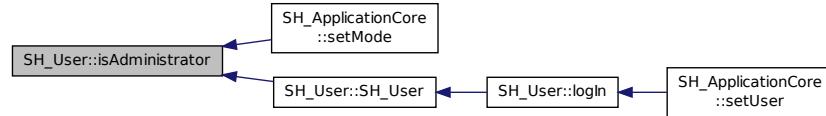
Définition à la ligne 82 du fichier [SH_User.h](#).

Références [SH_User::m_administrator](#).

Référencé par [SH_ApplicationCore::setMode\(\)](#), et [SH_User::SH_User\(\)](#).

```
00082 { return this->m_administrator; }
```

Voici le graphe des appelants de cette fonction :



4.66.3.4 bool SH_User::isManagerX() const [inline], [inherited]

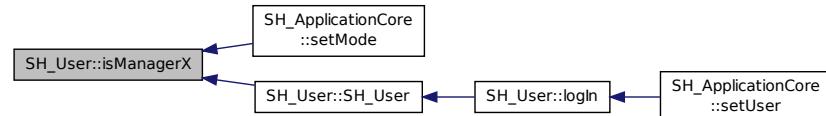
Définition à la ligne 68 du fichier [SH_User.h](#).

Références [SH_User::m_managerX](#).

Référencé par [SH_ApplicationCore::setMode\(\)](#), et [SH_User::SH_User\(\)](#).

```
00068 { return this->m_managerX; }
```

Voici le graphe des appelants de cette fonction :



4.66.3.5 bool SH_User::isManagerZ() const [inline], [inherited]

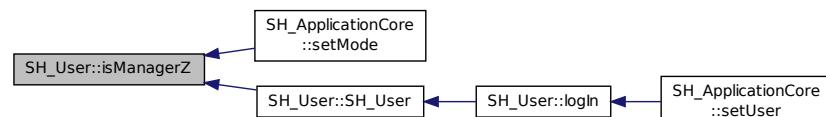
Définition à la ligne 75 du fichier [SH_User.h](#).

Références [SH_User::m_managerZ](#).

Référencé par [SH_ApplicationCore::setMode\(\)](#), et [SH_User::SH_User\(\)](#).

```
00075 { return this->m_managerZ; }
```

Voici le graphe des appelants de cette fonction :



4.66.3.6 bool SH_User::isReceptionist() const [inherited]

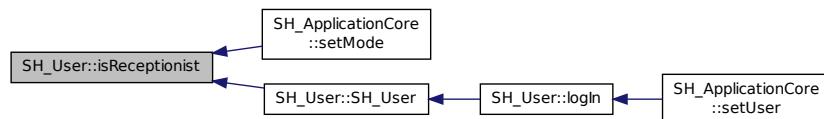
Définition à la ligne 64 du fichier [SH_User.cpp](#).

Références [SH_User::m_receptionist](#).

Référencé par [SH_ApplicationCore::setMode\(\)](#), et [SH_User::SH_User\(\)](#).

```
00065 {
00066     return this->m_receptionist;
00067 }
```

Voici le graphe des appelants de cette fonction :



4.66.3.7 bool SH_User::isValid() const [inherited]

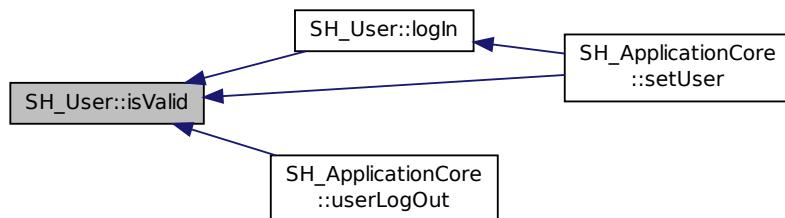
Définition à la ligne 32 du fichier [SH_User.cpp](#).

Références [SH_User::m_id](#), et [SH_User::m_name](#).

Référencé par [SH_User::login\(\)](#), [SH_ApplicationCore::setUser\(\)](#), et [SH_ApplicationCore::userLogOut\(\)](#).

```
00032 {
00033     return (!this->m_name.isEmpty()) && (this->m_id != 0);
00034 }
```

Voici le graphe des appelants de cette fonction :



4.66.3.8 SH_User* SH_User::login(QString login, QString pass) [static], [inherited]

Définition à la ligne 132 du fichier [SH_User.cpp](#).

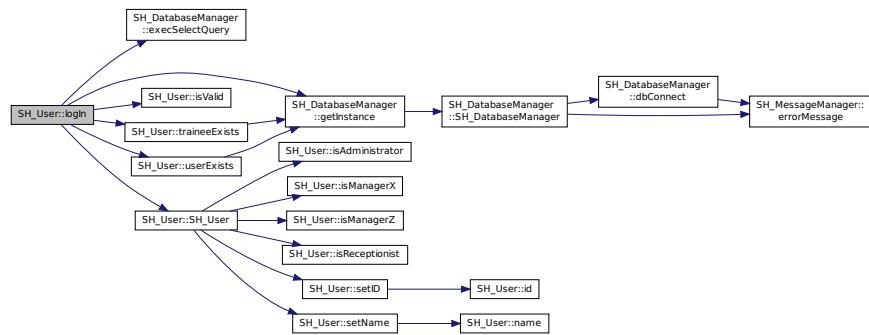
Références [SH_DatabaseManager::execSelectQuery\(\)](#), [SH_DatabaseManager::getInstance\(\)](#), [SH_User::isValid\(\)](#), [SH_User::SH_User\(\)](#), [SH_User::traineeExists\(\)](#), et [SH_User::userExists\(\)](#).

Référencé par [SH_ApplicationCore::setUser\(\)](#).

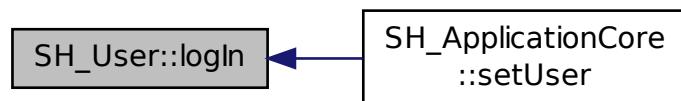
```

00133 {
00134     qDebug() << "log in";
00135     bool isValid = false;
00136     QCryptographicHash encPass(QCryptographicHash::Sha512);
00137     encPass.addData(pass.toUtf8());
00138     bool trainee=false;
00139     QStringList fields;
00140     QString table;
00141     if(userExists(login)) {
00142         fields << "ID" << "LOGIN" << "ISRECEPTIONIST" << "ISMANAGERX" << "ISMANAGERZ" << "ISADMINISTRATOR";
00143         table ="USERS";
00144     } else if(traineeExists(login)) {
00145         fields << "ID" << "LOGIN";
00146         table ="TRAINEES";
00147         trainee=true;
00148     }
00149     QSqlQuery result = SH_DatabaseManager::getInstance () ->
00150     execSelectQuery(table,fields,"LOGIN='"+login+"' AND ENCRYPTEDPASS='"+QString::fromLatin1(
00151     encPass.result().toHex()).toUpper()+"'");
00152     if(result.next()) {
00153         QSqlRecord rec = result.record();
00154         if(rec.isEmpty() || !result.isValid()) {
00155             isValid = false;
00156         } else {
00157             isValid = (rec.value(rec.indexOf("LOGIN")).toString() == login);
00158         }
00159         if(isValid) {
00160             if(trainee) {
00161                 return new SH_Trainee(rec.value(rec.indexOf("LOGIN")).toString(),rec.value(rec.
00162                 indexOf("ID")).toInt());
00163             } else {
00164                 return new SH_User(rec.value(rec.indexOf("LOGIN")).toString(),rec.value(rec.indexOf(
00165                 "ID")).toInt(),(rec.value(rec.indexOf("ISRECEPTIONIST")).toString()=="1"),(rec.value(rec.indexOf("ISMANAGERX
00166                 ").toString()=="1"),(rec.value(rec.indexOf("ISMANAGERZ")).toString()=="1"),(rec.value(rec.indexOf("ISADMINISTRATOR")).toString()=="1"));
00167             }
00168         }
00169     }
00170     return new SH_User();
00171 }
```

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appels de cette fonction :



4.66.3.9 `QString SH_User::name() const [inherited]`

Référencé par [SH_User::setName\(\)](#).

Voici le graphe des appelants de cette fonction :



4.66.3.10 `void SH_User::nameChanged() [signal], [inherited]`

4.66.3.11 `int SH_User::roles() const [inherited]`

4.66.3.12 `void SH_User::rolesChanged() [signal], [inherited]`

4.66.3.13 `bool SH_User::traineeExists(QString login) [static], [inherited]`

Définition à la ligne 121 du fichier [SH_User.cpp](#).

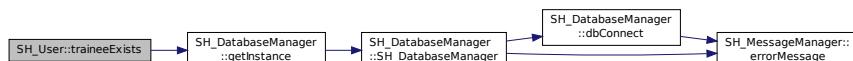
Références [SH_DatabaseManager::getInstance\(\)](#).

Référencé par [SH_User::exists\(\)](#), et [SH_User::login\(\)](#).

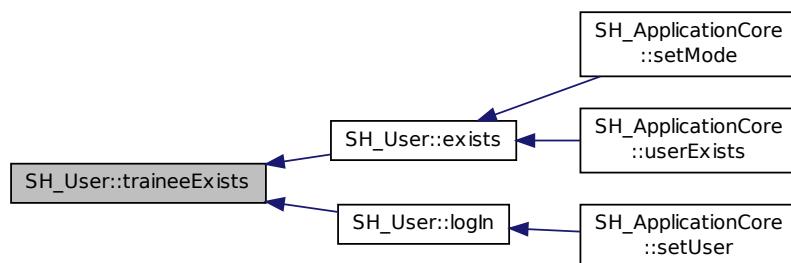
```

00121             {
00122     qDebug() << "trainee exists";
00123     return (SH_DatabaseManager::getInstance() ->dataCount("TRAINees", "
00124         LOGIN='"+login+"'") == 1);
  
```

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :



4.66.3.14 bool SH_User::userExists (QString login) [static], [inherited]

Définition à la ligne 110 du fichier [SH_User.cpp](#).

Références [SH_DatabaseManager::getInstance\(\)](#).

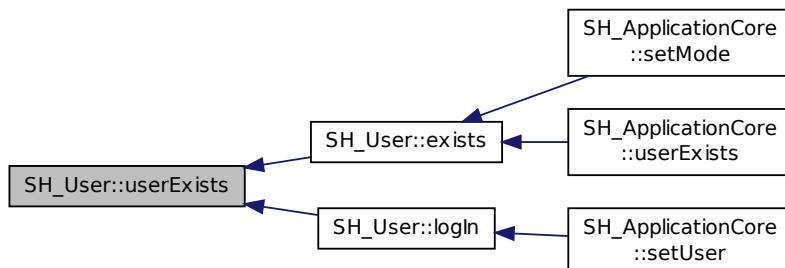
Référencé par [SH_User::exists\(\)](#), et [SH_User::login\(\)](#).

```
00110     qDebug() << "user exists";
00111     return (SH_DatabaseManager::getInstance() ->dataCount ("USERS", "LOGIN=' "+
00112         login+"'"') == 1);
00113 }
```

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :



4.66.3.15 void SH_User::validityChanged () [signal], [inherited]

4.66.4 Documentation des propriétés

4.66.4.1 bool SH_User::administrator [read], [inherited]

Définition à la ligne 23 du fichier [SH_User.h](#).

4.66.4.2 int SH_User::id [read], [inherited]

Définition à la ligne 18 du fichier [SH_User.h](#).

Référencé par [SH_ApplicationCore::launchBillThread\(\)](#).

4.66.4.3 bool SH_User::managerX [read], [inherited]

Définition à la ligne 21 du fichier [SH_User.h](#).

4.66.4.4 `bool SH_User::managerZ [read], [inherited]`

Définition à la ligne 22 du fichier [SH_User.h](#).

4.66.4.5 `QString SH_User::name [read], [inherited]`

Définition à la ligne 19 du fichier [SH_User.h](#).

Référencé par [SH_ApplicationCore::setMode\(\)](#), et [SH_ApplicationCore::setUser\(\)](#).

4.66.4.6 `bool SH_User::receptionist [read], [inherited]`

Définition à la ligne 20 du fichier [SH_User.h](#).

4.66.4.7 `int SH_User::roles [read], [inherited]`

Définition à la ligne 24 du fichier [SH_User.h](#).

4.66.4.8 `bool SH_User::valid [read], [inherited]`

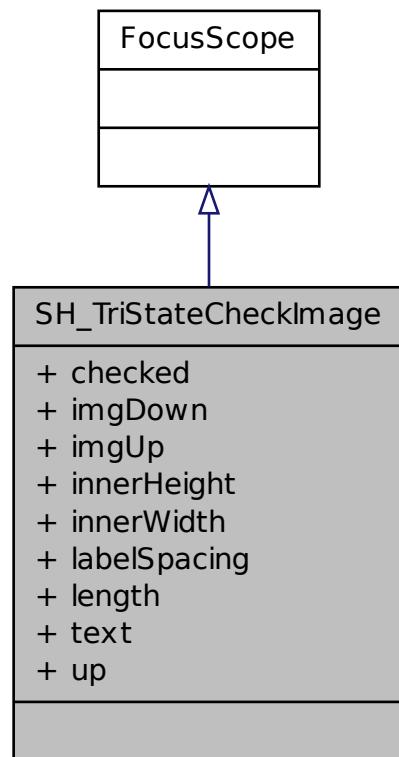
Définition à la ligne 25 du fichier [SH_User.h](#).

La documentation de cette classe a été générée à partir des fichiers suivants :

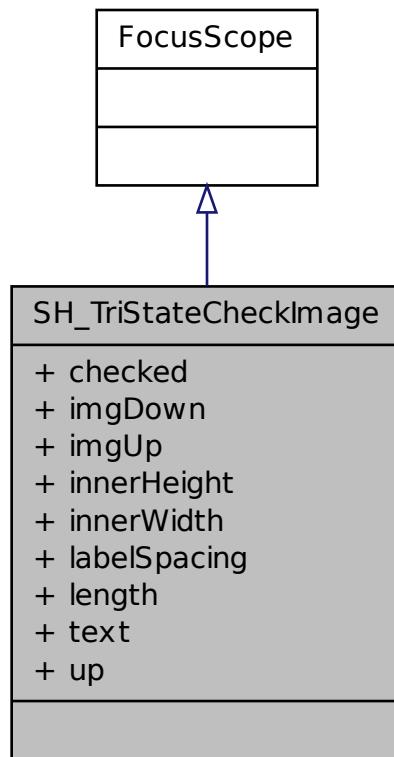
- [models/SH_Trainee.h](#)
- [models/SH_Trainee.cpp](#)

4.67 Référence de la classe SH_TriStateCheckImage

Graphe d'héritage de SH_TriStateCheckImage :



Graphe de collaboration de SH_TriStateCheckImage :



Signaux

- void `pressed ()`

Propriétés

- bool `checked`
- string `imgDown`
- string `imgUp`
- real `innerHeight`
- real `innerWidth`
- real `labelSpacing`
- real `length`
- alias `text`
- bool `up`

4.67.1 Description détaillée

Définition à la ligne 4 du fichier [SH_TriStateCheckImage.qml](#).

4.67.2 Documentation des fonctions membres

4.67.2.1 void SH_TriStateCheckImage::pressed () [signal]

4.67.3 Documentation des propriétés

4.67.3.1 `bool SH_TriStateCheckImage ::checked`

Définition à la ligne 7 du fichier [SH_TriStateCheckImage.qml](#).

4.67.3.2 `string SH_TriStateCheckImage ::imgDown`

Définition à la ligne 17 du fichier [SH_TriStateCheckImage.qml](#).

4.67.3.3 `string SH_TriStateCheckImage ::imgUp`

Définition à la ligne 15 du fichier [SH_TriStateCheckImage.qml](#).

4.67.3.4 `real SH_TriStateCheckImage ::innerHeight`

Définition à la ligne 21 du fichier [SH_TriStateCheckImage.qml](#).

4.67.3.5 `real SH_TriStateCheckImage ::innerWidth`

Définition à la ligne 23 du fichier [SH_TriStateCheckImage.qml](#).

4.67.3.6 `real SH_TriStateCheckImage ::labelSpacing`

Définition à la ligne 11 du fichier [SH_TriStateCheckImage.qml](#).

4.67.3.7 `real SH_TriStateCheckImage ::length`

Définition à la ligne 19 du fichier [SH_TriStateCheckImage.qml](#).

4.67.3.8 `alias SH_TriStateCheckImage ::text`

Définition à la ligne 13 du fichier [SH_TriStateCheckImage.qml](#).

4.67.3.9 `bool SH_TriStateCheckImage ::up`

Définition à la ligne 9 du fichier [SH_TriStateCheckImage.qml](#).

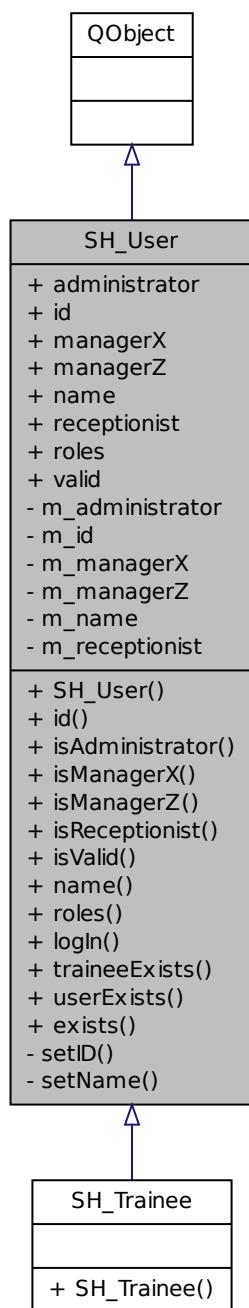
La documentation de cette classe a été générée à partir du fichier suivant :

– [views/qml/SHTriStateCheckImage.qml](#)

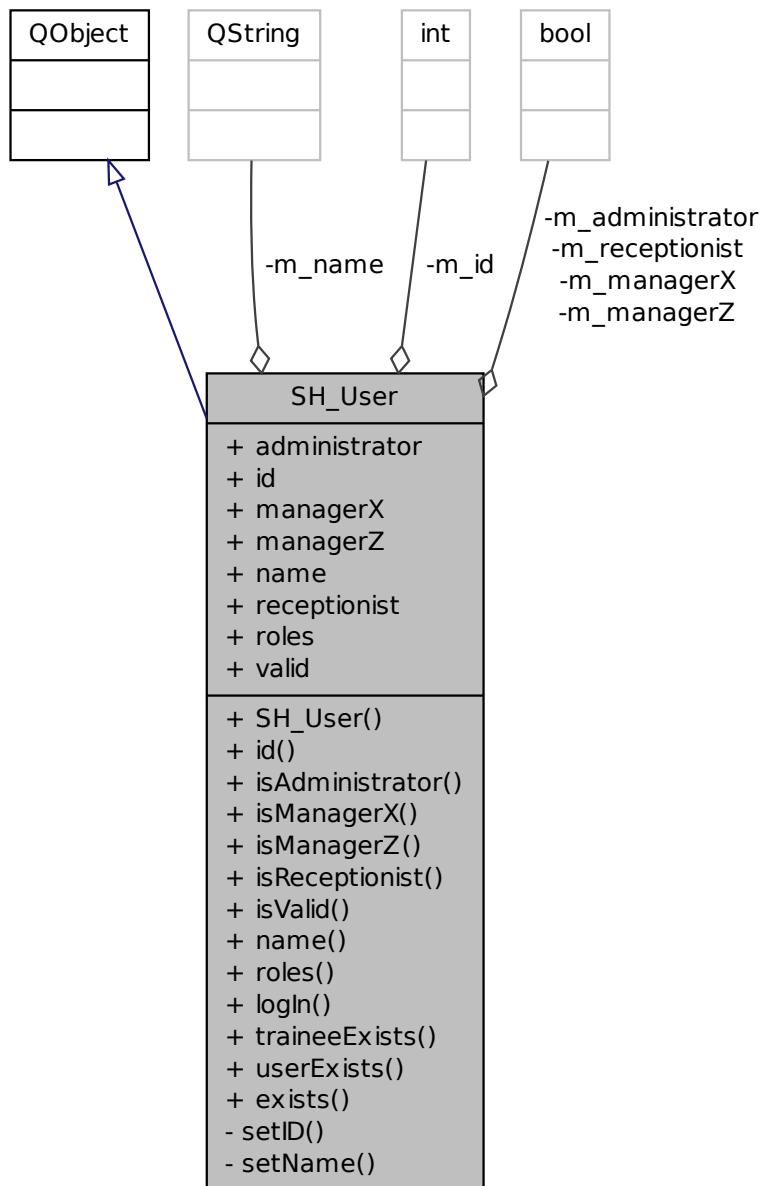
4.68 Référence de la classe SH_User

```
#include <SH_User.h>
```

Graphe d'héritage de SH_User :



Graphe de collaboration de SH_User :



Connecteurs publics

- static QVariant [exists](#) (QVariant login)

Signaux

- void [nameChanged](#) ()
- void [rolesChanged](#) ()
- void [validityChanged](#) ()

Fonctions membres publiques

- `SH_User (QString name="", int id=0, bool isReceptionist=false, bool isManagerX=false, bool isManagerZ=false, bool isAdministrator=false, QObject *parent=0)`
- `int id () const`
- `bool isAdministrator () const`
- `bool isManagerX () const`
- `bool isManagerZ () const`
- `bool isReceptionist () const`
- `bool isValid () const`
- `QString name () const`
- `int roles () const`

Fonctions membres publiques statiques

- `static SH_User * logIn (QString login, QString pass)`
- `static bool traineeExists (QString login)`
- `static bool userExists (QString login)`

Propriétés

- `bool administrator`
- `int id`
- `bool managerX`
- `bool managerZ`
- `QString name`
- `bool receptionist`
- `int roles`
- `bool valid`

Fonctions membres privées

- `void setId (int id)`
- `void setName (QString name)`

Attributs privés

- `bool m_administrator
 $m_administrator$`
- `int m_id
 m_id`
- `bool m_managerX
 $m_managerX$`
- `bool m_managerZ
 $m_managerZ$`
- `QString m_name
 m_name`
- `bool m_receptionist
 $m_receptionist$`

4.68.1 Description détaillée

Définition à la ligne 15 du fichier `SH_User.h`.

4.68.2 Documentation des constructeurs et destructeur

- 4.68.2.1 `SH_User ::SH_User (QString name = " ", int id = 0, bool isReceptionist = false, bool isManagerX = false, bool isManagerZ = false, bool isAdministrator = false, QObject * parent = 0)`

Paramètres

<i>name</i>	
<i>id</i>	
<i>isReceptionist</i>	
<i>isManagerX</i>	
<i>isManagerZ</i>	
<i>isAdministrator</i>	
<i>parent</i>	

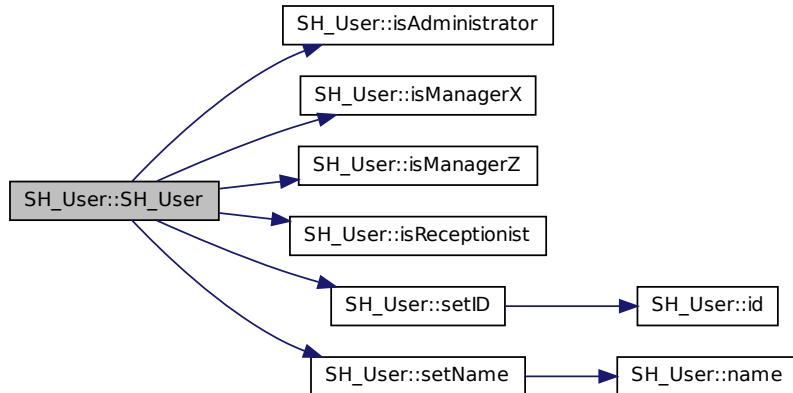
Définition à la ligne 15 du fichier [SH_User.cpp](#).

Références [isAdministrator\(\)](#), [isManagerX\(\)](#), [isManagerZ\(\)](#), [isReceptionist\(\)](#), [m_administrator](#), [m_managerX](#), [m_managerZ](#), [m_receptionist](#), [setID\(\)](#), et [setName\(\)](#).

Référencé par [login\(\)](#).

```
00016     : QObject (parent)
00017 {
00018     this->setName (name);
00019     this->setID (id);
00020     this->m_receptionist = isReceptionist;
00021     this->m_managerX = isManagerX;
00022     this->m_managerZ = isManagerZ;
00023     this->m_administrator = isAdministrator;
00024 }
```

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appels de cette fonction :



4.68.3 Documentation des fonctions membres

4.68.3.1 static QVariant SH_User::exists (QVariant *login*) [inline], [static], [slot]

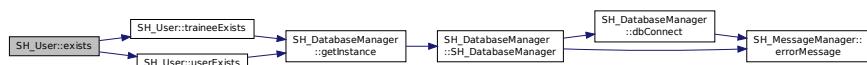
Définition à la ligne 132 du fichier [SH_User.h](#).

Références [traineeExists\(\)](#), et [userExists\(\)](#).

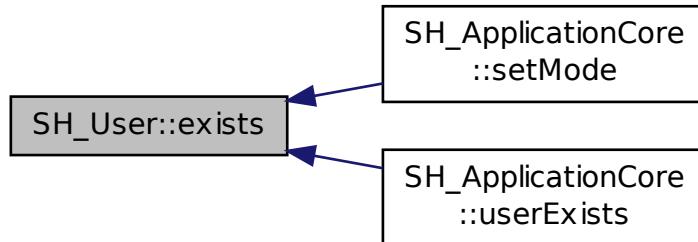
Référencé par [SH_ApplicationCore::setMode\(\)](#), et [SH_ApplicationCore::userExists\(\)](#).

```
00132 {return QVariant(SH_User::userExists(login.toString()) ||  
SH_User::traineeExists(login.toString()));}
```

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appels de cette fonction :



4.68.3.2 int SH_User::id () const [inline]

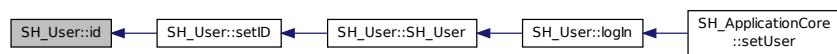
Définition à la ligne 54 du fichier [SH_User.h](#).

Références [m_id](#).

Référencé par [setID\(\)](#).

```
00054 { return this->m_id; }
```

Voici le graphe des appels de cette fonction :



4.68.3.3 bool SH_User::isAdministrator() const [inline]

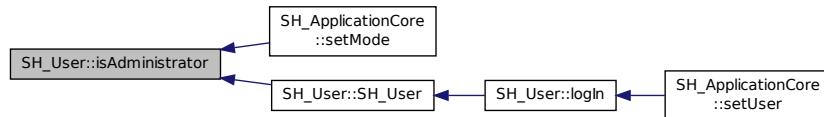
Définition à la ligne 82 du fichier [SH_User.h](#).

Références [m_administrator](#).

Référencé par [SH_ApplicationCore::setMode\(\)](#), et [SH_User\(\)](#).

```
00082 { return this->m_administrator; }
```

Voici le graphe des appelants de cette fonction :



4.68.3.4 bool SH_User::isManagerX() const [inline]

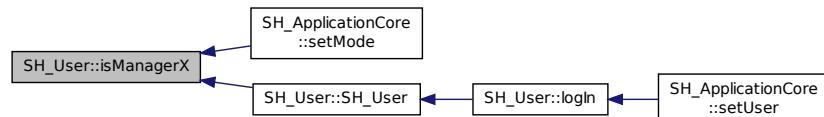
Définition à la ligne 68 du fichier [SH_User.h](#).

Références [m_managerX](#).

Référencé par [SH_ApplicationCore::setMode\(\)](#), et [SH_User\(\)](#).

```
00068 { return this->m_managerX; }
```

Voici le graphe des appelants de cette fonction :



4.68.3.5 bool SH_User::isManagerZ() const [inline]

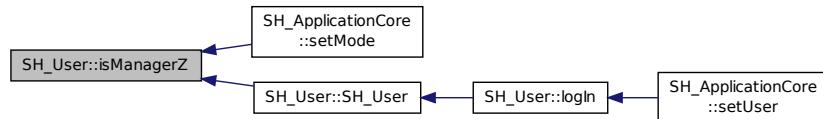
Définition à la ligne 75 du fichier [SH_User.h](#).

Références [m_managerZ](#).

Référencé par [SH_ApplicationCore::setMode\(\)](#), et [SH_User\(\)](#).

```
00075 { return this->m_managerZ; }
```

Voici le graphe des appels de cette fonction :



4.68.3.6 bool SH_User::isReceptionist() const

Définition à la ligne 64 du fichier [SH_User.cpp](#).

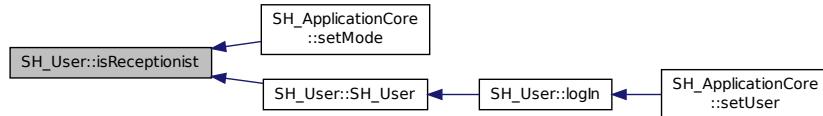
Références [m_receptionist](#).

Référencé par [SH_ApplicationCore::setMode\(\)](#), et [SH_User\(\)](#).

```

00065 {
00066     return this->m_receptionist;
00067 }
```

Voici le graphe des appels de cette fonction :



4.68.3.7 bool SH_User::isValid() const

Définition à la ligne 32 du fichier [SH_User.cpp](#).

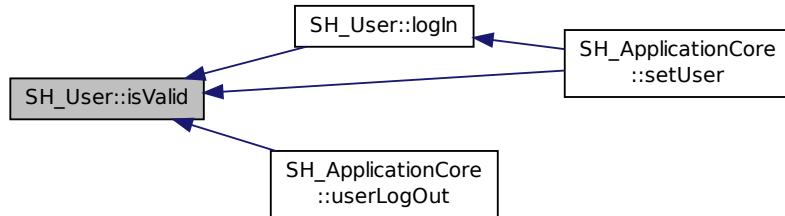
Références [m_id](#), et [m_name](#).

Référencé par [logIn\(\)](#), [SH_ApplicationCore::setUser\(\)](#), et [SH_ApplicationCore::userLogOut\(\)](#).

```

00032 {
00033     return (!this->m_name.isEmpty()) && (this->m_id != 0));
00034 }
```

Voici le graphe des appels de cette fonction :



4.68.3.8 `SH_User * SH_User ::login (QString login, QString pass) [static]`

Définition à la ligne 132 du fichier `SH_User.cpp`.

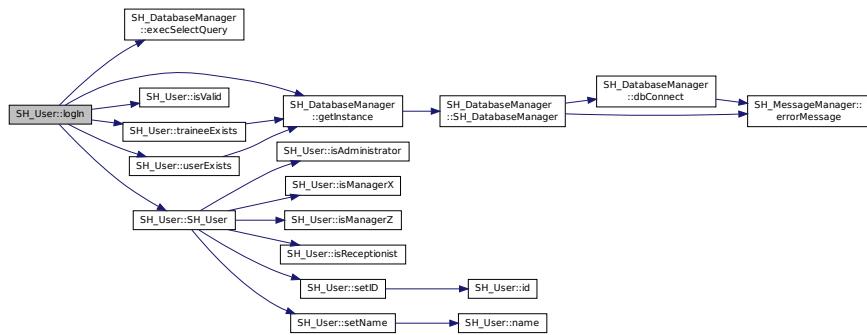
Références `SH_DatabaseManager ::execSelectQuery()`, `SH_DatabaseManager ::getInstance()`, `isValid()`, `SH_User()`, `traineeExists()`, et `userExists()`.

Référencé par `SH_ApplicationCore ::setUser()`.

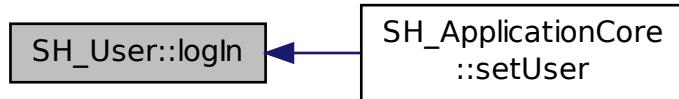
```

00133 {
00134     qDebug() << "log in";
00135     bool isValid = false;
00136     QCryptographicHash encPass(QCryptographicHash::Sha512);
00137     encPass.addData(pass.toUtf8());
00138     bool trainee=false;
00139     QStringList fields;
00140     QString table;
00141     if(userExists(login)) {
00142         fields << "ID" << "LOGIN" << "ISRECEPTIONIST" << "ISMANAGERX" << "ISMANAGERZ" << "ISADMINISTRATOR";
00143         table ="USERS";
00144     } else if(traineeExists(login)) {
00145         fields << "ID" << "LOGIN";
00146         table ="TRAINEES";
00147         trainee=true;
00148     }
00149     QSqlQuery result = SH_DatabaseManager::getInstance() ->
00150         execSelectQuery(table,fields,"LOGIN='"+login+"' AND ENCRYPTEDPASS='"+QString::fromLatin1(
00151             encPass.result().toHex()).toUpper()+"'");
00152     if(result.next()) {
00153         QSqlRecord rec = result.record();
00154         if(rec.isEmpty() || !result.isValid()) {
00155             isValid = false;
00156         } else {
00157             isValid = (rec.value(rec.indexOf("LOGIN")).toString() == login);
00158         }
00159         if(isValid) {
00160             if(trainee) {
00161                 return new SH_Trainee(rec.value(rec.indexOf("LOGIN")).toString(),rec.value(rec.
00162                     indexOf("ID")).toInt());
00163             } else {
00164                 return new SH_User(rec.value(rec.indexOf("LOGIN")).toString(),rec.value(rec.indexOf(
00165                     "ID")).toInt(),(rec.value(rec.indexOf("ISRECEPTIONIST")).toString()=="1"),(rec.value(rec.indexOf("ISMANAGERX
00166                     ").toString()=="1"),(rec.value(rec.indexOf("ISMANAGERZ")).toString()=="1"),(rec.value(rec.indexOf("ISADMINISTRATOR"))
00167                     ).toString()=="1"));
00168             }
00169         }
00170     }
00171     return new SH_User();
00172 }
```

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :



4.68.3.9 QString SH_User ::name() const

Référencé par [setName\(\)](#).

Voici le graphe des appelants de cette fonction :



4.68.3.10 void SH_User ::nameChanged() [signal]

4.68.3.11 int SH_User ::roles() const

4.68.3.12 void SH_User ::rolesChanged() [signal]

4.68.3.13 void SH_User ::setId(int id) [private]

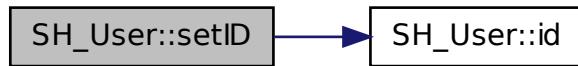
Définition à la ligne 99 du fichier [SH_User.cpp](#).

Références [id\(\)](#), et [m_id](#).

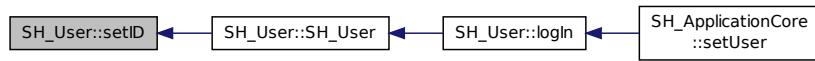
Référencé par [SH_User\(\)](#).

```
00100 {
00101     m_id = id;
00102 }
```

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :



4.68.3.14 void SH_User ::setName (QString name) [private]

Définition à la ligne 42 du fichier [SH_User.cpp](#).

Références [m_name](#), et [name\(\)](#).

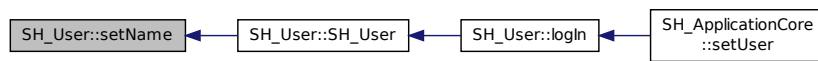
Référencé par [SH_User\(\)](#).

```
00043 {
00044     m_name = name;
00045 }
```

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :



4.68.3.15 bool SH_User::traineeExists (QString login) [static]

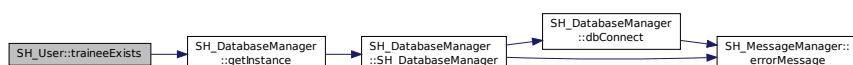
Définition à la ligne 121 du fichier [SH_User.cpp](#).

Références [SH_DatabaseManager::getInstance\(\)](#).

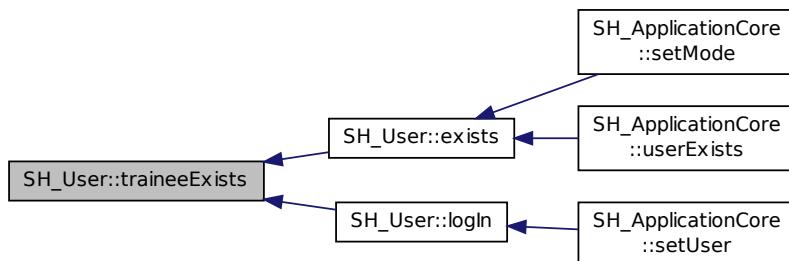
Référencé par [exists\(\)](#), et [login\(\)](#).

```
00121           {
00122     qDebug() << "trainee exists";
00123     return (SH_DatabaseManager::getInstance() ->dataCount ("TRAINEEES", "
00124     LOGIN='"+login+"'") == 1);
```

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appels de cette fonction :



4.68.3.16 bool SH_User::userExists (QString login) [static]

Définition à la ligne 110 du fichier [SH_User.cpp](#).

Références [SH_DatabaseManager::getInstance\(\)](#).

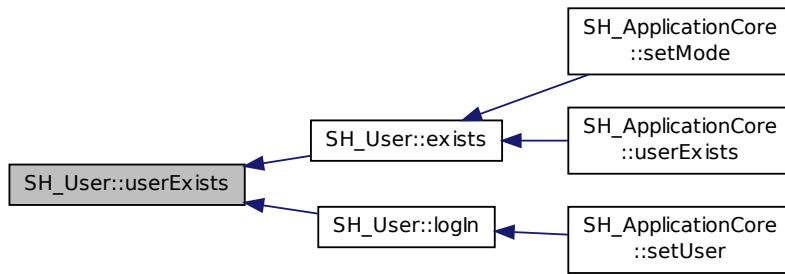
Référencé par [exists\(\)](#), et [login\(\)](#).

```
00110           {
00111     qDebug() << "user exists";
00112     return (SH_DatabaseManager::getInstance() ->dataCount ("USERS", "LOGIN='"+
00113     login+"'") == 1);
```

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appels de cette fonction :



4.68.3.17 void SH_User ::validityChanged() [signal]

4.68.4 Documentation des données membres

4.68.4.1 bool SH_User ::m_administrator [private]

m_administrator

Définition à la ligne 188 du fichier [SH_User.h](#).

Référencé par [isAdministrator\(\)](#), et [SH_User\(\)](#).

4.68.4.2 int SH_User ::m_id [private]

m_id

Définition à la ligne 192 du fichier [SH_User.h](#).

Référencé par [id\(\)](#), [isValid\(\)](#), et [setID\(\)](#).

4.68.4.3 bool SH_User ::m_managerX [private]

m_managerX

Définition à la ligne 180 du fichier [SH_User.h](#).

Référencé par [isManagerX\(\)](#), et [SH_User\(\)](#).

4.68.4.4 bool SH_User ::m_managerZ [private]

m_managerZ

Définition à la ligne 184 du fichier [SH_User.h](#).

Référencé par [isManagerZ\(\)](#), et [SH_User\(\)](#).

4.68.4.5 QString SH_User ::m_name [private]

m_name

Définition à la ligne 172 du fichier [SH_User.h](#).

Référencé par [isValid\(\)](#), et [setName\(\)](#).

4.68.4.6 bool SH_User::m_receptionist [private]

m_receptionist

Définition à la ligne [176](#) du fichier [SH_User.h](#).

Référencé par [isReceptionist\(\)](#), et [SH_User\(\)](#).

4.68.5 Documentation des propriétés

4.68.5.1 bool SH_User::administrator [read]

Définition à la ligne [23](#) du fichier [SH_User.h](#).

4.68.5.2 int SH_User::id [read]

Définition à la ligne [18](#) du fichier [SH_User.h](#).

Référencé par [SH_ApplicationCore::launchBillThread\(\)](#).

4.68.5.3 bool SH_User::managerX [read]

Définition à la ligne [21](#) du fichier [SH_User.h](#).

4.68.5.4 bool SH_User::managerZ [read]

Définition à la ligne [22](#) du fichier [SH_User.h](#).

4.68.5.5 QString SH_User::name [read]

Définition à la ligne [19](#) du fichier [SH_User.h](#).

Référencé par [SH_ApplicationCore::setMode\(\)](#), et [SH_ApplicationCore::setUser\(\)](#).

4.68.5.6 bool SH_User::receptionist [read]

Définition à la ligne [20](#) du fichier [SH_User.h](#).

4.68.5.7 int SH_User::roles [read]

Définition à la ligne [24](#) du fichier [SH_User.h](#).

4.68.5.8 bool SH_User::valid [read]

Définition à la ligne [25](#) du fichier [SH_User.h](#).

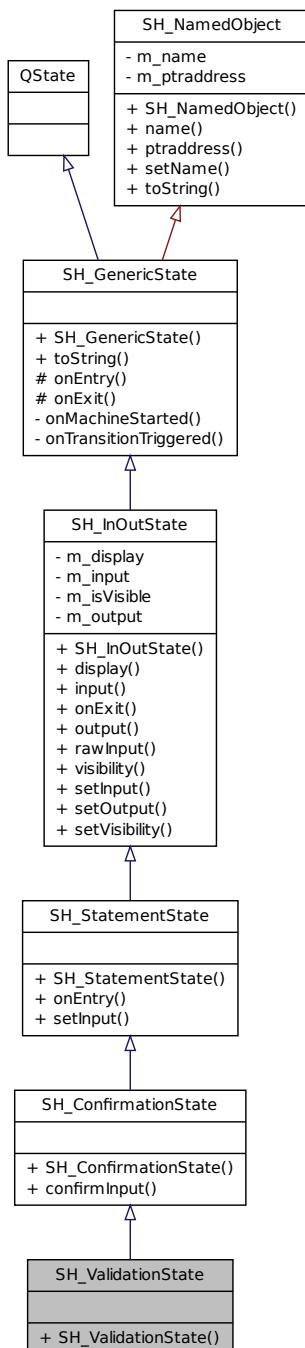
La documentation de cette classe a été générée à partir des fichiers suivants :

- [models/SH_User.h](#)
- [models/SH_User.cpp](#)

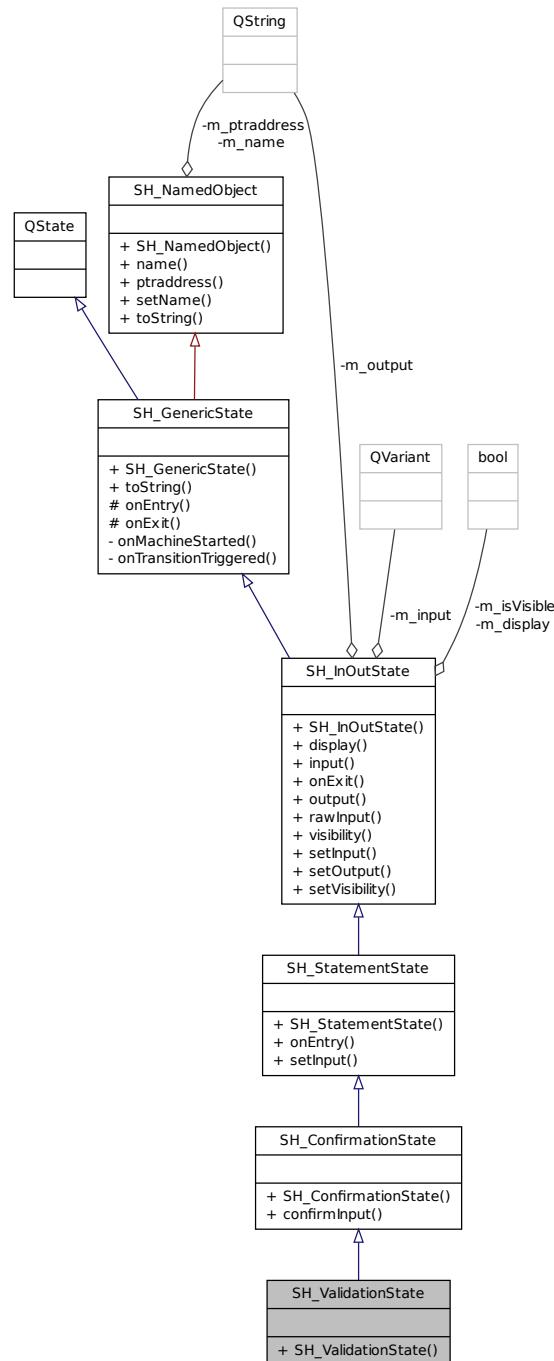
4.69 Référence de la classe SH_ValidationState

```
#include <SH_ValidationState.h>
```

Graphe d'héritage de SH_ValidationState :



Graphe de collaboration de SH_ValidationState :



Connecteurs publics

- void `confirmInput()`
- virtual void `setOutput` (const **QString** &`output`)
- virtual void `setVisibility` (**bool** `isVisible`)

Signaux

- void [next \(\)](#)
- void [resendInput \(QVariant input\)](#)
- void [sendOutput \(QVariant output\)](#)

Fonctions membres publiques

- [SH_ValidationState \(QString output, QString name, QState *parent=0\)](#)
- void [display \(bool canDisplay\)](#)
- virtual QVariant [input \(\) const](#)
- void [onEntry \(QEvent *event\)](#)
- void [onExit \(QEvent *event\)](#)
- virtual QString [output \(\) const](#)
- virtual QVariant [rawInput \(\) const](#)
- void [setInput \(const QVariant &input\)](#)
- QString [toString \(\)](#)
- bool [visibility \(\)](#)

4.69.1 Description détaillée

Définition à la ligne 10 du fichier [SH_ValidationState.h](#).

4.69.2 Documentation des constructeurs et destructeur

4.69.2.1 SH_ValidationState : :SH_ValidationState (QString output, QString name, QState * parent = 0)

Paramètres

<i>output</i>	
<i>name</i>	
<i>parent</i>	

Définition à la ligne 11 du fichier [SH_ValidationState.cpp](#).

```
00011
00012     SH_ConfirmationState(output, name, parent)
00013 {
00014
00015 }
```

:

4.69.3 Documentation des fonctions membres

4.69.3.1 void SH_ConfirmationState : :confirmInput () [slot], [inherited]

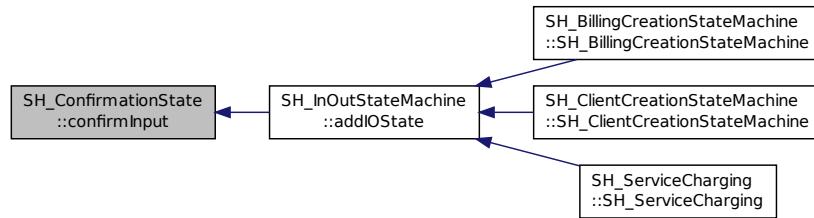
Définition à la ligne 21 du fichier [SH_ConfirmationState.cpp](#).

Références [SH_GenericState : :next\(\)](#).

Référencé par [SH_InOutStateMachine : :addIOState\(\)](#).

```
00022 {
00023     emit next();
00024 }
```

Voici le graphe des appelants de cette fonction :



4.69.3.2 void SH_InOutState ::display (bool canDisplay) [inherited]

Définition à la ligne 95 du fichier [SH_IOState.cpp](#).

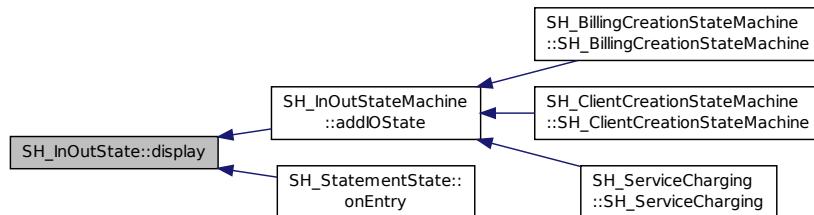
Références [SH_InOutState ::m_display](#), [SH_InOutState ::m_isVisible](#), [SH_InOutState ::m_output](#), et [SH_InOutState ::sendOutput\(\)](#).

Référencé par [SH_InOutStateMachine ::addIOState\(\)](#), et [SH_StatementState ::onEntry\(\)](#).

```

00096 {
00097     m_display=canDisplay;
00098     if(m_display && !m_output.isEmpty() && m_isVisible) {
00099         qDebug() << "resalut !" << QVariant(m_output);
00100         emit sendOutput(QVariant(m_output));
00101     }
00102 }
```

Voici le graphe des appelants de cette fonction :



4.69.3.3 QVariant SH_InOutState ::input () const [virtual], [inherited]

Définition à la ligne 20 du fichier [SH_IOState.cpp](#).

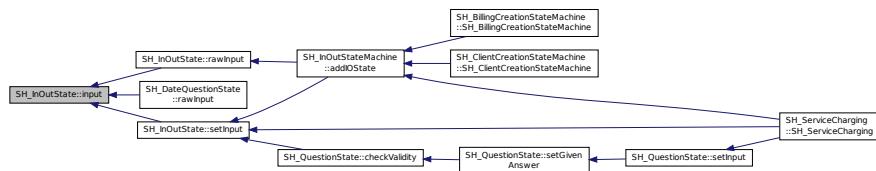
Références [SH_InOutState ::m_input](#).

Référencé par [SH_InOutState ::rawInput\(\)](#), [SH_DateQuestionState ::rawInput\(\)](#), et [SH_InOutState ::setInput\(\)](#).

```

00021 {
00022     return m_input;
00023 }
```

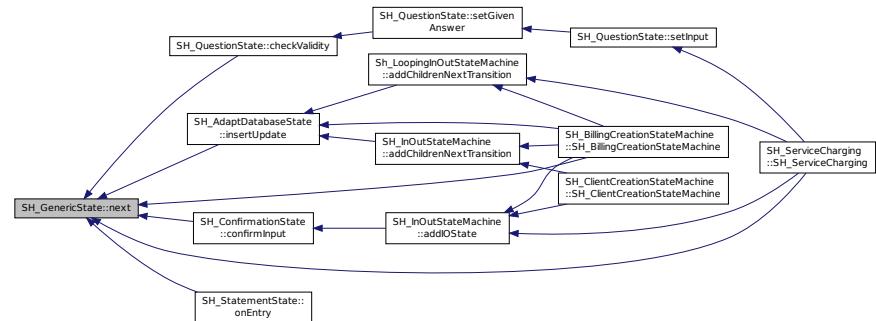
Voici le graphe des appels de cette fonction :



4.69.3.4 void SH_GenericState::next() [signal], [inherited]

Référencé par [SH_QuestionState](#) : `:checkValidity()`, [SH_ConfirmationState](#) : `:confirmInput()`, [SH_AdaptDatabase-State](#) : `:insertUpdate()`, [SH_StatementState](#) : `:onEntry()`, [SH_BillingCreationStateMachine](#) : `:SH_BillingCreation-StateMachine()`, et [SH_ServiceCharging](#) : `:SH_ServiceCharging()`.

Voici le graphe des appelants de cette fonction :



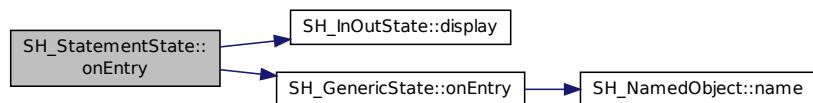
4.69.3.5 void SH_StatementState : :onEntry (QEvent * *event*) [inherited]

Définition à la ligne 33 du fichier SH_StatementState.cpp.

Références SH_InOutState ::display(), SH_GenericState ::next(), et SH_GenericState ::onEntry().

```
00034 {  
00035     SH_GenericState::onEntry(event);  
00036     display(true);  
00037     emit next();  
00038 }
```

Voici le graphe d'appel pour cette fonction :



4.69.3.6 void SH_InOutState ::onExit (QEvent * event) [inherited]

Définition à la ligne 110 du fichier [SH_IOState.cpp](#).

Références [SH_InOutState ::m_input](#), [SH_InOutState ::m_isVisible](#), [SH_GenericState ::onExit\(\)](#), et [SH_InOutState ::resendInput\(\)](#).

```
00111 {
00112     SH_GenericState::onExit(event);
00113     if(m_isVisible) {
00114         emit resendInput(m_input);
00115     }
00116 }
```

Voici le graphe d'appel pour cette fonction :



4.69.3.7 QString SH_InOutState ::output() const [virtual], [inherited]

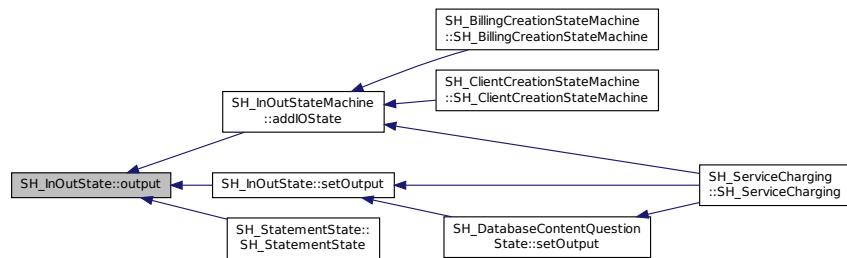
Définition à la ligne 56 du fichier [SH_IOState.cpp](#).

Références [SH_InOutState ::m_output](#).

Référencé par [SH_InOutStateMachine ::addIOState\(\)](#), [SH_InOutState ::setOutput\(\)](#), et [SH_StatementState ::SH_StatementState\(\)](#).

```
00057 {
00058     return m_output;
00059 }
```

Voici le graphe des appels de cette fonction :



4.69.3.8 QVariant SH_InOutState ::rawInput() const [virtual], [inherited]

Réimplémentée dans [SH_DateQuestionState](#), et [SH_DatabaseContentQuestionState](#).

Définition à la ligne 30 du fichier [SH_IOState.cpp](#).

Références [SH_InOutState ::input\(\)](#).

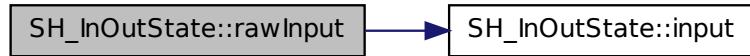
Référencé par [SH_InOutStateMachine ::addIOState\(\)](#).

```

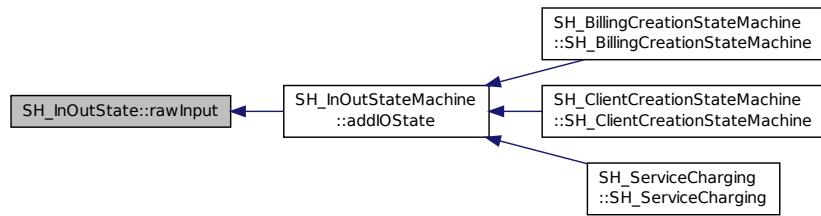
00031 {
00032     return input();
00033 }

```

Voici le graphe d'appel pour cette fonction :



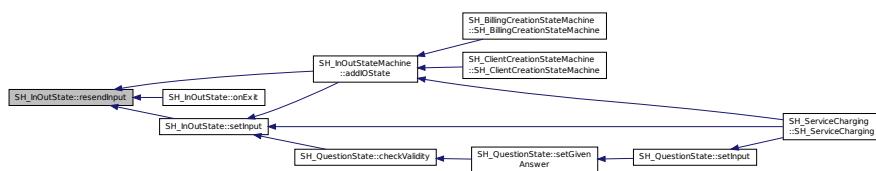
Voici le graphe des appelants de cette fonction :



4.69.3.9 void SH_InOutState ::resendInput (QVariant *input*) [signal], [inherited]

Référencé par [SH_InOutStateMachine ::addIOState\(\)](#), [SH_InOutState ::onExit\(\)](#), et [SH_InOutState ::setInput\(\)](#).

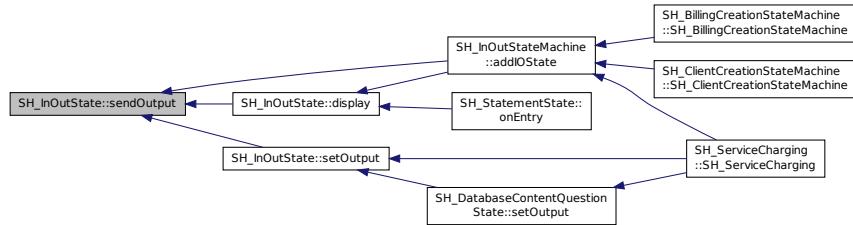
Voici le graphe des appelants de cette fonction :



4.69.3.10 void SH_InOutState ::sendOutput (QVariant *output*) [signal], [inherited]

Référencé par [SH_InOutStateMachine ::addIOState\(\)](#), [SH_InOutState ::display\(\)](#), et [SH_InOutState ::setOutput\(\)](#).

Voici le graphe des appelants de cette fonction :



4.69.3.11 void SH_StatementState ::setInput (const QVariant & input) [virtual], [inherited]

Réimplémentée à partir de [SH_InOutState](#).

Définition à la ligne 21 du fichier [SH_StatementState.cpp](#).

```

00022 {
00023     Q_UNUSED(input);
00024     /*DO NOTHING*/
00025 }
```

4.69.3.12 void SH_InOutState ::setOutput (const QString & output) [virtual], [slot], [inherited]

Réimplémentée dans [SH_DatabaseContentQuestionState](#).

Définition à la ligne 68 du fichier [SH_IOState.cpp](#).

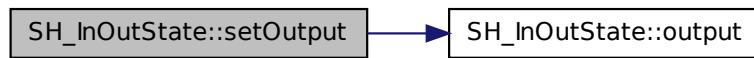
Références [SH_InOutState ::m_isVisible](#), [SH_InOutState ::m_output](#), [SH_InOutState ::output\(\)](#), et [SH_InOutState ::sendOutput\(\)](#).

Référencé par [SH_DatabaseContentQuestionState ::setOutput\(\)](#), et [SH_ServiceCharging ::SH_ServiceCharging\(\)](#).

```

00069 {
00070     m_output = output;
00071     if(m_isVisible) {
00072         emit sendOutput(QVariant(m_output));
00073     }
00074 }
```

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :



4.69.3.13 void SH_InOutState ::setVisibility (bool isVisible) [virtual], [slot], [inherited]

Définition à la ligne 81 du fichier [SH_IOState.cpp](#).

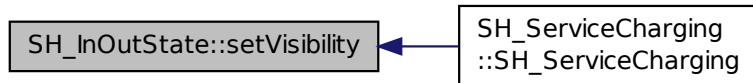
Références [SH_InOutState ::m_isVisible](#).

Référencé par [SH_ServiceCharging ::SH_ServiceCharging\(\)](#).

```

00082 {
00083     m_isVisible = isVisible;
00084 }
```

Voici le graphe des appelants de cette fonction :



4.69.3.14 QString SH_GenericState ::toString () [virtual], [inherited]

Réimplémentée à partir de [SH_NamedObject](#).

Définition à la ligne 21 du fichier [SH_GenericDebugableState.cpp](#).

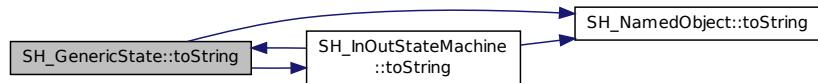
Références [SH_NamedObject ::toString\(\)](#), et [SH_InOutStateMachine ::toString\(\)](#).

Référencé par [SH_InOutStateMachine ::addChildsNextTransition\(\)](#), [SH_DateQuestionState ::rawInput\(\)](#), et [SH_InOutStateMachine ::toString\(\)](#).

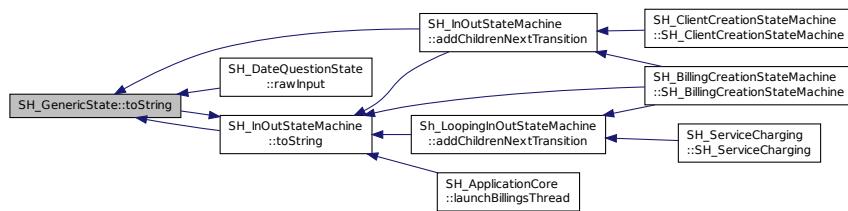
```

00022 {
00023     QStateMachine* machine = this->machine();
00024     SH_InOutStateMachine* mach = qobject_cast<
00025         SH_InOutStateMachine *>(machine);
00026     if(mach) {
00027         return SH_NamedObject::toString() + " [in "+mach->
00028             toString()+"] ";
00029     } else {
00030         return SH_NamedObject::toString();
00031     }
00032 }
```

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appels de cette fonction :



4.69.3.15 bool SH_InOutState ::visibility() [inherited]

Définition à la ligne 91 du fichier [SH_IOState.cpp](#).

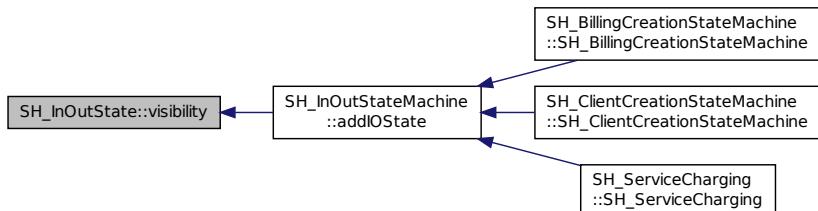
Références [SH_InOutState ::m_isVisible](#).

Référencé par [SH_InOutStateMachine ::addIOState\(\)](#).

```

00091     {
00092         return m_isVisible;
00093     }
  
```

Voici le graphe des appels de cette fonction :

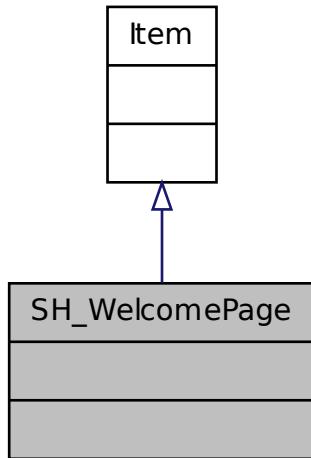


La documentation de cette classe a été générée à partir des fichiers suivants :

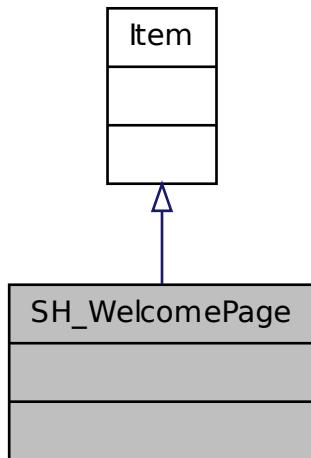
- [logic/SH_ValidationState.h](#)
- [logic/SH_ValidationState.cpp](#)

4.70 Référence de la classe SH_WelcomePage

Graphe d'héritage de SH_WelcomePage :



Graphe de collaboration de SH_WelcomePage :



Signaux

- void `clicked ()`
- void `loggedOut ()`
- void `logOut ()`
- void `quit ()`

- void [reload \(\)](#)

4.70.1 Description détaillée

Définition à la ligne [4](#) du fichier [SH_WelcomePage.qml](#).

4.70.2 Documentation des fonctions membres

4.70.2.1 void [SH_WelcomePage ::clicked \(\) \[signal\]](#)

4.70.2.2 void [SH_WelcomePage ::loggedOut \(\) \[signal\]](#)

4.70.2.3 void [SH_WelcomePage ::logOut \(\) \[signal\]](#)

4.70.2.4 void [SH_WelcomePage ::quit \(\) \[signal\]](#)

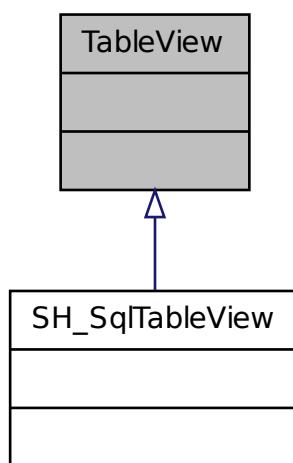
4.70.2.5 void [SH_WelcomePage ::reload \(\) \[signal\]](#)

La documentation de cette classe a été générée à partir du fichier suivant :

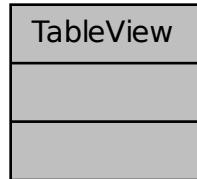
- [views/qml/**SH_WelcomePage.qml**](#)

4.71 Référence de la classe TableView

Graphe d'héritage de TableView :



Graphe de collaboration de TableView :

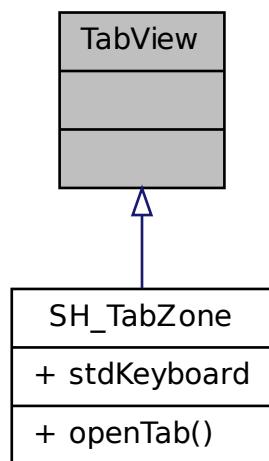


La documentation de cette classe a été générée à partir du fichier suivant :

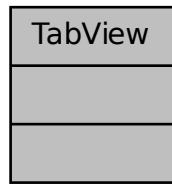
- `views/qml/SH_SqlTableView.qml`

4.72 Référence de la classe TabView

Graphe d'héritage de TabView :



Graphe de collaboration de TabView :



La documentation de cette classe a été générée à partir du fichier suivant :

– views/qml/[SH_TabZone.qml](#)

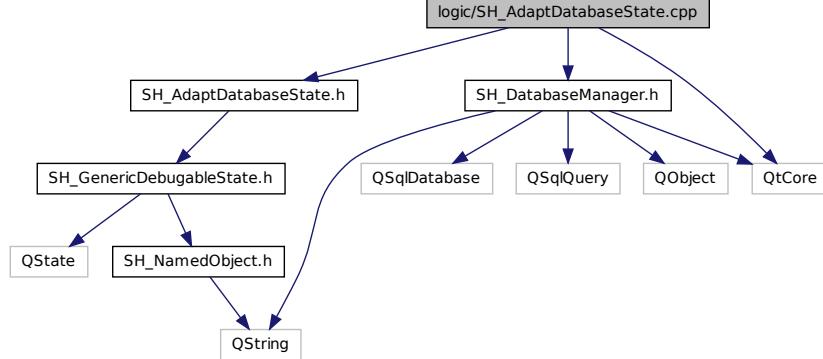
Chapitre 5

Documentation des fichiers

5.1 Référence du fichier logic/SH_AdaptDatabaseState.cpp

```
#include "SH_AdaptDatabaseState.h"
#include "SH_DatabaseManager.h"
#include <QtCore>
```

Graphe des dépendances par inclusion de SH_AdaptDatabaseState.cpp :

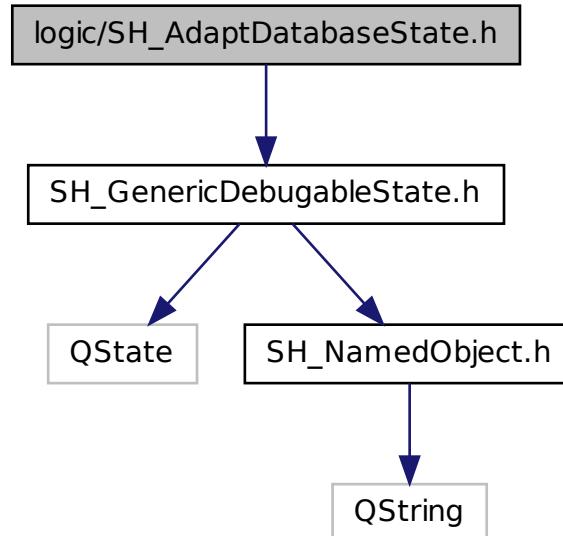


5.2 SH_AdaptDatabaseState.cpp

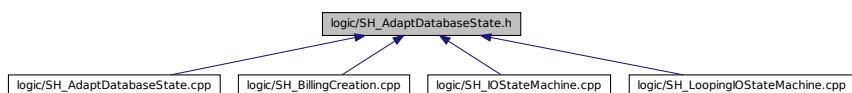
```
00001 #include "SH_AdaptDatabaseState.h"
00002 #include "SH_DatabaseManager.h"
00003 #include <QtCore>
00004
00009 SH_AdaptDatabaseState::SH_AdaptDatabaseState(QString name,
    QState *parent) :
00010     SH_GenericState(name, parent)
00011 {
00012 }
00013
00018 QVariant SH_AdaptDatabaseState::insertUpdate(QString table, QVariantMap
    content)
00019 {
00020     QVariant id = SH_DatabaseManager::getInstance()->
        execInsertReturningQuery(table, content, "id");
00021     if(id.isValid()) {
00022         emit next();
00023     }
00024     return id;
00025 }
```

5.3 Référence du fichier logic/SH_AdaptDatabaseState.h

```
#include "SH_GenericDebugableState.h"
Graphe des dépendances par inclusion de SH_AdaptDatabaseState.h :
```



Ce graphe montre quels fichiers incluent directement ou indirectement ce fichier :



Classes

- class [SH_AdaptDatabaseState](#)

5.4 SH_AdaptDatabaseState.h

```

00001 #ifndef ADAPTDATABASE_H
00002 #define ADAPTDATABASE_H
00003 #include "SH_GenericDebugableState.h"
00004
00009 class SH_AdaptDatabaseState : public SH_GenericState
00010 {
00011     Q_OBJECT
00012 public:
00019     SH_AdaptDatabaseState(QString name, QState *parent = 0);
00020
00028     QVariant insertUpdate(QString table, QVariantMap content);
00029
00030 signals:
00031
  
```

```

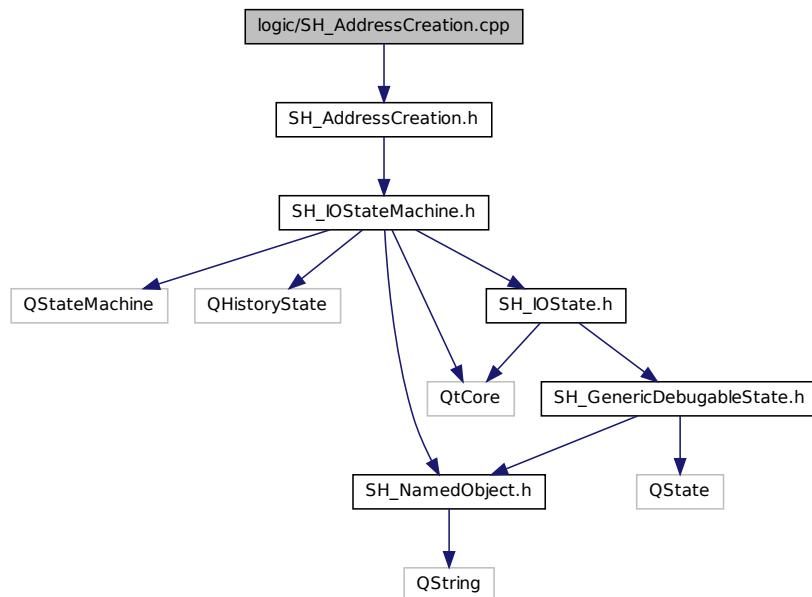
00032 public slots:
00033
00034 };
00035
00036 #endif /* ADAPTDATABASE_H */

```

5.5 Référence du fichier logic/SH_AddressCreation.cpp

#include "SH_AddressCreation.h"

Graphe des dépendances par inclusion de SH_AddressCreation.cpp :



5.6 SH_AddressCreation.cpp

```

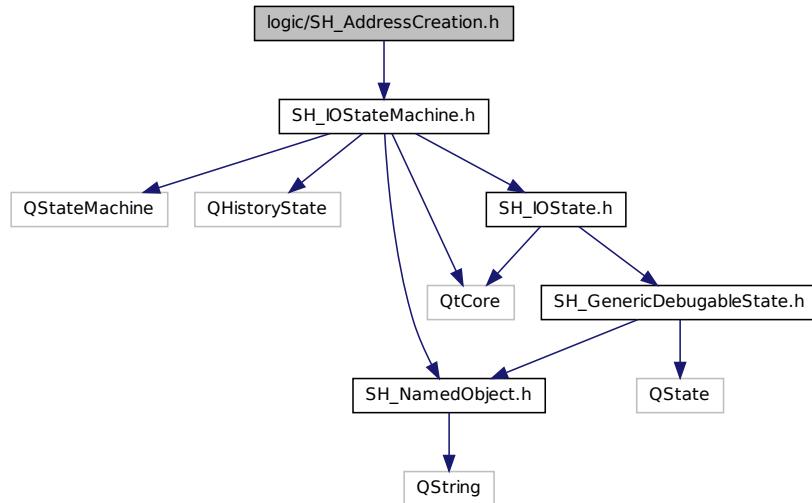
00001 #include "SH_AddressCreation.h"
00002
00008 SH_AddressCreationStateMachine::SH_AddressCreationStateMachine
00009     (QString name, QObject *parent) :
00010         SH_InOutStateMachine("ADDRESSES",name, parent)
00011 /*TODO: rue, numéro, code postal, ville, pays, destinataire*/
00012 emit next();
00013 }
00014
00015

```

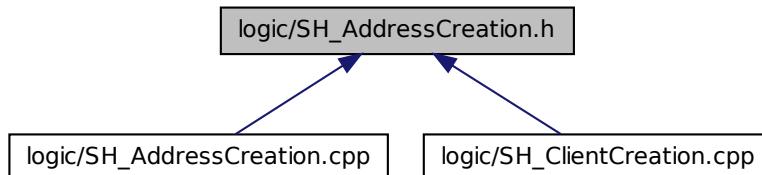
5.7 Référence du fichier logic/SH_AddressCreation.h

#include "SH_IStateMachine.h"

Graphe des dépendances par inclusion de SH_AddressCreation.h :



Ce graphe montre quels fichiers incluent directement ou indirectement ce fichier :



Classes

- class `SH_AddressCreationStateMachine`
The `SH_AddressCreationStateMachine` class.

5.8 SH_AddressCreation.h

```

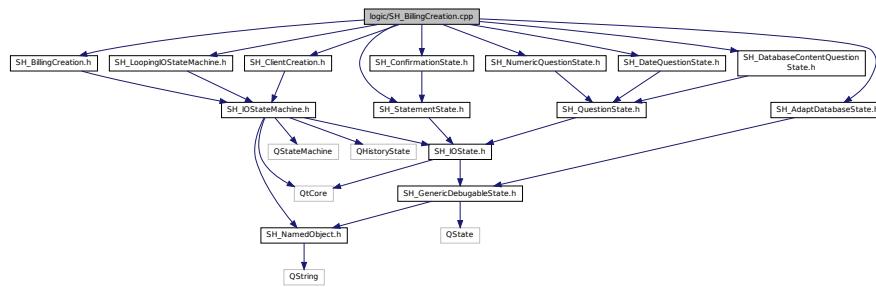
00001 #ifndef ADDRESSCREATION_H
00002 #define ADDRESSCREATION_H
00003 #include "SH_IOSStateMachine.h"
00007 class SH_AddressCreationStateMachine : public
    SH_InOutStateMachine
00008 {
00009     Q_OBJECT
00010 public:
00017     SH_AddressCreationStateMachine(QString name,
        QObject *parent = 0);
00018
00019 signals:
00020
00021 public slots:
00022
  
```

```
00023 };
00024
00025 #endif /* ADDRESSCREATION_H */
```

5.9 Référence du fichier logic/SH_BillingCreation.cpp

```
#include "SH_BillingCreation.h"
#include "SH_StatementState.h"
#include "SH_NumericQuestionState.h"
#include "SH_DateQuestionState.h"
#include "SH_DatabaseContentQuestionState.h"
#include "SH_ConfirmationState.h"
#include "SH_LoopingIOStateMachine.h"
#include "SH_AdaptDatabaseState.h"
#include "SH_ClientCreation.h"
```

Graphe des dépendances par inclusion de SH_BillingCreation.cpp :



5.10 SH_BillingCreation.cpp

```
00001 #include "SH_BillingCreation.h"
00002 #include "SH_StatementState.h"
00003 #include "SH_NumericQuestionState.h"
00004 #include "SH_DateQuestionState.h"
00005 #include "SH_DatabaseContentQuestionState.h"
00006 #include "SH_ConfirmationState.h"
00007 #include "SH_LoopingIOStateMachine.h"
00008 #include "SH_AdaptDatabaseState.h"
00009 #include "SH_ClientCreation.h"
00010
00014 SH_BillingCreationStateMachine::SH_BillingCreationStateMachine
    (QString name, QObject *parent) :
00015     SH_InOutStateMachine("BILLINGS",name, parent)
00016 {
00017     qDebug() << "facturation";
00018
00019     SH_StatementState* intro = new SH_StatementState("Création d'une
        facturation", "intro billing creation");
00020     SH_NumericQuestionState* nbAdults = new
        SH_NumericQuestionState("Veuillez entrer le nombre d'adultes", "adults billing
        creation", 0);
00021     SH_NumericQuestionState* nbChildren = new
        SH_NumericQuestionState("Veuillez entrer le nombre d'enfants", "children billing
        creation", 0);
00022     SH_DateQuestionState* arrivingDate = new
        SH_DateQuestionState("Veuillez entrer la date d'arrivée", "arriving date billing
        creation", true,true);
00023     SH_DateQuestionState* departureDate = new
        SH_DateQuestionState("Veuillez entrer la date de départ prévue", "departure date
        billing creation", false,true);
00024     SH_DatabaseContentQuestionState* client = new
        SH_DatabaseContentQuestionState("Veuillez entrer le nom du client à facturer
        ", "main client billing creation", "CLIENTS", "NAME");
00025     SH_ClientCreationStateMachine* clientCreation = new
        SH_ClientCreationStateMachine("main client creation in billing creation");
00026     SH_NumericQuestionState* nbRooms = new
```

```

SH_NumericQuestionState("Veuillez entrer le nombre de chambres", "nb rooms billing
creation", 1);
00027     /*DatabaseContentQuestionState* type = new DatabaseContentQuestionState("Veuillez choisir le type de
facturation","billing type billing creation", "BILLINGTYPES", "CODE");*/
00028     SH_DatabaseContentQuestionState* type = new
SH_DatabaseContentQuestionState("Veuillez choisir le type de facturation","
billing type billing creation", "BILLINGTYPES", "ID");
00029     Sh_LoopingInOutStateMachine* roomsAffectation = new
Sh_LoopingInOutStateMachine("ROOMSOCCUPATION", "rooms affectation billing
creation");
00030     Sh_LoopingInOutStateMachine* billsCreation = new
Sh_LoopingInOutStateMachine("BILLS", "bills creation billing creation");
00031     Sh_LoopingInOutStateMachine* clientList = new
Sh_LoopingInOutStateMachine("CLIENTS", "bills creation billing creation");
00032     SH_ConfirmationState* confirmPart1 = new
SH_ConfirmationState("Veuillez appuyer sur la touche \"CONFIRMER\" pour passer à
l'étape suivante", "confirm part 1");
00033     SH_AdaptDatabaseState* saveState = new
SH_AdaptDatabaseState("enregistrement de la machine "+

toString());
00034     SH_ConfirmationState* confirmAll = new
SH_ConfirmationState("Veuillez appuyer sur la touche \"CONFIRMER\" pour passer à
l'étape suivante", "confirm all");
00035     QFinalState* final = new QFinalState();
00036
00037
00038
00039     connect(nbAdults, &SH_GenericState::exited, [=]() {
00040         clientList->setLimit(getContentValue("NBADULTS").toInt()-1);
00041     });
00042
00043     connect(nbRooms, &SH_GenericState::exited, [=]() {
00044         roomsAffectation->setLimit(getContentValue("NBROOMS").toInt());
00045     });
00046
00047     connect(type, &SH_GenericState::exited, [=]() {
00048         billsCreation->setLimit(getContentValue("NBROOMS").toInt() * (
getContentValue("BILLINGTYPE_ID").toInt() % 3));
00049     });
00050
00051     connect(saveState, &SH_GenericState::exited, [=]() {
00052         roomsAffectation->setPersistentContentValue(
getContentValue("ID"), "BILLING_ID");
00053         billsCreation->setPersistentContentValue(
getContentValue("ID"), "BILLING_ID");
00054     });
00055
00056
00057
00058
00059
00060     SH_DatabaseContentQuestionState* rooms = new
SH_DatabaseContentQuestionState("Veuillez entrer un numéro de chambre", "room
billing creation", "ROOMS", "NUMBER");
00061     QFinalState* finalRooms = new QFinalState();
00062     roomsAffectation->addChildsNextTransition(rooms, finalRooms);
00063     roomsAffectation->addIOState(rooms, "ROOM_NUMBER");
00064     roomsAffectation->addState(finalRooms);
00065     roomsAffectation->setInitialState(rooms);
00066
00067
00068
00069     SH_DatabaseContentQuestionState* supplClient = new
SH_DatabaseContentQuestionState("Veuillez entrer le nom du client (adulte)
supplémentaire ou appuyer sur la touche \"CONFIRMER\" pour passer à la suite de la facturation", "other client
billing creation", "CLIENTS", "NAME");
00070     SH_ClientCreationStateMachine* supplClientCreation = new
SH_ClientCreationStateMachine("other client creation in billing creation");
00071     connect(clientList, &SH_InOutStateMachine::confirmInput, [=]() {
00072         clientList->stopLooping();
00073         supplClient->next();
00074     });
00075     QFinalState* finalClients = new QFinalState();
00076     clientList->addChildsNextTransition(supplClient, finalClients);
00077     connect(supplClient, &SH_QuestionState::answerInvalid, [=]() {
00078         supplClientCreation->setContentValue(supplClient->
givenAnswer(), "NAME");
00079         supplClient->addTransition(supplClient, SIGNAL(next()), supplClientCreation);
00080         emit supplClient->next();
00081     });
00082     clientList->addChildsNextTransition(supplClientCreation, finalClients);
00083     clientList->addState(finalClients);
00084     clientList->addState(supplClient);
00085     clientList->setInitialState(supplClient);
00086
00087
00088

```

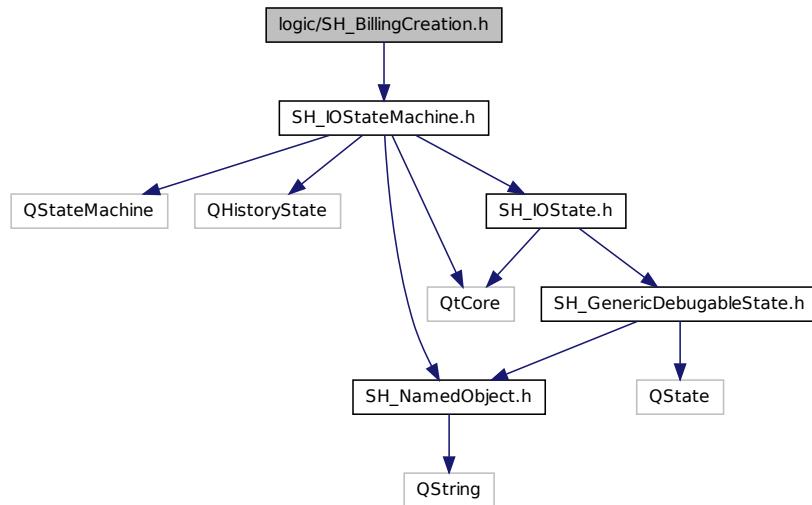
```

00089
00090     QFinalState* finalBills = new QFinalState();
00091     SH_GenericState* bills = new SH_GenericState("bill id attribution");
00092     connect(bills, &SH_GenericState::entered, [=]() {
00093         this->setContentValue(QVariant(billsCreation->current()), "BILLINGBILL_ID");
00094         int billingType = getContentValue("BILLINGTYPE_ID").toInt();
00095         int billType;
00096         if(billingType <= 2) {
00097             billType = 1+billingType; /*nb facture par chambre*/
00098         } else {
00099             billType = (billsCreation->current() % (1+(billingType % 3)));
00100         }
00101
00102         this->setContentValue(QVariant(billType), "BILLTYPE_ID");
00103         emit bills->next();
00104     });
00105     billsCreation->addChildrenNextTransition(bills, finalBills);
00106     billsCreation->addState(finalBills);
00107     billsCreation->addState(bills);
00108     billsCreation->setInitialState(bills);
00109
00110
00111
00112
00113     this->addChildsNextTransition(intro, nbAdults);
00114     this->addChildsNextTransition(nbAdults, nbChildren);
00115     this->addChildsNextTransition(nbChildren, arrivingDate);
00116     this->addChildsNextTransition(arrivingDate, departureDate);
00117     this->addChildsNextTransition(departureDate, client);
00118     this->addChildsNextTransition(client, nbRooms);
00119     connect(client, &SH_QuestionState::answerInvalid, [=]() {
00120         clientCreation->setContentValue(client->givenAnswer(), "NAME");
00121         client->addTransition(client, SIGNAL(next()), clientCreation);
00122         emit client->next();
00123     });
00124     this->addChildsNextTransition(clientCreation, nbRooms);
00125     this->addChildsNextTransition(nbRooms, type);
00126     /*this->addChildsNextTransition(type, final);*/
00127     this->addChildsNextTransition(type, confirmPart1);
00128     confirmPart1->addTransition(confirmPart1, SIGNAL(next()), confirmPart1);
00129     connect(confirmPart1, &SH_GenericState::exited, [=]() {
00130         connect(saveState, &SH_GenericState::entered, [=]() {
00131             setContentValue(saveState->insertUpdate(
00132                 m_tableName, m_iocContent), "ID");
00133         });
00134         saveState->addTransition(saveState, SIGNAL(next()), confirmAll);
00135         saveState->addTransition(saveState, SIGNAL(next()), roomsAffection);
00136         this->addChildsNextTransition(roomsAffection, billsCreation);
00137         this->addChildsNextTransition(billsCreation, clientList);
00138         this->addChildsNextTransition(clientList, confirmAll);
00139         this->addChildsNextTransition(confirmAll, final);
00140
00141         this->addIOState(intro, "");
00142         this->addIOState(nbAdults, "NBADULTS");
00143         this->addIOState(nbChildren, "NBCHILDREN");
00144         this->addIOState(arrivingDate, "ARRIVINGDATE");
00145         this->addIOState(departureDate, "EXPECTEDDEPARTUREDATE");
00146         this->addIOState(client, "CLIENT_ID");
00147         this->addIOState(nbRooms, "NBROOMS");
00148         this->addIOState(type, "BILLINGTYPE_ID");
00149         this->addIOState(confirmPart1, "");
00150         this->addIOState(confirmAll, "");
00151         this->addIOStateMachine(billsCreation);
00152         this->addIOStateMachine(roomsAffection);
00153         this->addIOStateMachine(clientList);
00154         this->addState(final);
00155
00156     this->setInitialState(intro);
00157 }
```

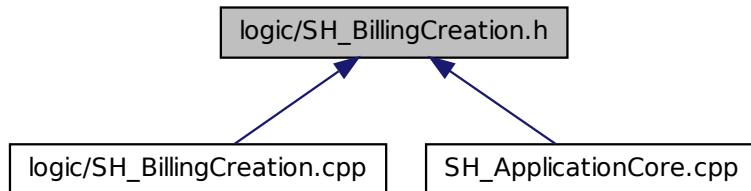
5.11 Référence du fichier logic/SH_BillingCreation.h

```
#include "SH_IOSMachine.h"
```

Graphe des dépendances par inclusion de SH_BillingCreation.h :



Ce graphe montre quels fichiers incluent directement ou indirectement ce fichier :



Classes

- class `SH_BillingCreationStateMachine`
The `SH_BillingCreationStateMachine` class.

5.12 SH_BillingCreation.h

```

00001 #ifndef BILLINGCREATIONSTATEMACHINE_H
00002 #define BILLINGCREATIONSTATEMACHINE_H
00003 #include "SH_IStateMachine.h"
00007 class SH_BillingCreationStateMachine : public
    SH_InOutStateMachine
00008 {
00009     Q_OBJECT
00010 public:
00016     SH_BillingCreationStateMachine(QString name,
        QObject *parent = 0);
00017
00018 signals:
00019
00020 public slots:
  
```

```

00021
00022 };
00023
00024 #endif /* BILLINGCREATIONSTATEMACHINE_H */

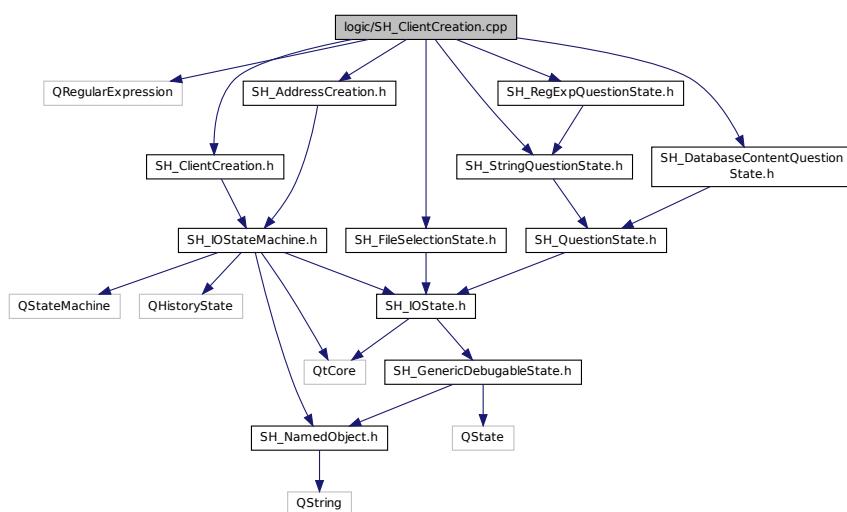
```

5.13 Référence du fichier logic/SH_ClientCreation.cpp

```

#include <QRegularExpression>
#include "SH_ClientCreation.h"
#include "SH_DatabaseContentQuestionState.h"
#include "SH_StringQuestionState.h"
#include "SH_FileSelectionState.h"
#include "SH_RegExpQuestionState.h"
#include "SH_AddressCreation.h"
Graphe des dépendances par inclusion de SH_ClientCreation.cpp :

```



5.14 SH_ClientCreation.cpp

```

00001 #include <QRegularExpression>
00002 #include "SH_ClientCreation.h"
00003 #include "SH_DatabaseContentQuestionState.h"
00004 #include "SH_StringQuestionState.h"
00005 #include "SH_FileSelectionState.h"
00006 #include "SH_RegExpQuestionState.h"
00007 #include "SH_AddressCreation.h"
00008
00013 SH_ClientCreationStateMachine::SH_ClientCreationStateMachine
    (QString name, QObject *parent) :
00014     SH_InOutStateMachine("CLIENTS",name, parent)
00015 {
00016     SH_AddressCreationStateMachine* address = new
        SH_AddressCreationStateMachine("address in client creation");
00017     SH_DatabaseContentQuestionState* nationality = new
        SH_DatabaseContentQuestionState("Veuillez entrer le code de nationalité du
        client", "nationality in client creation", "NATIONALITIES","CODE");
00018     SH_FileSelectionState* IDscan = new
        SH_FileSelectionState("Veuillez sélectionner l'image scannée des papiers d'identité du
        client","ID image in client creation");
00019     SH_RegExpQuestionState* phone = new
        SH_RegExpQuestionState("Veuillez entrer le numéro de téléphone du client avec le
        préfixe international","phone client creation",QRegularExpression("[[+|00][[1-9]]{3}\d{8}]?"));
00020     SH_RegExpQuestionState* email = new
        SH_RegExpQuestionState("Veuillez entrer l'adresse email du client","email client
        creation",QRegularExpression("[\d\w.\-%]+\@\d\w.-+\.\w+]?"));

```

```

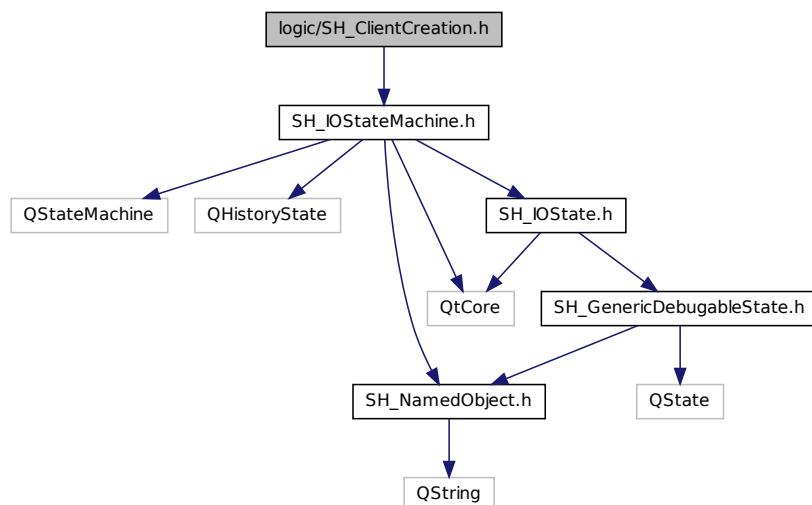
00021     QFinalState* final = new QFinalState();
00022
00023     this->addState(final);
00024     this->addIOState(IDscan, "IDSCAN");
00025     this->addIOState(nationality, "NATIONALITY_CODE");
00026     this->addIOState(phone, "PHONE");
00027     this->addIOState(email, "EMAIL");
00028     connect(address, &SH_InOutStateMachine::exited, [=]() {setContentValue(address->
00029         getContentValue("ID"), "HOMEADDRESS_ID");});
00030     this->addChildsNextTransition(IDscan, address);
00031     this->addChildsNextTransition(address, phone);
00032     this->addChildsNextTransition(phone, email);
00033     this->addChildsNextTransition(email, nationality);
00034     this->setInitialState(IDscan);
00035
00036     /*connect (IDscan,&QState::entered, [=]() {
00037         connect(this, &IOStateMachine::confirmInput, IDscan, &GenericState::next);
00038     });*/

```

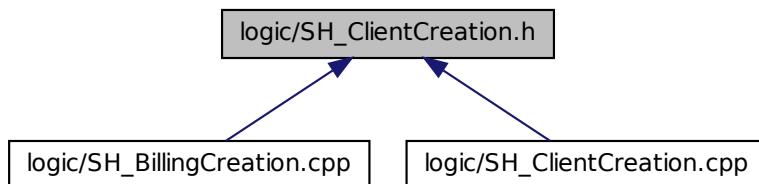
5.15 Référence du fichier logic/SH_ClientCreation.h

#include "SH_IOStateMachine.h"

Graphe des dépendances par inclusion de SH_ClientCreation.h :



Ce graphe montre quels fichiers incluent directement ou indirectement ce fichier :



Classes

- class **SH_ClientCreationStateMachine**
The [SH_ClientCreationStateMachine](#) class.

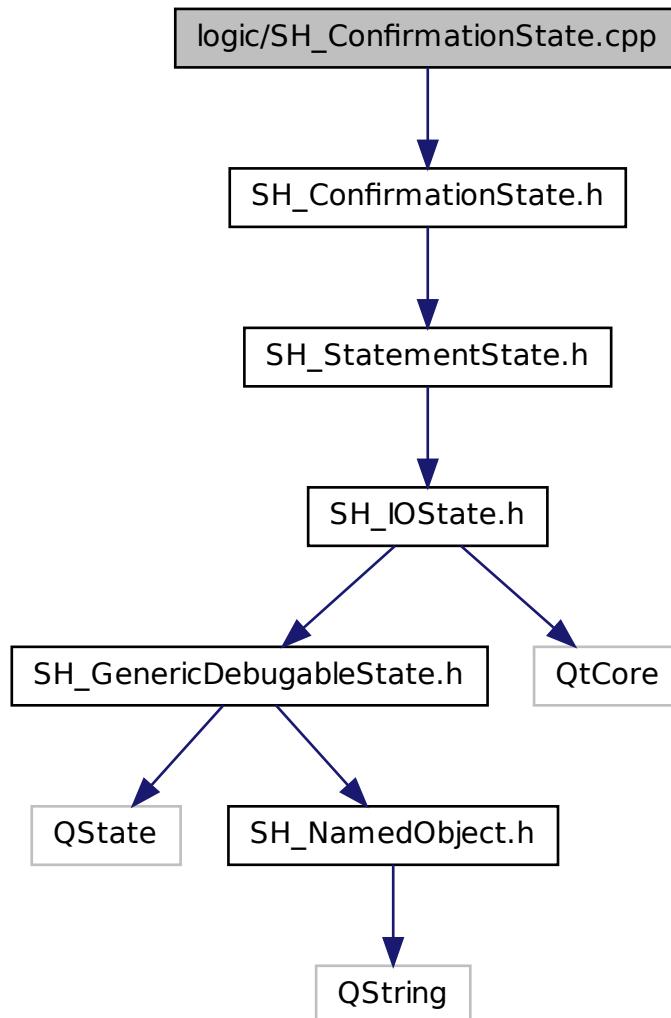
5.16 SH_ClientCreation.h

```
00001 #ifndef CLIENTCREATION_H
00002 #define CLIENTCREATION_H
00003 #include "SH_IOStateMachine.h"
00004
00008 class SH_ClientCreationStateMachine : public
00009     SH_InOutStateMachine
00010     Q_OBJECT
00011 public:
00012     SH_ClientCreationStateMachine(QString name,
00013         QObject *parent = 0);
00014
00015 signals:
00016
00017 public slots:
00018
00019 };
00020
00021
00022
00023
00024
00025
00026 #endif /* CLIENTCREATION_H */
```

5.17 Référence du fichier logic/SH_ConfirmationState.cpp

```
#include "SH_ConfirmationState.h"
```

Graphe des dépendances par inclusion de SH_ConfirmationState.cpp :



5.18 SH_ConfirmationState.cpp

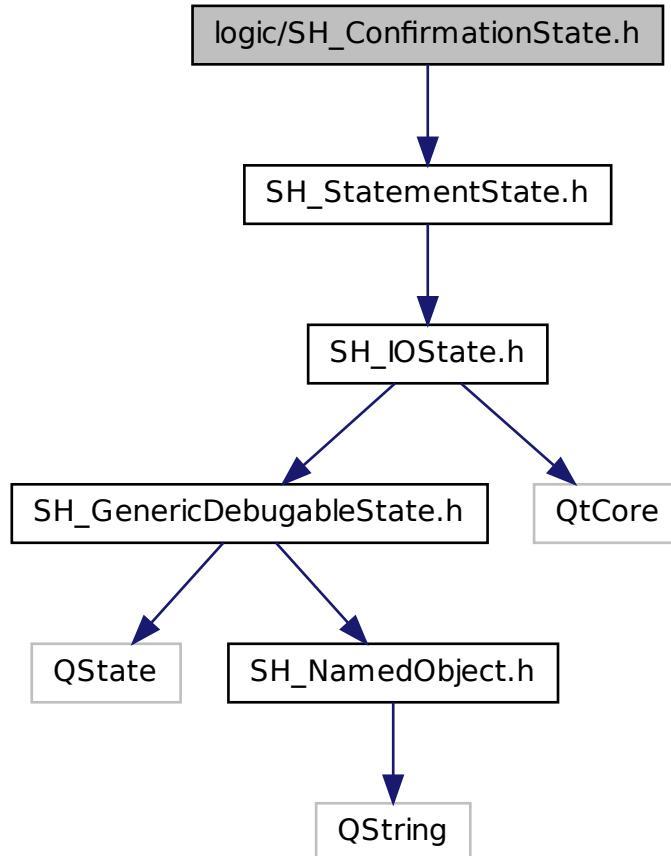
```

00001 #include "SH_ConfirmationState.h"
00002
00003
00010 SH_ConfirmationState::SH_ConfirmationState(QString output,
00011     QString name, QState *parent) :
00012 {
00013     qDebug() << "confirmation !";
00014 }
00015
00021 void SH_ConfirmationState::confirmInput()
00022 {
00023     emit next();
00024 }
  
```

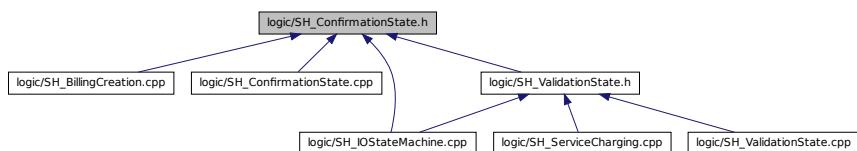
5.19 Référence du fichier logic/SH_ConfirmationState.h

```
#include "SH_StatementState.h"
```

Graphe des dépendances par inclusion de SH_ConfirmationState.h :



Ce graphe montre quels fichiers incluent directement ou indirectement ce fichier :



Classes

- class [SH_ConfirmationState](#)

La classe ConfirmationState représente un état dans lequel le système attend que l'utilisateur appuie sur une touche de confirmation.

5.20 SH_ConfirmationState.h

```

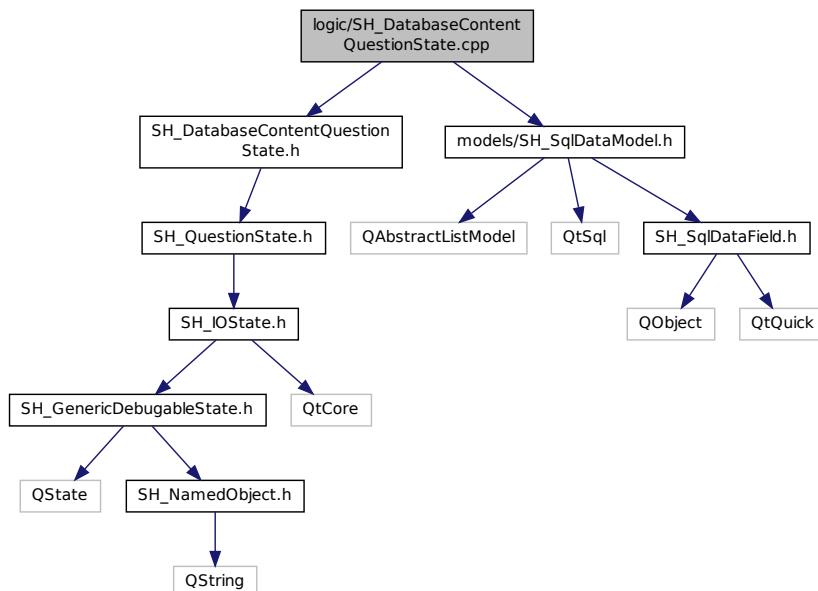
00001 #ifndef CONFIRMATIONSTATE_H
00002 #define CONFIRMATIONSTATE_H
00003 #include "SH_StatementState.h"
00004
00013 class SH_ConfirmationState : public SH_StatementState
00014 {
00015     Q_OBJECT
00016 public:
00025     SH_ConfirmationState(QString output, QString name,
00026     QState *parent = 0);
00027
00028 signals:
00030
00031 public slots:
00037     void confirmInput();
00038
00039 };
00040
00041 #endif /* CONFIRMATIONSTATE_H */

```

5.21 Référence du fichier logic/SH_DatabaseContentQuestionState.cpp

```
#include "SH_DatabaseContentQuestionState.h"
#include "models/SH_SqlDataModel.h"
```

Graphe des dépendances par inclusion de SH_DatabaseContentQuestionState.cpp :



5.22 SH_DatabaseContentQuestionState.cpp

```

00001 #include "SH_DatabaseContentQuestionState.h"
00002 #include "models/SH_SqlDataModel.h"
00003
00010 SH_DatabaseContentQuestionState::SH_DatabaseContentQuestionState(
00011     QString question, QString name, QString databaseTable, QString tableField, QString databaseCondition,
00012     QState *parent) :
00011     SH_QuestionState(question, name, parent), m_table(databaseTable), m_condition(
00012         databaseCondition), m_field(tableField)

```

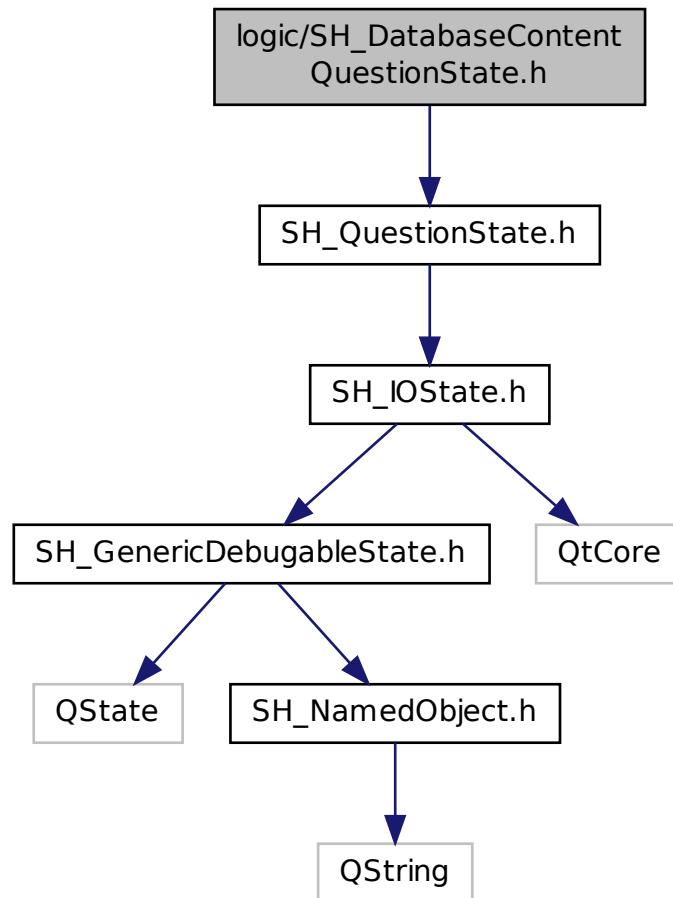
```

00012 {
00013     qDebug() << "multiple choice list with datas from " << databaseTable << "!";
00014     SH_SqlDataModel *sqlDatas = new SH_SqlDataModel();
00015     QStringList fields;
00016     fields << "ID" << m_field;
00017     sqlDatas->fetch(m_table, m_condition, "", fields);
00018     QVariantMap results = sqlDatas->datas();
00019     QVariantList idValues = results.values("ID");
00020     QVariantList fieldsValues = results.values(m_field);
00021     for(int i = 0; i < idValues.length(); i++) {
00022         qDebug() << "new choice " << idValues.at(i).toString() << ":" << fieldsValues.at(i).toString();
00023         m_choices.insert(idValues.at(i).toInt(), fieldsValues.at(i));
00024     }
00025 }
00026
00027 bool SH_DatabaseContentQuestionState::isAnswerValid(const
00028             QVariant &givenAnswer)
00029 {
00030     qDebug() << m_choices.values();
00031     return m_choices.isEmpty() || m_choices.values().contains(givenAnswer);
00032 }
00033
00034 void SH_DatabaseContentQuestionState::setOutput(const QString &
00035             output)
00036 {
00037     SH_QuestionState::setOutput(output);
00038     if(m_choices.size() < 8) {
00039         m_choicesDisplayed = true;
00040         emit displayChoiceList();
00041     }
00042 }
00043
00044 QVariant SH_DatabaseContentQuestionState::rawInput() const
00045 {
00046     return m_choices.key(this->givenAnswer());
00047 }
00048
00049 QMap<int, QVariant> SH_DatabaseContentQuestionState::choiceList(
00050 )
00051 {
00052     if(m_choicesDisplayed) {
00053         return m_choices;
00054     }
00055     return QMap<int,QVariant>();
00056 }
00057
00058 }
```

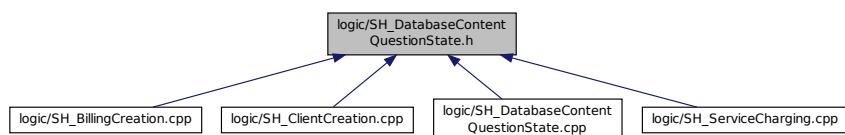
5.23 Référence du fichier logic/SH_DatabaseContentQuestionState.h

```
#include "SH_QuestionState.h"
```

Graphe des dépendances par inclusion de SH_DatabaseContentQuestionState.h :



Ce graphe montre quels fichiers incluent directement ou indirectement ce fichier :



Classes

- class [SH_DatabaseContentQuestionState](#)

5.24 SH_DatabaseContentQuestionState.h

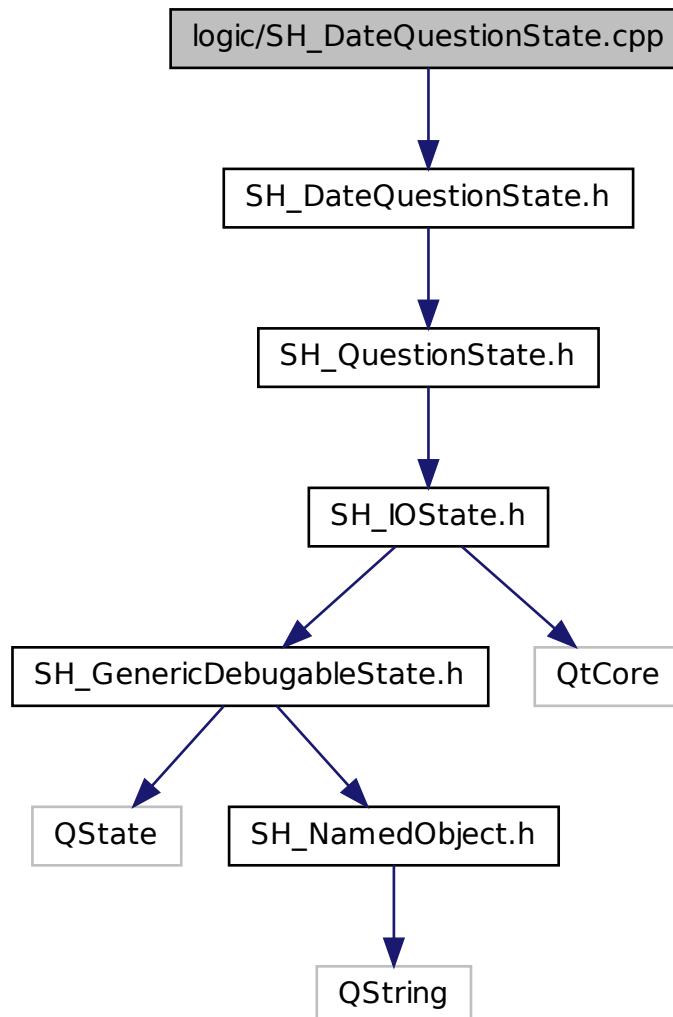
```
00001 #ifndef DATABASECONTENTQUESTIONSTATE_H
```

```
00002 #define DATABASECONTENTQUESTIONSTATE_H
00003 #include "SH_QuestionState.h"
00004
00010 class SH_DatabaseContentQuestionState : public
00011     SH_QuestionState
00012 {
00013     Q_OBJECT
00014 public:
00025     SH_DatabaseContentQuestionState(QString question, QString
00026         name, QString databaseTable, QString tableField, QString databaseCondition = "", 
00027         QState *parent = 0);
00032     virtual bool isAnswerValid(const QVariant &givenAnswer);
00033
00034
00040     void setOutput(const QString &output);
00041
00047     virtual QVariant rawInput() const;
00048
00054     QMap<int, QVariant> choiceList();
00055
00056 signals:
00061     void displayChoiceList();
00062 public slots:
00063
00064 private:
00068     QString m_table;
00072     QString m_condition;
00076     QString m_field;
00080     QMap<int, QVariant> m_choices;
00084     bool m_choicesDisplayed;
00085 };
00086
00087 #endif /* DATABASECONTENTQUESTIONSTATE_H */
```

5.25 Référence du fichier logic/SH_DateQuestionState.cpp

```
#include "SH_DateQuestionState.h"
```

Graphe des dépendances par inclusion de SH_DateQuestionState.cpp :



5.26 SH_DateQuestionState.cpp

```

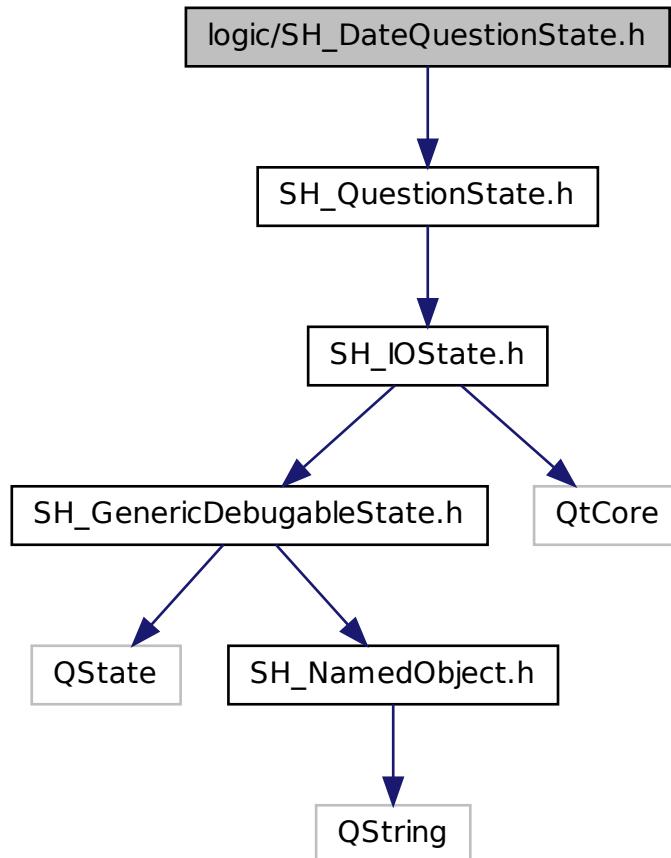
00001 #include "SH_DateQuestionState.h"
00002
00009 SH_DateQuestionState::SH_DateQuestionState(QString question,
00010     QString name, bool past, bool future, QState *parent) :
00011     SH_QuestionState(question + " (au format jj-mm-aaaa)", name, parent), m_past(past),
00012     m_future(future)
00013 {
00014 }
00015
00020 bool SH_DateQuestionState::isValid(const QVariant &givenAnswer)
00021 {
00022     QDate answer = QDate::fromString(givenAnswer.toString(),QString("dd-MM-yyyy"));
00023     if(answer.isValid()) {
00024         qDebug() << "date conforme";
00025         return ((m_future && answer >= QDate::currentDate()) || (m_past && answer <=
00026             QDate::currentDate()));
00027     } else {
00028         return false;
  
```

```
00028      }
00029  }
00030
00037 bool SH_DateQuestionState::getPast() const
00038 {
00039     return m_past;
00040 }
00041
00048 void SH_DateQuestionState::setPast(bool value)
00049 {
00050     m_past = value;
00051 }
00052
00059 bool SH_DateQuestionState::getFuture() const
00060 {
00061     return m_future;
00062 }
00063
00070 void SH_DateQuestionState::setFuture(bool value)
00071 {
00072     m_future = value;
00073 }
00074
00080 QVariant SH_DateQuestionState::rawInput() const
00081 {
00082     return QVariant(input().toDate().toString()); /*TODO set format*/
00083 }
```

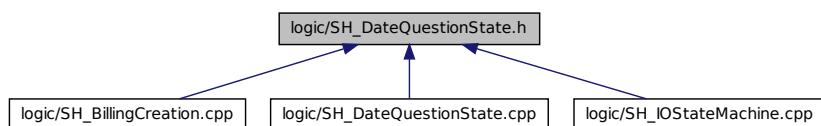
5.27 Référence du fichier logic/SH_DateQuestionState.h

```
#include "SH_QuestionState.h"
```

Graphe des dépendances par inclusion de SH_DateQuestionState.h :



Ce graphe montre quels fichiers incluent directement ou indirectement ce fichier :



Classes

- class [SH_DateQuestionState](#)

5.28 SH_DateQuestionState.h

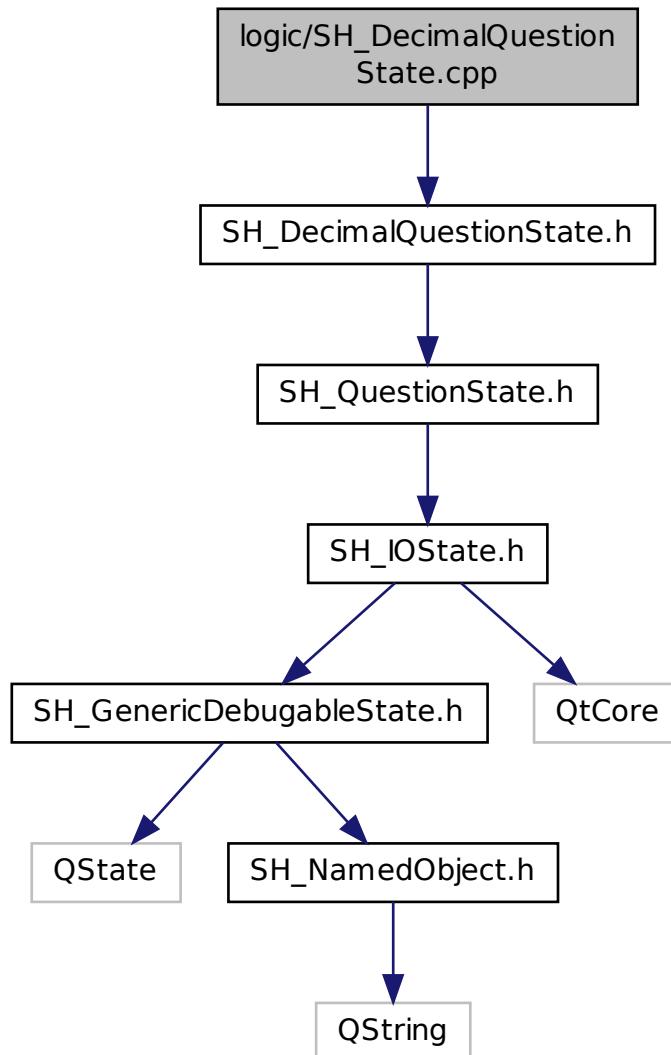
```
00001 #ifndef DATEQUESTIONSTATE_H
```

```
00002 #define DATEQUESTIONSTATE_H
00003 #include "SH_QuestionState.h"
00004
00010 class SH_DateQuestionState : public SH_QuestionState
00011 {
00012     Q_OBJECT
00013 public:
00024     SH_DateQuestionState(QString question, QString name, bool past = true, bool
00025 future = false, QState *parent = 0);
00031 virtual bool isAnswerValid(const QVariant &givenAnswer);
00032
00039 bool getPast() const;
00046 void setPast(bool value);
00047
00054 bool getFuture() const;
00061 void setFuture(bool value);
00062
00068 QVariant rawInput() const;
00069 signals:
00070
00071 public slots:
00072
00073
00074 private:
00078 bool m_past;
00082 bool m_future;
00083 };
00084
00085 #endif /* DATEQUESTIONSTATE_H */
```

5.29 Référence du fichier logic/SH_DecimalQuestionState.cpp

```
#include "SH_DecimalQuestionState.h"
```

Graphe des dépendances par inclusion de SH_DecimalQuestionState.cpp :



5.30 SH_DecimalQuestionState.cpp

```

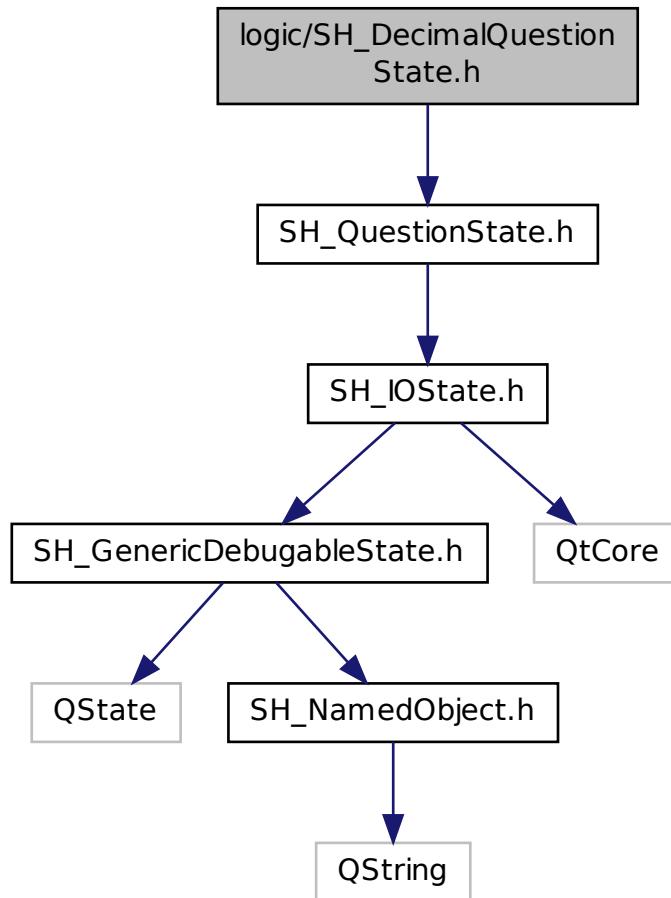
00001 #include "SH_DecimalQuestionState.h"
00002
00003
00010 SH_DecimalQuestionState::SH_DecimalQuestionState(QString
    question, QString name, qreal min, qreal max, QState *parent) :
00011     SH_QuestionState(question, name, parent), m_min(min), m_max(max)
00012 {
00013
00014 }
00015
00022 bool SH_DecimalQuestionState::isValid(const QVariant &
    givenAnswer)
00023 {
00024     bool ok;
00025     qreal answer = givenAnswer.toReal(&ok);
00026     if(ok) {
  
```

```
00027     return ((m_max <= m_min || answer <= m_max) && answer >=
00028         m_min);
00029     } else {
00030         return false;
00031     }
00032 }
00033 qreal SH_DecimalQuestionState::min() const
00034 {
00035     return m_min;
00036 }
00037
00038 void SH_DecimalQuestionState::setMin(const qreal &min)
00039 {
00040     m_min = min;
00041 }
00042
00043 qreal SH_DecimalQuestionState::max() const
00044 {
00045     return m_max;
00046 }
00047
00048 void SH_DecimalQuestionState::setMax(const qreal &max)
00049 {
00050     m_max = max;
00051 }
```

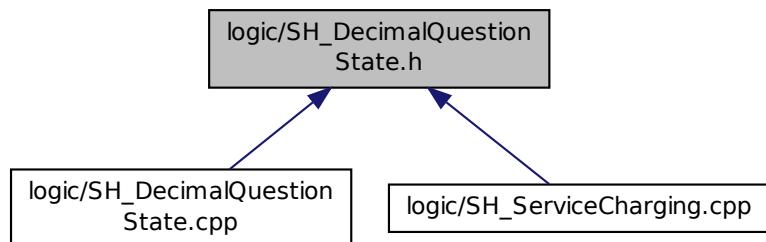
5.31 Référence du fichier logic/SH_DecimalQuestionState.h

```
#include "SH_QuestionState.h"
```

Graphe des dépendances par inclusion de SH_DecimalQuestionState.h :



Ce graphe montre quels fichiers incluent directement ou indirectement ce fichier :



Classes

- class SH_DecimalQuestionState

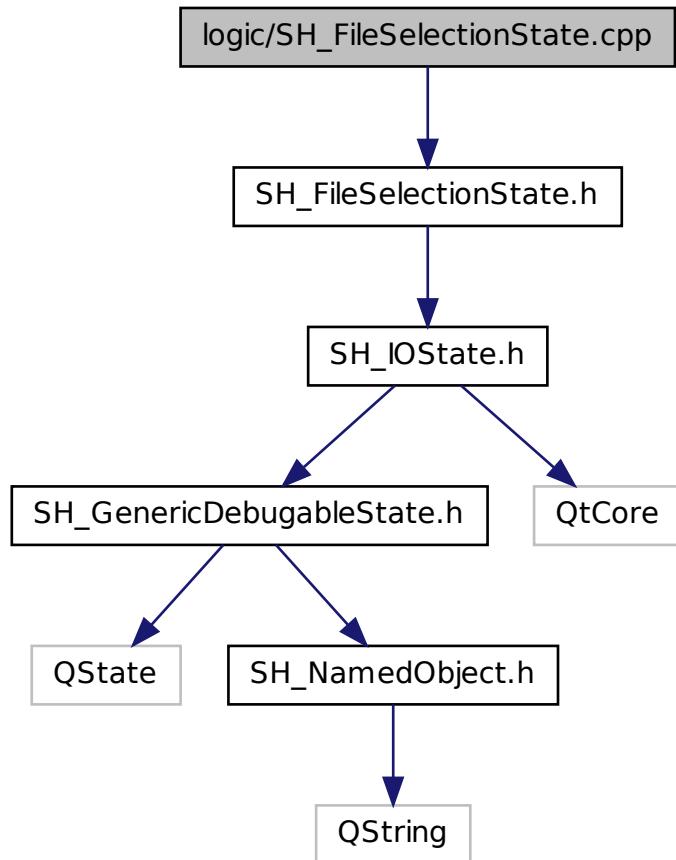
5.32 SH_DecimalQuestionState.h

```
00001 #ifndef DECIMALQUESTIONSTATE_H
00002 #define DECIMALQUESTIONSTATE_H
00003 #include "SH_QuestionState.h"
00004
00011 class SH_DecimalQuestionState : public SH_QuestionState
00012 {
00013     Q_OBJECT
00014 public:
00025     SH_DecimalQuestionState(QString question, QString
00026         name, qreal min=0, qreal max=-1, QState *parent = 0);
00032     virtual bool isAnswerValid(const QVariant &givenAnswer);
00033
00040     qreal min() const;
00047     void setMin(const qreal &min);
00048
00055     qreal max() const;
00062     void setMax(const qreal &max);
00063
00064 signals:
00065
00066 public slots:
00067
00068 private:
00072     qreal m_min;
00076     qreal m_max;
00077 };
00078 #endif /* DECIMALQUESTIONSTATE_H */
```

5.33 Référence du fichier logic/SH_FileSelectionState.cpp

```
#include "SH_FileSelectionState.h"
```

Graphe des dépendances par inclusion de SH_FileSelectionState.cpp :



5.34 SH_FileSelectionState.cpp

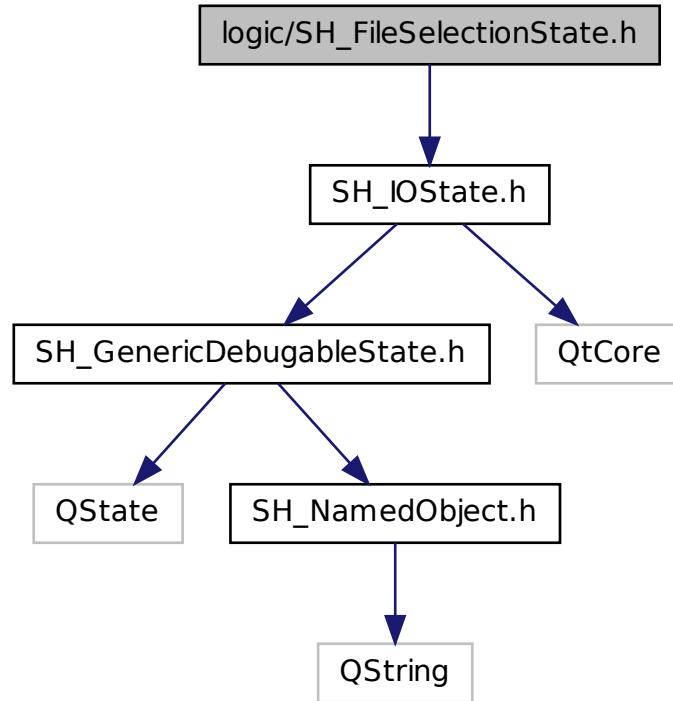
```

00001 #include "SH_FileSelectionState.h"
00002
00008 SH_FileSelectionState::SH_FileSelectionState(QString output,
    QString name, QState *parent) :
00009     SH_InOutState(output, name, parent)
00010 {
00011
00012 }
  
```

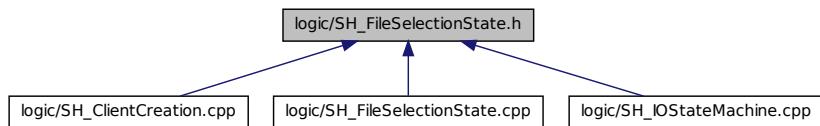
5.35 Référence du fichier logic/SH_FileSelectionState.h

```
#include "SH_IOState.h"
```

Graphe des dépendances par inclusion de SH_FileSelectionState.h :



Ce graphe montre quels fichiers incluent directement ou indirectement ce fichier :



Classes

- class `SH_FileSelectionState`
The `SH_FileSelectionState` class.

5.36 SH_FileSelectionState.h

```

00001 #ifndef FILESELECTIONSTATE_H
00002 #define FILESELECTIONSTATE_H
00003 #include "SH_IOState.h"
00007 class SH_FileSelectionState : public SH_InOutState
00008 {
00009     Q_OBJECT
0010 public:
  
```

```

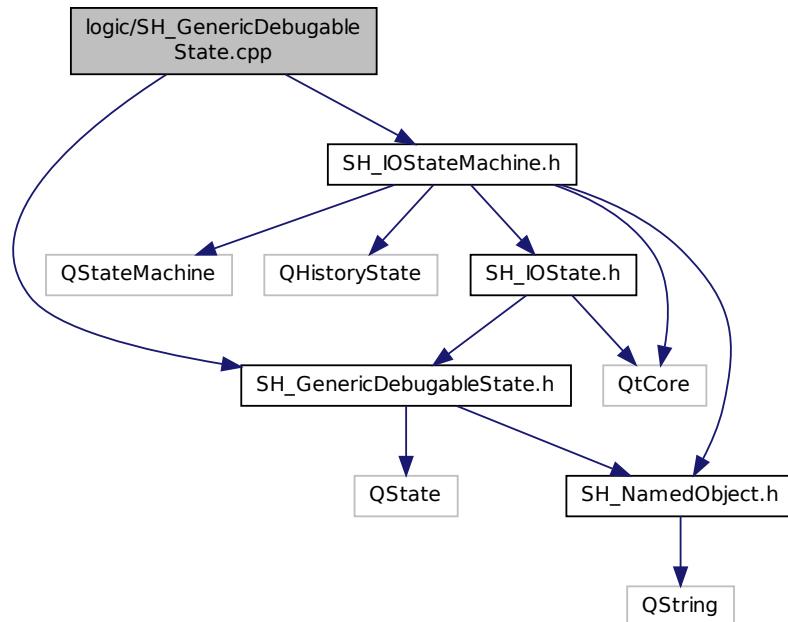
00018     SH_FileSelectionState(QString output, QString
00019         name, QState *parent = 0);
00020 signals:
00021 public slots:
00022 };
00023
00024 #endif /* FILESELECTIONSTATE_H */

```

5.37 Référence du fichier logic/SH_GenericDebugableState.cpp

```
#include "SH_GenericDebugableState.h"
#include "SH_IOSStateMachine.h"
```

Graphe des dépendances par inclusion de SH_GenericDebugableState.cpp :



5.38 SH_GenericDebugableState.cpp

```

00001 #include "SH_GenericDebugableState.h"
00002 #include "SH_IOSStateMachine.h"
00003
00010 SH_GenericState::SH_GenericState(QString name,
00011     QState *parent) :
00012     QState(parent), SH_NamedObject(name)
00013 {
00014 }
00021 QString SH_GenericState::toString()
00022 {
00023     QStateMachine* machine = this->machine();
00024     SH_InOutStateMachine* mach = qobject_cast<
00025         SH_InOutStateMachine *>(machine);
00026     if(mach) {
00027         return SH_NamedObject::toString() + " [in "+mach->
00028             toString()+"] ";
00029     } else {
00030         return SH_NamedObject::toString();
00031     }

```

```

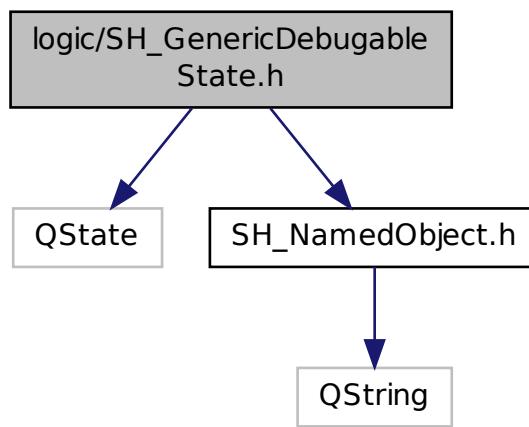
00029      }
00030  }
00031
00036 void SH_GenericState::onTransitionTriggered()
00037 {
00038     QAbstractTransition* tr = qobject_cast<QAbstractTransition*>(sender());
00039     if (tr == 0) return;
00040
00041     SH_GenericState* sourceState = qobject_cast<SH_GenericState*>(tr->
00042         sourceState());
00042     SH_GenericState* targetState = qobject_cast<SH_GenericState*>(tr->
00043         targetState());
00043
00044     QString log;
00045     QTextStream logStream(&log);
00046     logStream << machine()->objectName() << " transition from ";
00047     if (sourceState) logStream << sourceState->name();
00048     else logStream << tr->sourceState();
00049     logStream << " to ";
00050     if (targetState) logStream << targetState->name();
00051     else logStream << tr->targetState();
00052     logStream.flush();
00053     qDebug() << "Machine: " << log;
00054 }
00055
00056
00062 void SH_GenericState::onEntry(QEvent *event)
00063 {
00064     Q_UNUSED(event);
00065     qDebug() << "Machine: " << machine()->objectName() << " entered " << name();
00066 }
00067
00073 void SH_GenericState::onExit(QEvent *event)
00074 {
00075     Q_UNUSED(event);
00076     qDebug() << "Machine: " << machine()->objectName() << " exited " << name();
00077 }
00083 void SH_GenericState::onMachineStarted()
00084 {
00085     foreach (QAbstractTransition* tr, transitions())
00086         connect(tr, SIGNAL(triggered()), this, SLOT(onTransitionTriggered()));
00087 }

```

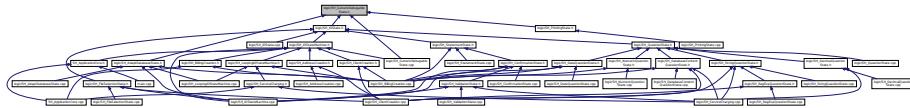
5.39 Référence du fichier logic/SH_GenericDebugableState.h

```
#include <QState>
#include "SH_NamedObject.h"
```

Graphe des dépendances par inclusion de SH_GenericDebugableState.h :



Ce graphe montre quels fichiers incluent directement ou indirectement ce fichier :



Classes

- class [SH_GenericState](#)

5.40 SH_GenericDebugableState.h

```

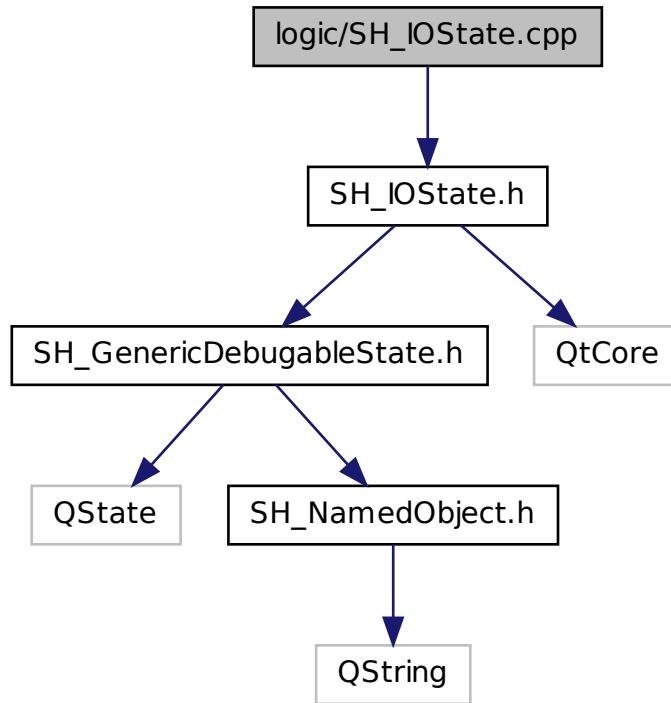
00001 #ifndef GENERICSTATE_H
00002 #define GENERICSTATE_H
00003
00004 #include <QState>
00005 #include "SH_NamedObject.h"
00006
00012 class SH_GenericState : public QState, SH_NamedObject
00013 {
00014     Q_OBJECT
00015 public:
00023     SH_GenericState(QString name="", QState *parent = 0);
00030     QString toString();
00031
00032 signals:
00039     void next();
00040
00041 protected:
00047     void onEntry(QEvent *event);
00053     void onExit(QEvent *event);
00054
00055
00056 private:
00057
00058 private slots:
00063     void onMachineStarted();
00068     void onTransitionTriggered();
00069
00070 };
00071
00072 #endif /* GENERICSTATE_H*/

```

5.41 Référence du fichier logic/SH_IOSState.cpp

```
#include "SH_IOSState.h"
```

Graphe des dépendances par inclusion de SH_IOState.cpp :



5.42 SH_IOState.cpp

```

00001 #include "SH_IOState.h"
00002
00009 SH_InOutState::SH_InOutState(QString output, QString name,
00010     QState *parent) :
00011     SH_GenericState(name, parent), m_output(output), m_isVisible(true)
00012 {
00013 }
00014
00020 QVariant SH_InOutState::input() const
00021 {
00022     return m_input;
00023 }
00024
00030 QVariant SH_InOutState::rawInput() const
00031 {
00032     return input();
00033 }
00034
00041 void SH_InOutState::setInput(const QVariant &input)
00042 {
00043     qDebug() << "new input " << input.toString();
00044     m_input = input;
00045     if(m_isVisible) {
00046         emit resendInput(m_input);
00047     }
00048 }
00049
00056 QString SH_InOutState::output() const
00057 {
00058     return m_output;
00059 }
00060
00061
  
```

```

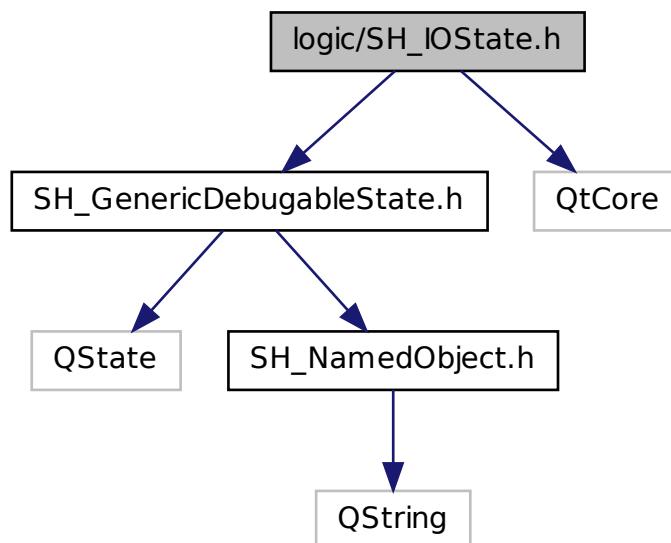
00068 void SH_InOutState::setOutput(const QString &output)
00069 {
00070     m_output = output;
00071     if(m_isVisible) {
00072         emit sendOutput(QVariant(m_output));
00073     }
00074 }
00075
00081 void SH_InOutState::setVisibility(bool isVisible)
00082 {
00083     m_isVisible = isVisible;
00084 }
00085
00091 bool SH_InOutState::visibility() {
00092     return m_isVisible;
00093 }
00094
00095 void SH_InOutState::display(bool canDisplay)
00096 {
00097     m_display=canDisplay;
00098     if(m_display && !m_output.isEmpty() && m_isVisible) {
00099         qDebug() << "resalut !" << QVariant(m_output);
00100         emit sendOutput(QVariant(m_output));
00101     }
00102 }
00103
00110 void SH_InOutState::onExit(QEvent *event)
00111 {
00112     SH_GenericState::onExit(event);
00113     if(m_isVisible) {
00114         emit resendInput(m_input);
00115     }
00116 }

```

5.43 Référence du fichier logic/SH_IOState.h

```
#include "SH_GenericDebugableState.h"
#include <QtCore>
```

Graphe des dépendances par inclusion de SH_IOState.h :



Ce graphe montre quels fichiers incluent directement ou indirectement ce fichier :



Classes

- class [SH_InOutState](#)

5.44 SH_IOState.h

```

00001 #ifndef IOSTATE_H
00002 #define IOSTATE_H
00003 #include "SH_GenericDebugableState.h"
00004 #include <QtCore>
00005
00011 class SH_InOutState : public SH_GenericState
00012 {
00013     Q_OBJECT
00014 public:
00023     SH_InOutState(QString output, QString name, QState *parent = 0);
00024
00031     virtual QVariant input() const;
00032
00039     virtual QVariant rawInput() const;
00040
00047     virtual QString output() const;
00048
00049
00056     void onExit(QEvent *event);
00057
00063     bool visibility();
00064
00070     void display(bool canDisplay);
00071
00072 signals:
00079     void sendOutput(QVariant output);
00086     void resendInput(QVariant input);
00087
00088 public slots:
00095     virtual void setInput(const QVariant &input);
00102     virtual void setOutput(const QString &output);
00103
00104
00110     virtual void setVisibility(bool isVisible);
00111
00112 private:
00116     QVariant m_input;
00120     QString m_output;
00124     bool m_isVisible;
00128     bool m_display;
00129 };
00130
00131 #endif /* IOSTATE_H*/

```

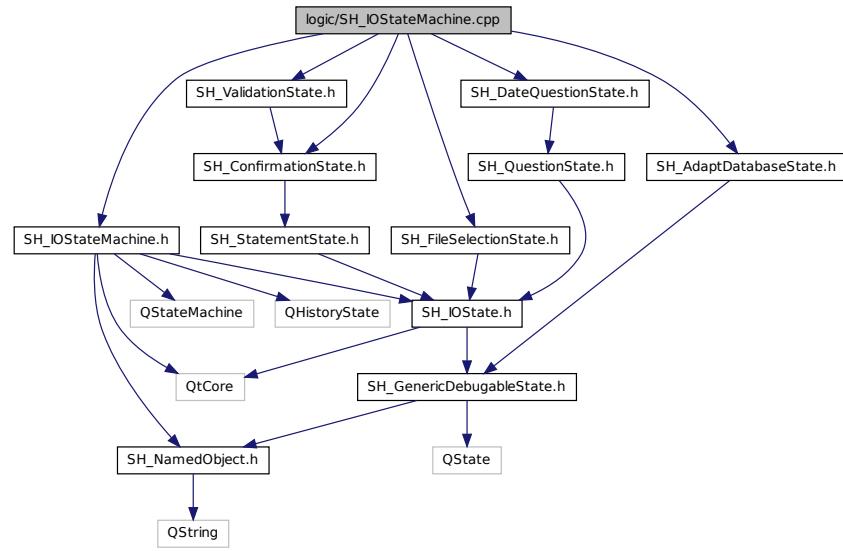
5.45 Référence du fichier logic/SH_IOSMachine.cpp

```

#include "SH_IOSMachine.h"
#include "SH_ValidationState.h"
#include "SH_ConfirmationState.h"
#include "SH_FileSelectionState.h"
#include "SH_DateQuestionState.h"
#include "SH_AdaptDatabaseState.h"

```

Graphe des dépendances par inclusion de SH_IOStateMachine.cpp :



5.46 SH_IStateMachine.cpp

```
00001 #include "SH_IStateMachine.h"
00002 #include "SH_ValidationState.h"
00003 #include "SH_ConfirmationState.h"
00004 #include "SH_FileSelectionState.h"
00005 #include "SH_DateQuestionState.h"
00006 #include "SH_AdaptDatabaseState.h"
00007
00014 SH_InOutStateMachine::SH_InOutStateMachine(QString tableName,
00015     QString name, QObject *parent) :
00016     QStateMachine(parent), SH_NamedObject(name), m_tableName(tableName)
00017 {
00018     qDebug() << "nouvelle IStateMachine";
00019
00026 QString SH_InOutStateMachine::toString()
00027 {
00028     QObject* parent = this->parent();
00029     SH_GenericState* par = qobject_cast<SH_GenericState *>(parent);
00030     if(par) {
00031         return SH_NamedObject::toString() + " [descending from "+par->
00032             toString()+" ] ";
00033     } else {
00034         return SH_NamedObject::toString();
00035     }
00036
00043 QVariantMap SH_InOutStateMachine::ioContent() const
00044 {
00045     return m_ioContent;
00046 }
00047
00054 void SH_InOutStateMachine::setIOcontent(const QVariantMap &ioContent)
00055 {
00056     m_ioContent = ioContent;
00057 }
00058
00065 QVariant SH_InOutStateMachine::getContentValue(QString field)
00066 {
00067     return m_ioContent.value(field);
00068 }
00069
00076 QString SH_InOutStateMachine::tableName() const
00077 {
00078     return m_tableName;
00079 }
00080
```

```

00087 void SH_InOutStateMachine::setTableName(const QString &tableName)
00088 {
00089     m_tableName = tableName;
00090 }
00091
00092
00093 void SH_InOutStateMachine::setContentValue(QVariant content, QString
00094     field)
00095 {
00096     m_ioContent.insert(field, content);
00097 }
00098
00099 void SH_InOutStateMachine::addIOState(
00100     SH_InOutState *state, QString field)
00101 {
00102     /*à faire au moment de l'entrée dans l'état state*/
00103     connect(state, &QState::entered, [=]() {
00104         qDebug() << "entered !";
00105         state->display(true);
00106         connect(this, &SH_InOutStateMachine::receiveInput, state, &
00107             SH_InOutState::setInput); /* la réception d'une valeur entraîne son enregistrement
00108             comme entrée de l'utilisateur auprès de l'état*/
00109         connect(this, &SH_InOutStateMachine::receiveInput, [=](QString in
00110             ){ qDebug() << "hello world !"; state->setInput(in);}); /* la réception d'une valeur entraîne son
00111             enregistrement comme entrée de l'utilisateur auprès de l'état*/
00112         connect(state, &SH_InOutState::sendOutput, [=](QVariant out) {qDebug() <<
00113             "connected !"; emit this->sendText(out.toString(), false);});
00114         connect(state, &SH_InOutState::resendInput, [=](QVariant in) {emit this->
00115             resendText(in.toString(), true);});
00116         if(state->visibility()) {
00117             state->sendOutput(QVariant(state->output()));
00118         } else {
00119             qDebug() << "invisible";
00120         }
00121     });
00122     SH_ValidationState *validationState = qobject_cast<
00123         SH_ValidationState*>(state);
00124     if(validationState) {
00125         /*à faire au moment de l'entrée dans l'état state*/
00126         connect(validationState, &QState::entered, [=]() {
00127             connect(this, &SH_InOutStateMachine::validateInput,
00128                 validationState, &SH_ValidationState::confirmInput);
00129         });
00130     }
00131     SH_ConfirmationState *confirmationState = qobject_cast<
00132         SH_ConfirmationState*>(state);
00133     if(confirmationState) {
00134         /*à faire au moment de l'entrée dans l'état state*/
00135         connect(confirmationState, &QState::entered, [=]() {
00136             connect(this, &SH_InOutStateMachine::validateInput,
00137                 confirmationState, &SH_ConfirmationState::confirmInput);
00138         });
00139     }
00140     SH_DateQuestionState *dateState = qobject_cast<
00141         SH_DateQuestionState*>(state);
00142     if(dateState) {
00143         /*à faire au moment de l'entrée dans l'état state*/
00144         connect(dateState, &QState::entered, this, &
00145             SH_InOutStateMachine::displayCalendar);
00146     }
00147     SH_FileSelectionState *fileState = qobject_cast<
00148         SH_FileSelectionState*>(state);
00149     if(fileState) {
00150         /*à faire au moment de l'entrée dans l'état state*/
00151         connect(fileState, &QState::entered, this, &
00152             SH_InOutStateMachine::displayFileDialog);
00153     }
00154     /*à faire au moment de la sortie de l'état state*/
00155     connect(state, &QState::exited, [=]() {
00156         qDebug() << "exited !";
00157         if(!field.isEmpty()) {
00158             setContentValue(state->rawInput(), field);
00159             /*gestion de l'historique des états pour pouvoir revenir à l'état state après l'avoir quitté*/
00160             QHistoryState* hState = new QHistoryState(state);
00161             setIOStateHistory(hState, field);
00162         }
00163         state->disconnect(this); /*plus aucune action sur l'état ne pourra être provoquée par la machine*/
00164     });
00165     QAbstractState* astate = qobject_cast<QAbstractState *>(state);
00166     if(astate) {
00167         addState(astate);
00168     }
00169 }
00170
00171 void SH_InOutStateMachine::addIOStateMachine(

```

```

00176     SH_InOutStateMachine *fsm)
00177     /*à faire au moment de l'entrée dans la machine d'état fsm*/
00178     connect(fsm, &QState::entered, [=]() {
00179         connect(this, &SH_InOutStateMachine::receiveInput, fsm, &
00180             SH_InOutStateMachine::receiveInput);
00181         connect(this, &SH_InOutStateMachine::sendText, fsm, &
00182             SH_InOutStateMachine::sendText);
00183         connect(this, &SH_InOutStateMachine::resendText, fsm, &
00184             SH_InOutStateMachine::resendText);
00185         connect(this, &SH_InOutStateMachine::confirmInput, fsm, &
00186             SH_InOutStateMachine::confirmInput);
00187         connect(this, &SH_InOutStateMachine::validateInput, fsm, &
00188             SH_InOutStateMachine::validateInput);
00189         connect(this, &SH_InOutStateMachine::replaceInput, fsm, &
00190             SH_InOutStateMachine::replaceInput);
00191         connect(this, &SH_InOutStateMachine::cancelReplacement, fsm,
00192             &SH_InOutStateMachine::cancelReplacement);
00193         connect(this, &SH_InOutStateMachine::displayCalendar, fsm, &
00194             SH_InOutStateMachine::displayCalendar);
00195     });
00196     /*à faire au moment de la sortie de la machine d'état fsm*/
00197     connect(fsm, &QState::exited, [=]() {
00198         fsm->disconnect(this); /*plus aucune action sur la machine d'état fille ne pourra être provoquée
00199         par la machine mère*/
00200     });
00201
00202 QMap<QString, QHistoryState *> SH_InOutStateMachine::ioStatesHistory()
00203 {
00204     const
00205     return m_ioStatesHistory;
00206 }
00207
00208
00209
00210 void SH_InOutStateMachine::setIOStatesHistory(const QMap<QString,
00211     QHistoryState *> &ioStatesHistory)
00212 {
00213     m_ioStatesHistory = ioStatesHistory;
00214 }
00215
00216
00217
00218
00219
00220 void SH_InOutStateMachine::setIOStateHistory(QHistoryState *state,
00221     QString field)
00222 {
00223     m_ioStatesHistory.insert(field, state); /*remplacement si plusieurs fois*/
00224 }
00225
00226
00227
00228
00229
00230
00231
00232 QHistoryState *SH_InOutStateMachine::historyValue(QString field)
00233 {
00234     return m_ioStatesHistory.value(field);
00235 }
00236
00237
00238
00239
00240
00241
00242
00243
00244 void SH_InOutStateMachine::addChildrenNextTransition(
00245     QAbstractState *previousState, QAbstractState *nextState)
00246 {
00247     SH_InOutStateMachine* fsmPreviousState = qobject_cast<
00248         SH_InOutStateMachine*>(previousState);
00249     SH_GenericState* genPreviousState = qobject_cast<
00250         SH_GenericState*>(previousState);
00251     QFinalState* final = qobject_cast<QFinalState*>(nextState);
00252     if(final) {
00253         SH_AdaptDatabaseState* saveState = new
00254             SH_AdaptDatabaseState("enregistrement de la machine "+
00255             toString());
00256         if(genPreviousState) {
00257             genPreviousState->addTransition(genPreviousState, SIGNAL(next()), saveState);
00258         }
00259         if(fsmPreviousState) {
00260             fsmPreviousState->addTransition(fsmPreviousState, SIGNAL(next()), saveState);
00261         }
00262         if(genPreviousState || fsmPreviousState) {
00263             connect(previousState, &QAbstractState::exited, [=]() {
00264                 connect(saveState, &QAbstractState::entered, [=]() {
00265                     emit this->sendText("Merci !");
00266                     setContentValue(saveState->insertUpdate(
00267                         m_tableName, m_ioContent), "ID");
00268                     emit this->clearAll();
00269                 });
00270             });
00271             saveState->addTransition(saveState, SIGNAL(next()), final);
00272         }
00273     } else {
00274
00275
00276
00277
00278
00279
00280
00281
00282
00283
00284
00285
00286
00287
00288
00289
00290
00291
00292
00293
00294
00295
00296
00297
00298
00299
00300
00301
00302
00303
00304
00305
00306
00307
00308
00309
00310
00311
00312
00313
00314
00315
00316
00317
00318
00319
00320
00321
00322
00323
00324
00325
00326
00327
00328
00329
00330
00331
00332
00333
00334
00335
00336
00337
00338
00339
00340
00341
00342
00343
00344
00345
00346
00347
00348
00349
00350
00351
00352
00353
00354
00355
00356
00357
00358
00359
00360
00361
00362
00363
00364
00365
00366
00367
00368
00369
00370
00371
00372
00373
00374
00375
00376
00377
00378
00379
00380
00381
00382
00383
00384
00385
00386
00387
00388
00389
00390
00391
00392
00393
00394
00395
00396
00397
00398
00399
00400
00401
00402
00403
00404
00405
00406
00407
00408
00409
00410
00411
00412
00413
00414
00415
00416
00417
00418
00419
00420
00421
00422
00423
00424
00425
00426
00427
00428
00429
00430
00431
00432
00433
00434
00435
00436
00437
00438
00439
00440
00441
00442
00443
00444
00445
00446
00447
00448
00449
00450
00451
00452
00453
00454
00455
00456
00457
00458
00459
00460
00461
00462
00463
00464
00465
00466
00467
00468
00469
00470
00471
00472
00473
00474
00475
00476
00477
00478
00479
00480
00481
00482
00483
00484
00485
00486
00487
00488
00489
00490
00491
00492
00493
00494
00495
00496
00497
00498
00499
00500
00501
00502
00503
00504
00505
00506
00507
00508
00509
00510
00511
00512
00513
00514
00515
00516
00517
00518
00519
00520
00521
00522
00523
00524
00525
00526
00527
00528
00529
00530
00531
00532
00533
00534
00535
00536
00537
00538
00539
00540
00541
00542
00543
00544
00545
00546
00547
00548
00549
00550
00551
00552
00553
00554
00555
00556
00557
00558
00559
00560
00561
00562
00563
00564
00565
00566
00567
00568
00569
00570
00571
00572
00573
00574
00575
00576
00577
00578
00579
00580
00581
00582
00583
00584
00585
00586
00587
00588
00589
00590
00591
00592
00593
00594
00595
00596
00597
00598
00599
00600
00601
00602
00603
00604
00605
00606
00607
00608
00609
00610
00611
00612
00613
00614
00615
00616
00617
00618
00619
00620
00621
00622
00623
00624
00625
00626
00627
00628
00629
00630
00631
00632
00633
00634
00635
00636
00637
00638
00639
00640
00641
00642
00643
00644
00645
00646
00647
00648
00649
00650
00651
00652
00653
00654
00655
00656
00657
00658
00659
00660
00661
00662
00663
00664
00665
00666
00667
00668
00669
00670
00671
00672
00673
00674
00675
00676
00677
00678
00679
00680
00681
00682
00683
00684
00685
00686
00687
00688
00689
00690
00691
00692
00693
00694
00695
00696
00697
00698
00699
00700
00701
00702
00703
00704
00705
00706
00707
00708
00709
00710
00711
00712
00713
00714
00715
00716
00717
00718
00719
00720
00721
00722
00723
00724
00725
00726
00727
00728
00729
00730
00731
00732
00733
00734
00735
00736
00737
00738
00739
00740
00741
00742
00743
00744
00745
00746
00747
00748
00749
00750
00751
00752
00753
00754
00755
00756
00757
00758
00759
00760
00761
00762
00763
00764
00765
00766
00767
00768
00769
00770
00771
00772
00773
00774
00775
00776
00777
00778
00779
00780
00781
00782
00783
00784
00785
00786
00787
00788
00789
00790
00791
00792
00793
00794
00795
00796
00797
00798
00799
00800
00801
00802
00803
00804
00805
00806
00807
00808
00809
00810
00811
00812
00813
00814
00815
00816
00817
00818
00819
00820
00821
00822
00823
00824
00825
00826
00827
00828
00829
00830
00831
00832
00833
00834
00835
00836
00837
00838
00839
00840
00841
00842
00843
00844
00845
00846
00847
00848
00849
00850
00851
00852
00853
00854
00855
00856
00857
00858
00859
00860
00861
00862
00863
00864
00865
00866
00867
00868
00869
00870
00871
00872
00873
00874
00875
00876
00877
00878
00879
00880
00881
00882
00883
00884
00885
00886
00887
00888
00889
00890
00891
00892
00893
00894
00895
00896
00897
00898
00899
00900
00901
00902
00903
00904
00905
00906
00907
00908
00909
00910
00911
00912
00913
00914
00915
00916
00917
00918
00919
00920
00921
00922
00923
00924
00925
00926
00927
00928
00929
00930
00931
00932
00933
00934
00935
00936
00937
00938
00939
00940
00941
00942
00943
00944
00945
00946
00947
00948
00949
00950
00951
00952
00953
00954
00955
00956
00957
00958
00959
00960
00961
00962
00963
00964
00965
00966
00967
00968
00969
00970
00971
00972
00973
00974
00975
00976
00977
00978
00979
00980
00981
00982
00983
00984
00985
00986
00987
00988
00989
00990
00991
00992
00993
00994
00995
00996
00997
00998
00999
01000
01001
01002
01003
01004
01005
01006
01007
01008
01009
01010
01011
01012
01013
01014
01015
01016
01017
01018
01019
01020
01021
01022
01023
01024
01025
01026
01027
01028
01029
01030
01031
01032
01033
01034
01035
01036
01037
01038
01039
01040
01041
01042
01043
01044
01045
01046
01047
01048
01049
01050
01051
01052
01053
01054
01055
01056
01057
01058
01059
01060
01061
01062
01063
01064
01065
01066
01067
01068
01069
01070
01071
01072
01073
01074
01075
01076
01077
01078
01079
01080
01081
01082
01083
01084
01085
01086
01087
01088
01089
01090
01091
01092
01093
01094
01095
01096
01097
01098
01099
01100
01101
01102
01103
01104
01105
01106
01107
01108
01109
01110
01111
01112
01113
01114
01115
01116
01117
01118
01119
01120
01121
01122
01123
01124
01125
01126
01127
01128
01129
01130
01131
01132
01133
01134
01135
01136
01137
01138
01139
01140
01141
01142
01143
01144
01145
01146
01147
01148
01149
01150
01151
01152
01153
01154
01155
01156
01157
01158
01159
01160
01161
01162
01163
01164
01165
01166
01167
01168
01169
01170
01171
01172
01173
01174
01175
01176
01177
01178
01179
01180
01181
01182
01183
01184
01185
01186
01187
01188
01189
01190
01191
01192
01193
01194
01195
01196
01197
01198
01199
01200
01201
01202
01203
01204
01205
01206
01207
01208
01209
01210
01211
01212
01213
01214
01215
01216
01217
01218
01219
01220
01221
01222
01223
01224
01225
01226
01227
01228
01229
01230
01231
01232
01233
01234
01235
01236
01237
01238
01239
01240
01241
01242
01243
01244
01245
01246
01247
01248
01249
01250
01251
01252
01253
01254
01255
01256
01257
01258
01259
01260
01261
01262
01263
01264
01265
01266
01267
01268
01269
01270
01271
01272
01273
01274
01275
01276
01277
01278
01279
01280
01281
01282
01283
01284
01285
01286
01287
01288
01289
01290
01291
01292
01293
01294
01295
01296
01297
01298
01299
01300
01301
01302
01303
01304
01305
01306
01307
01308
01309
01310
01311
01312
01313
01314
01315
01316
01317
01318
01319
01320
01321
01322
01323
01324
01325
01326
01327
01328
01329
01330
01331
01332
01333
01334
01335
01336
01337
01338
01339
01340
01341
01342
01343
01344
01345
01346
01347
01348
01349
01350
01351
01352
01353
01354
01355
01356
01357
01358
01359
01360
01361
01362
01363
01364
01365
01366
01367
01368
01369
01370
01371
01372
01373
01374
01375
01376
01377
01378
01379
01380
01381
01382
01383
01384
01385
01386
01387
01388
01389
01390
01391
01392
01393
01394
01395
01396
01397
01398
01399
01400
01401
01402
01403
01404
01405
01406
01407
01408
01409
01410
01411
01412
01413
01414
01415
01416
01417
01418
01419
01420
01421
01422
01423
01424
01425
01426
01427
01428
01429
01430
01431
01432
01433
01434
01435
01436
01437
01438
01439
01440
01441
01442
01443
01444
01445
01446
01447
01448
01449
01450
01451
01452
01453
01454
01455
01456
01457
01458
01459
01460
01461
01462
01463
01464
01465
01466
01467
01468
01469
01470
01471
01472
01473
01474
01475
01476
01477
01478
01479
01480
01481
01482
01483
01484
01485
01486
01487
01488
01489
01490
01491
01492
01493
01494
01495
01496
01497
01498
01499
01500
01501
01502
01503
01504
01505
01506
01507
01508
01509
01510
01511
01512
01513
01514
01515
01516
01517
01518
01519
01520
01521
01522
01523
01524
01525
01526
01527
01528
01529
01530
01531
01532
01533
01534
01535
01536
01537
01538
01539
01540
01541
01542
01543
01544
01545
01546
01547
01548
01549
01550
01551
01552
01553
01554
01555
01556
01557
01558
01559
01560
01561
01562
01563
01564
01565
01566
01567
01568
01569
01570
01571
01572
01573
01574
01575
01576
01577
01578
01579
01580
01581
01582
01583
01584
01585
01586
01587
01588
01589
01590
01591
01592
01593
01594
01595
01596
01597
01598
01599
01600
01601
01602
01603
01604
01605
01606
01607
01608
01609
01610
01611
01612
01613
01614
01615
01616
01617
01618
01619
01620
01621
01622
01623
01624
01625
01626
01627
01628
01629
01630
01631
01632
01633
01634
01635
01636
01637
01638
01639
01640
01641
01642
01643
01644
01645
01646
01647
01648
01649
01650
01651
01652
01653
01654
01655
01656
01657
01658
01659
01660
01661
01662
01663
01664
01665
01666
01667
01668
01669
01670
01671
01672
01673
01674
01675
01676
01677
01678
01679
01680
01681
01682
01683
01684
01685
01686
01687
01688
01689
01690
01691
01692
01693
01694
01695
01696
01697
01698
01699
01700
01701
01702
01703
01704
01705
01706
01707
01708
01709
01710
01711
01712
01713
01714
01715
01716
01717
01718
01719
01720
01721
01722
01723
01724
01725
01726
01727
01728
01729
01730
01731
01732
01733
01734
01735
01736
01737
01738
01739
01740
01741
01742
01743
01744
01745
01746
01747
01748
01749
01750
01751
01752
01753
01754
01755
01756
01757
01758
01759
01760
01761
01762
01763
01764
01765
01766
01767
01768
01769
01770
01771
01772
01773
01774
01775
01776
01777
01778
01779
01780
01781
01782
01783
01784
01785
01786
01787
01788
01789
01790
01791
01792
01793
01794
01795
01796
01797
01798
01799
01800
01801
01802
01803
01804
01805
01806
01807
01808
01809
01810
01811
01812
01813
01814
01815
01816
01817
01818
01819
01820
01821
01822
01823
01824
01825
01826
01827
01828
01829
01830
01831
01832
01833
01834
01835
01836
01837
01838
01839
01840
01841
01842
01843
01844
01845
01846
01847
01848
01849
01850
01851
01852
01853
01854
01855
01856
01857
01858
01859
01860
01861
01862
01863
01864
01865
01866
01867
01868
01869
01870
01871
01872
01873
01874
01875
01876
01877
01878
01879
01880
01881
01882
01883
01884
01885
01886
01887
01888
01889
01890
01891
01892
01893
01894
01895
01896
01897
01898
01899
01900
01901
01902
01903

```

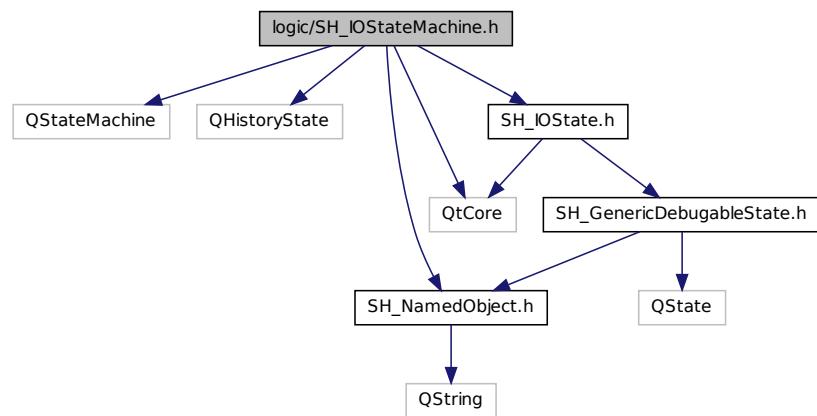
```

00274     if(genPreviousState) {
00275         qDebug() << "next transition between " << genPreviousState->toString() << " and " <<
00276         nextState;
00277         genPreviousState->addTransition(genPreviousState, SIGNAL(next()), nextState);
00278     }
00279     if(fsmPreviousState) {
00280         qDebug() << "next transition between " << fsmPreviousState->toString() << " and " <<
00281         nextState;
00282         fsmPreviousState->addTransition(fsmPreviousState, SIGNAL(next()), nextState);
00283     }
00284     if(genPreviousState) {
00285         /*à faire au moment de l'entrée dans l'état previousState*/
00286         connect(genPreviousState, &QAbstractState::entered, [=]() {
00287             connect(this, &SH_InOutStateMachine::replaceInput, [=](
00288                 QString field) {
00289                 /*après avoir demandé à revenir sur un état précédent, on attend la fin de l'état actuel
00290                 puis on retourne à l'historique de l'état désiré; celui-ci fini, on passe à l'état qui aurait suivi celui
00291                 pendant lequel on a demandé à revenir sur un état précédent*/
00292                 QHistoryState* hState = historyValue(field);
00293                 if(hState) { /*si l'historique existe (on a déjà quitté l'état voulu)*/
00294                     hState->parentState()->addTransition(hState->parentState(), SIGNAL(
00295                         next()), nextState);
00296                     genPreviousState->addTransition(genPreviousState, SIGNAL(
00297                         next()), hState);
00298                 }
00299             });
00300         });
00301     }
00302 }
00303 }
```

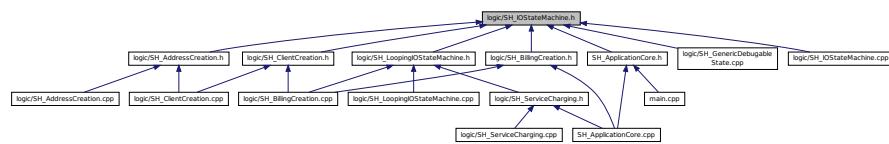
5.47 Référence du fichier logic/SH_IOStateMachine.h

```
#include <QStateMachine>
#include <QHistoryState>
#include <QtCore>
#include "SH_NamedObject.h"
#include "SH_IOSstate.h"
```

Graphe des dépendances par inclusion de SH_IOStateMachine.h :



Ce graphe montre quels fichiers incluent directement ou indirectement ce fichier :



Classes

- class [SH_InOutStateMachine](#)

5.48 SH_IStateMachine.h

```

00001 #ifndef IOSTATEMACHINE_H
00002 #define IOSTATEMACHINE_H
00003
00004 #include <QStateMachine>
00005 #include <QHistoryState>
00006 #include <QtCore>
00007 #include "SH_NamedObject.h"
00008 #include "SH_IState.h"
00009
0015 class SH_InOutStateMachine : public QStateMachine,
0016     SH_NamedObject
0017 {
0018     Q_OBJECT
0019 public:
0020     SH_InOutStateMachine(QString tableName, QString
0021         name="", QObject *parent = 0);
0022
0023     QString toString();
0024
0025     QVariantMap ioContent() const;
0026     void setIOcontent(const QVariantMap &ioContent);
0027     QVariant getContentValue(QString field);
0028
0029     QString tableName() const;
0030     void setTableName(const QString &tableName);
0031
0032     void setIOStateHistory(QHistoryState *state, QString field);
0033     QHistoryState* historyValue(QString field);
0034
0035 signals: /*messagers à envoyer ou transmettre*/
0036     void next();
0037     void clearAll();
0038     void sendText(QString text, bool editable=false);
0039     void resendText(QString text, bool editable=false);
0040     void receiveInput(QString input);
0041     void confirmInput();
0042     void validateInput();
0043     void replaceInput(QString field);
0044     void cancelReplacement();
0045     void displayCalendar();
0046
0047     void displayFileDialog();
0048
0049 public slots: /*réception de messagers*/
0050     void setContentValue(QVariant content, QString field);
0051     void addIState(SH_InOutState *state, QString field);
0052
0053     void addIOStateMachine(SH_InOutStateMachine* fsm);
0054     void addChildrenNextTransition(QAbstractState* previousState, QAbstractState *
0055         nextState);
0056
0057 protected:
0058     QMap<QString, QHistoryState *> ioStatesHistory() const;
0059     QVariantMap m_ioContent;
0060     QString m_tableName;
0061     QMap<QString, QHistoryState*> m_ioStatesHistory;
0062
0063 private:
0064     void setIOStatesHistory(const QMap<QString, QHistoryState *> &ioStatesHistory);
0065 };

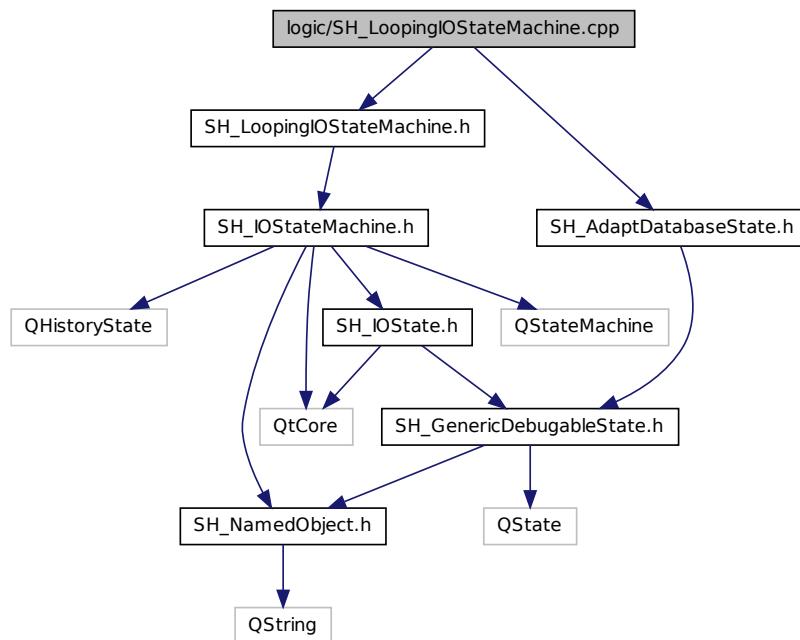
```

```
00230
00231 #endif /* IOSTATEMACHINE_H */
```

5.49 Référence du fichier logic/SH_LoopingIOStateMachine.cpp

```
#include "SH_LoopingIOStateMachine.h"
#include "SH_AdaptDatabaseState.h"
```

Graphe des dépendances par inclusion de SH_LoopingIOStateMachine.cpp :



5.50 SH_LoopingIOStateMachine.cpp

```
00001 #include "SH_LoopingIOStateMachine.h"
00002 #include "SH_AdaptDatabaseState.h"
00003
00010 Sh_LoopingInOutStateMachine::Sh_LoopingInOutStateMachine
    (QString tableName, QString name, int limit, QObject *parent) :
00011     SH_InOutStateMachine(tableName, name, parent), m_limit(limit), m_current(-1)
00012 {
00013
00014 }
00015
00016
00023 int Sh_LoopingInOutStateMachine::current() const
00024 {
00025     return m_current;
00026 }
00027
00034 void Sh_LoopingInOutStateMachine::setCurrent(int current)
00035 {
00036     m_current = current;
00037 }
00038
00039 void Sh_LoopingInOutStateMachine::setPersistentContentValue
    (QVariant value, QString field)
00040 {
00041     m_persistentContent.insert(field, value);
00042 }
00043
```

```

00050 int Sh_LoopingInOutStateMachine::limit() const
00051 {
00052     return m_limit;
00053 }
00054
00055 void Sh_LoopingInOutStateMachine::setLimit(int limit)
00056 {
00057     m_limit = limit;
00058     emit limitChanged();
00059 }
00060
00061
00062 void Sh_LoopingInOutStateMachine::stopLooping() {
00063     if(m_limit == 0) {
00064         m_limit = m_current + 1;
00065     } else {
00066         m_current = m_limit - 1;
00067     }
00068 }
00069
00070
00071 void Sh_LoopingInOutStateMachine::addChildrenNextTransition
00072     (QAbstractState *previousState, QAbstractState *nextState)
00073 {
00074     SH_GenericState* genPreviousState = qobject_cast<
00075         SH_GenericState*>(previousState);
00076     SH_InOutStateMachine* fsmPreviousState = qobject_cast<
00077         SH_InOutStateMachine*>(previousState);
00078     QFinalState* final = qobject_cast<QFinalState*>(nextState);
00079     if(final) {
00080         /*à faire au moment de l'entrée dans l'état previousState*/
00081         connect(previousState, &QAbstractState::entered, [=]() {
00082             m_current++;
00083             m_contents.append(m_ioContent);
00084             m_ioContent.clear();
00085             m_ioContent = m_persistentContent;
00086             if(m_limit == 0 || m_current < m_limit) {
00087                 if(genPreviousState) {
00088                     connect(genPreviousState, &QAbstractState::entered, [=]() {
00089                         genPreviousState->addTransition(genPreviousState, SIGNAL(
00090                             next()), initialState());
00091                     });
00092                 }
00093                 if(fsmPreviousState) {
00094                     connect(fsmPreviousState, &QAbstractState::entered, [=]() {
00095                         fsmPreviousState->addTransition(fsmPreviousState, SIGNAL(
00096                             next()), initialState());
00097                     });
00098                 } else {
00099                     SH_AdaptDatabaseState* nextSaveState = new
00100                         SH_AdaptDatabaseState("enregistrement 0 de la machine "+
00101                         toString());
00102                     if(genPreviousState) {
00103                         genPreviousState->addTransition(genPreviousState, SIGNAL(
00104                             next()), nextSaveState);
00105                     }
00106                     if(fsmPreviousState) {
00107                         fsmPreviousState->addTransition(fsmPreviousState, SIGNAL(
00108                             next()), nextSaveState);
00109                     }
00110                     if(genPreviousState || fsmPreviousState) {
00111                         for(int i = 1; i < m_limit; i++) {
00112                             SH_AdaptDatabaseState* saveState = nextSaveState;
00113                             nextSaveState = new SH_AdaptDatabaseState(QString(
00114                             "enregistrement %1 de la machine %2").arg(QString::number(i)).arg(toString()));
00115                             saveState->addTransition(saveState, SIGNAL(next()), nextSaveState);
00116                             connect(saveState, &QAbstractState::exited, [=]() {
00117                                 connect(nextSaveState, &QAbstractState::entered, [=]() {
00118                                     setContentValue(nextSaveState->
00119                                         insertUpdate(m_tableName, m_contents[i]), "ID");
00120                                     });
00121                                 });
00122                             }
00123                         nextSaveState->addTransition(nextSaveState, SIGNAL(next()), final);
00124                     }
00125                 });
00126             });
00127         }
00128     }
00129 }
00130 }
00131 }
00132 }
00133 }
00134 }
00135 }
00136 }
00137 }
00138 }
00139 }
00140 }
00141 */

```

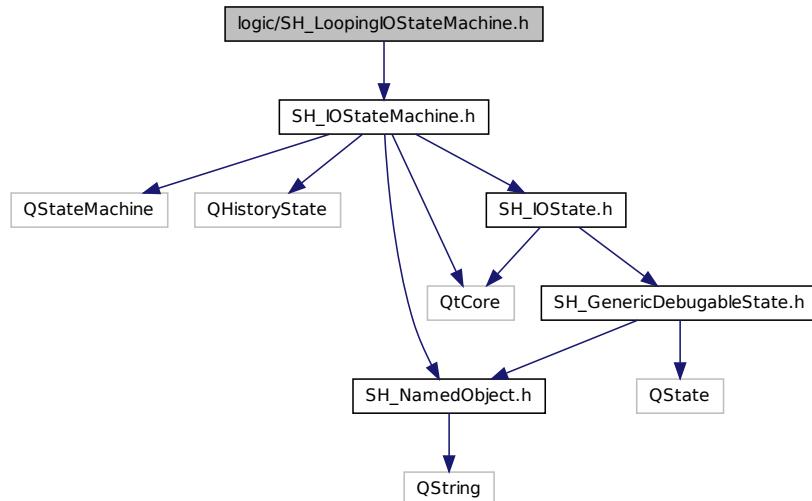
```

00142     connect(genPreviousState, &QAbstractState::entered, [=]() {
00143         connect(this, &SH_InOutStateMachine::replaceInput, [=](
00144             QString field) {
00145             /*après avoir demandé à revenir sur un état précédent, on attend la fin de l'état actuel
00146             puis on retourne à l'historique de l'état désiré; celui-ci fini, on passe à l'état qui aurait du suivre celui
00147             pendant lequel on a demandé à revenir sur un état précédent*/
00148             QHistoryState* hState = historyValue(field);
00149             if(hState) { /*si l'historique existe (on a déjà quitté l'état voulu)*/
00150                 hState->parentState()->addTransition(hState->parentState(), SIGNAL(
00151                     next()), nextState);
00152                 genPreviousState->addTransition(genPreviousState, SIGNAL(
00153                     next()), hState);
00154             });
00155         });
00156     });
00157 }
00158 }
```

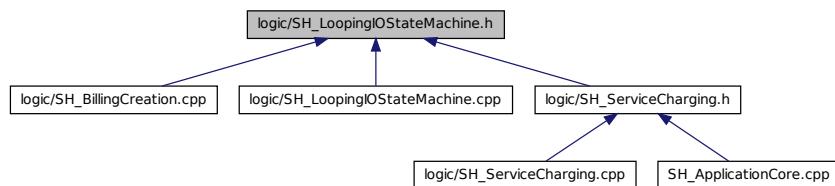
5.51 Référence du fichier logic/SH_LoopingIOStateMachine.h

#include "SH_IOSStateMachine.h"

Graphe des dépendances par inclusion de SH_LoopingIOStateMachine.h :



Ce graphe montre quels fichiers incluent directement ou indirectement ce fichier :



Classes

- class [Sh_LoopingInOutStateMachine](#)

5.52 SH_LoopingIStateMachine.h

```

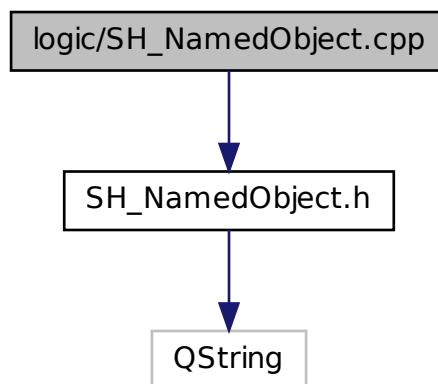
00001 #ifndef LOOPINGSTATEMACHINE_H
00002 #define LOOPINGSTATEMACHINE_H
00003 #include "SH_IStateMachine.h"
00004
00010 class Sh_LoopingInOutStateMachine : public
00011     SH_InOutStateMachine
00012 {
00013     Q_OBJECT
00013     Q_PROPERTY(int limit READ limit WRITE setLimit NOTIFY
00013         limitChanged)
00014
00015 public:
00025     Sh_LoopingInOutStateMachine(QString tableName, QString
00025         name="looping", int limit=0, QObject *parent = 0);
00026
00033     int current() const;
00040     void setCurrent(int current);
00041
00049     void setPersistentContentValue(QVariant value, QString field);
00050
00057     int limit() const;
00064     void setLimit(int limit);
00065
00072     void addChildrenNextTransition(QAbstractState *previousState, QAbstractState *
00072         nextState);
00077     void stopLooping();
00078 signals:
00083     void limitChanged();
00084
00085 public slots:
00086
00087 private:
00091     int m_limit;
00095     int m_current;
00099     QList<QVariantMap> m_contents;
00103     QVariantMap m_persistentContent;
00104 };
00105
00106 #endif /* LOOPINGSTATEMACHINE_H*/

```

5.53 Référence du fichier logic/SH_NamedObject.cpp

#include "SH_NamedObject.h"

Graphe des dépendances par inclusion de SH_NamedObject.cpp :



5.54 SH_NamedObject.cpp

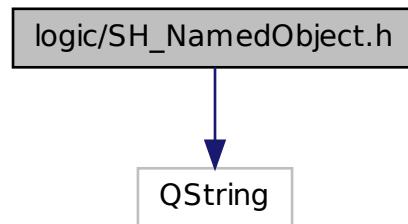
```

00001 #include "SH_NamedObject.h"
00002
00009 SH_NamedObject::SH_NamedObject(QString name) :
00010     m_name(name)
00011 {
00012     /*m_ptraddress = QString(&this);*/
00013 }
00014
00021 QString SH_NamedObject::toString()
00022 {
00023     return this->m_name;
00024 }
00025
00032 QString SH_NamedObject::name() const
00033 {
00034     return m_name;
00035 }
00036
00043 void SH_NamedObject::setName(const QString &name)
00044 {
00045     m_name = name;
00046 }
00047
00054 QString SH_NamedObject::ptraddress() const
00055 {
00056     return m_ptraddress;
00057 }

```

5.55 Référence du fichier logic/SH_NamedObject.h

#include <QString>
Graphe des dépendances par inclusion de SH_NamedObject.h :



Ce graphe montre quels fichiers incluent directement ou indirectement ce fichier :



Classes

- class [SH_NamedObject](#)

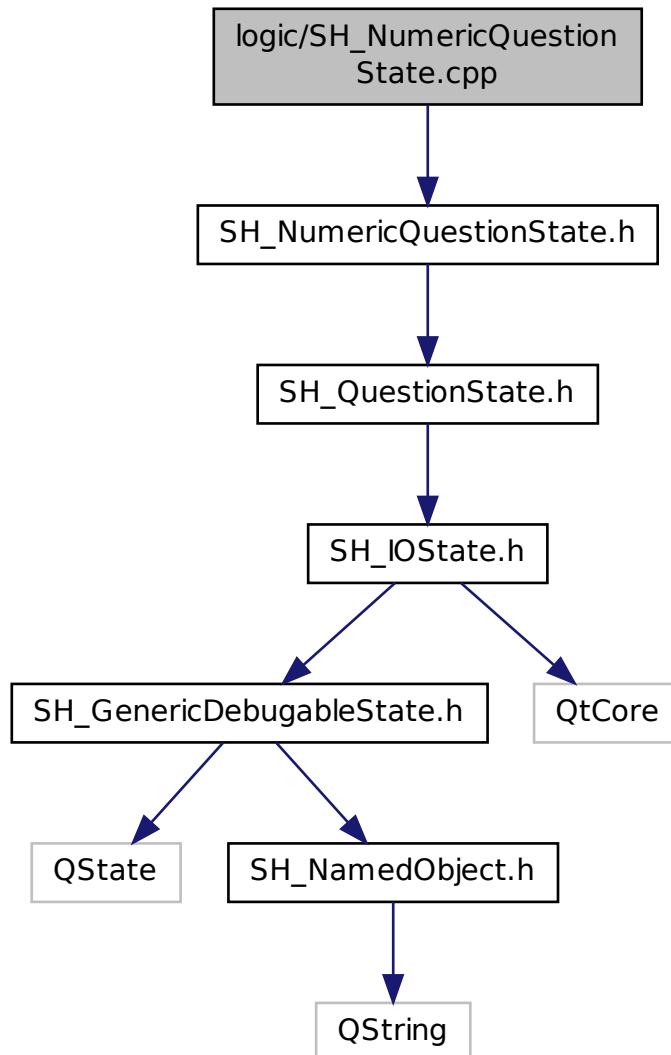
5.56 SH_NamedObject.h

```
00001 #ifndef NAMEDOBJECT_H
00002 #define NAMEDOBJECT_H
00003 #include <QString>
00004
00010 class SH_NamedObject
00011 {
00012     public:
00019     SH_NamedObject(QString name);
00026     virtual QString toString();
00027
00034     virtual QString name() const;
00041     virtual void setName(const QString &name);
00042
00049     QString ptraddress() const;
00050
00051     private:
00055     QString m_name;
00059     QString m_ptraddress;
00060 };
00061
00062 #endif /* NAMEDOBJECT_H */
```

5.57 Référence du fichier logic/SH_NumericQuestionState.cpp

```
#include "SH_NumericQuestionState.h"
```

Graphe des dépendances par inclusion de SH_NumericQuestionState.cpp :



5.58 SH_NumericQuestionState.cpp

```

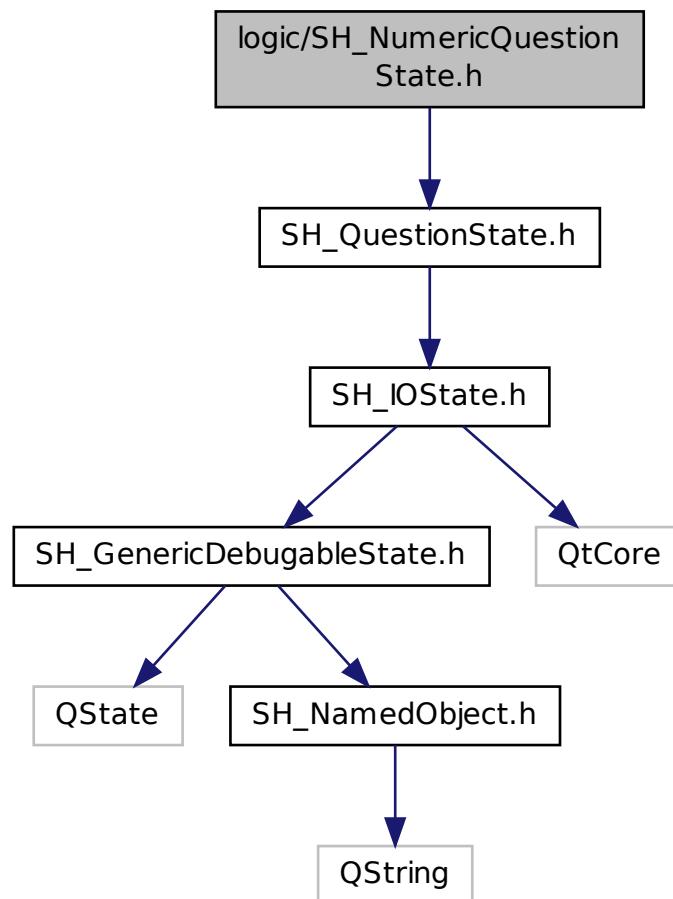
00001 #include "SH_NumericQuestionState.h"
00002
00003
00010 SH_NumericQuestionState::SH_NumericQuestionState(QString question, QString name, int min, int max, QState *parent) :
00011     SH_QuestionState(question, name, parent), m_min(min), m_max(max)
00012 {
00013
00014 }
00015
00022 bool SH_NumericQuestionState::isValid(const QVariant & givenAnswer)
00023 {
00024     qDebug() << "is answer valid";
00025     bool ok;
00026     int answer = givenAnswer.toInt(&ok);
  
```

```
00027     if(ok) {
00028         return ((m_max <= m_min || answer <= m_max) && answer >=
00029             m_min);
00030     } else {
00031         return false;
00032     }
00033
00040 int SH_NumericQuestionState::min() const
00041 {
00042     return m_min;
00043 }
00044
00050 void SH_NumericQuestionState::setMin(int min)
00051 {
00052     m_min = min;
00053 }
00054
00061 int SH_NumericQuestionState::max() const
00062 {
00063     return m_max;
00064 }
00065
00072 void SH_NumericQuestionState::setMax(int max)
00073 {
00074     m_max = max;
00075 }
```

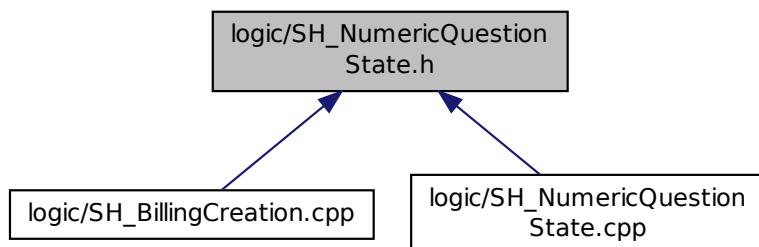
5.59 Référence du fichier logic/SH_NumericQuestionState.h

```
#include "SH_QuestionState.h"
```

Graphe des dépendances par inclusion de SH_NumericQuestionState.h :



Ce graphe montre quels fichiers incluent directement ou indirectement ce fichier :



Classes

- class **SH_NumericQuestionState**

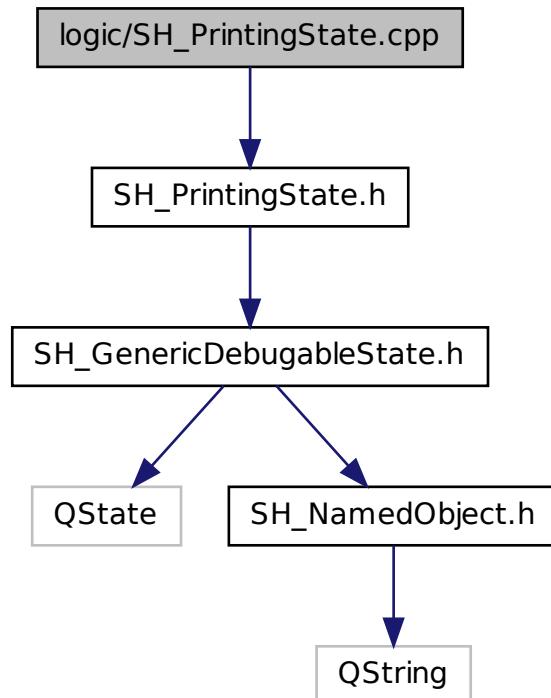
5.60 SH_NumericQuestionState.h

```
00001 #ifndef NUMERICQUESTIONSTATE_H
00002 #define NUMERICQUESTIONSTATE_H
00003 #include "SH_QuestionState.h"
00004
00010 class SH_NumericQuestionState : public SH_QuestionState
00011 {
00012     Q_OBJECT
00013 public:
00024     SH_NumericQuestionState(QString question, QString
00025         name, int min=0, int max=-1, QState *parent = 0);
00031 virtual bool isAnswerValid(const QVariant &givenAnswer);
00032
00039 int min() const;
00046 void setMin(int min);
00047
00054 int max() const;
00061 void setMax(int max);
00062
00063 signals:
00064
00065 public slots:
00066
00067
00068 private:
00072 int m_min;
00076 int m_max;
00077 };
00078
00079 #endif /* NUMERICQUESTIONSTATE_H */
```

5.61 Référence du fichier logic/SH_PrintingState.cpp

```
#include "SH_PrintingState.h"
```

Graphe des dépendances par inclusion de SH_PrintingState.cpp :



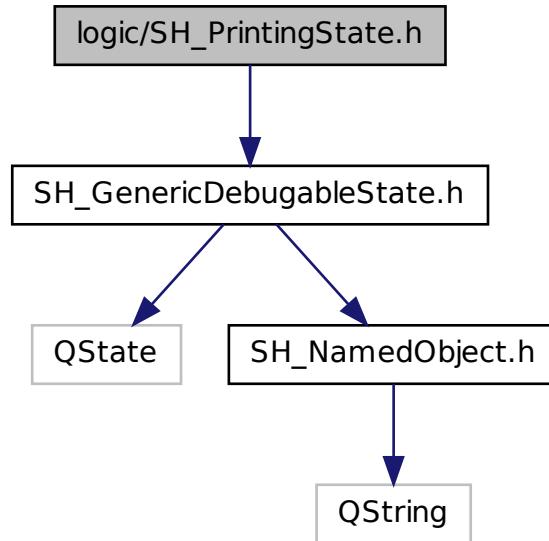
5.62 SH_PrintingState.cpp

```
00001 #include "SH_PrintingState.h"
00002
00009 SH_PrintingState::SH_PrintingState(QString name,
    QState *parent) :
00010     SH_GenericState(name, parent)
00011 {
00012 }
```

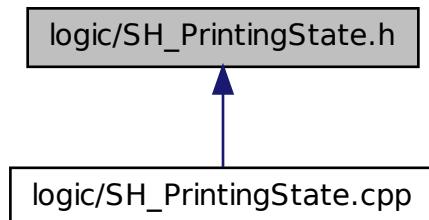
5.63 Référence du fichier logic/SH_PrintingState.h

```
#include "SH_GenericDebugableState.h"
```

Graphe des dépendances par inclusion de SH_PrintingState.h :



Ce graphe montre quels fichiers incluent directement ou indirectement ce fichier :



Classes

- class [SH_PrintingState](#)

5.64 SH_PrintingState.h

```

00001 #ifndef PRINTINGSTATE_H
00002 #define PRINTINGSTATE_H
00003 #include "SH_GenericDebugableState.h"
00004
00010 class SH_PrintingState : public SH_GenericState
00011 {
00012     Q_OBJECT
  
```

```

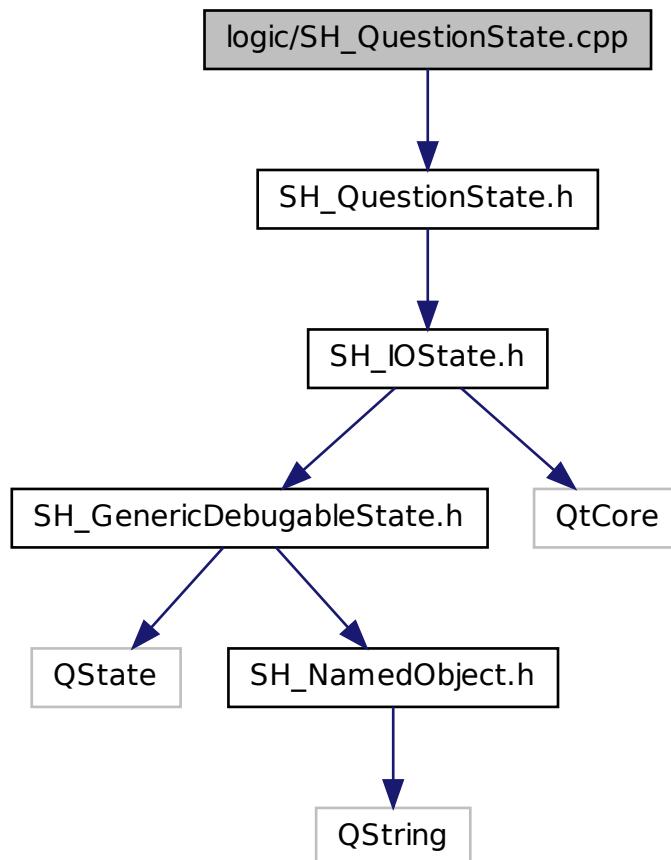
00013 public:
00021     SH_PrintingState(QString name, QState *parent = 0);
00022
00023 signals:
00029     void printStarted();
00035     void printFinished();
00036
00037 public slots:
00038
00039 };
00040
00041 #endif /* PRINTINGSTATE_H */

```

5.65 Référence du fichier logic/SH_QuestionState.cpp

#include "SH_QuestionState.h"

Graphe des dépendances par inclusion de SH_QuestionState.cpp :



5.66 SH_QuestionState.cpp

```

00001 #include "SH_QuestionState.h"
00002
00009 SH_QuestionState::SH_QuestionState(QString question, QString name,
    QState *parent) :
00010     SH_InOutState(question, name, parent)

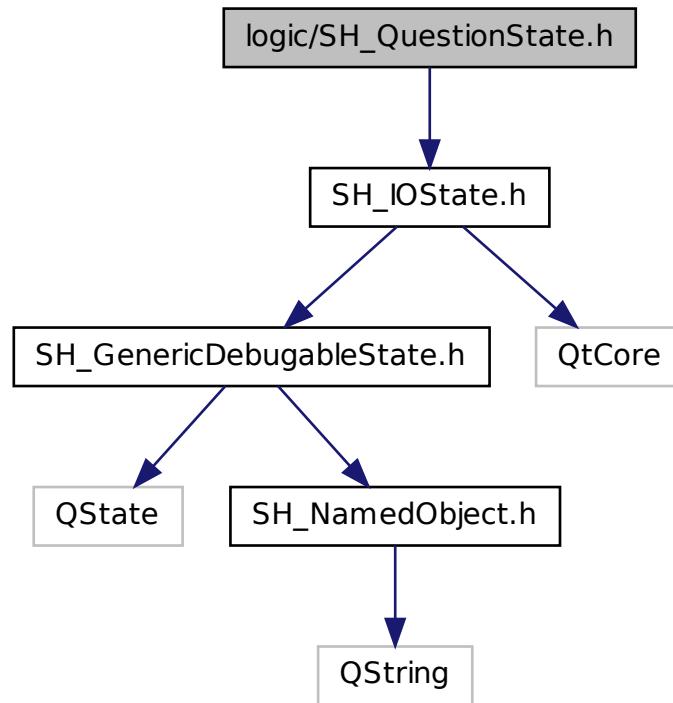
```

```
00011 {
00012 }
00013
00020 bool SH_QuestionState::checkValidity()
00021 {
00022     bool ok = this->isAnswerValid(this->givenAnswer());
00023     if(ok) {
00024         qDebug() << this->givenAnswer() << " answer valid !";
00025         qDebug() << this->givenAnswer();
00026         SH_InOutState::setInput(this->givenAnswer());
00027         emit answerValid();
00028         emit next();
00029     } else {
00030         qDebug() << this->givenAnswer() << " answer invalid :-(";
00031         qDebug() << this->givenAnswer();
00032         emit answerInvalid();
00033     }
00034     return ok;
00035 }
00036
00043 void SH_QuestionState::setInput(const QVariant &input)
00044 {
00045     qDebug() << "new answer " << input.toString();
00046     this->setGivenAnswer(input);
00047 }
00048
00055 QVariant SH_QuestionState::givenAnswer() const
00056 {
00057     return this->m_givenAnswer;
00058 }
00059
00066 void SH_QuestionState::setGivenAnswer(const QVariant &givenAnswer)
00067 {
00068     this->m_givenAnswer = givenAnswer;
00069     this->checkValidity();
00070 }
```

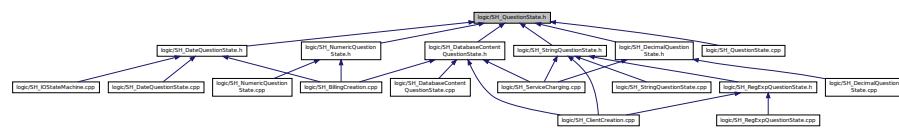
5.67 Référence du fichier logic/SH_QuestionState.h

```
#include "SH_IOSState.h"
```

Graphe des dépendances par inclusion de SH_QuestionState.h :



Ce graphe montre quels fichiers incluent directement ou indirectement ce fichier :



Classes

- class SH QuestionState

5.68 SH_QuestionState.h

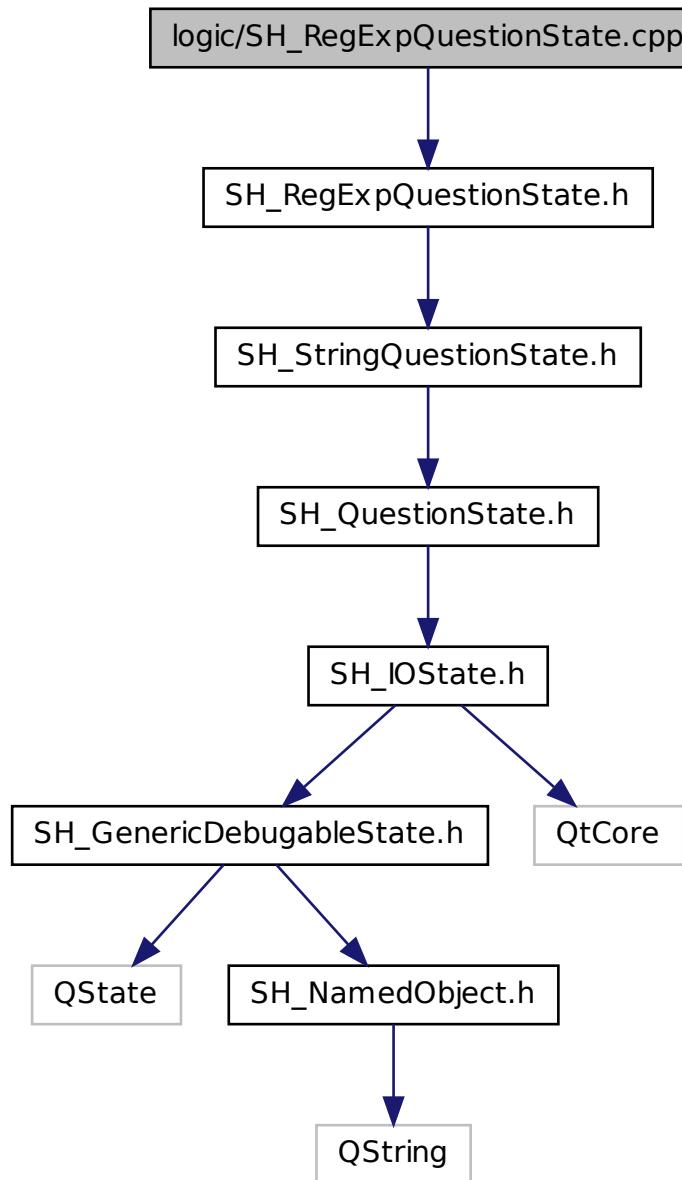
```
00001 #ifndef QUESTIONSTATE_H
00002 #define QUESTIONSTATE_H
00003 #include "SH_IOSState.h"
00004
00011 class SH_QuestionState : public SH_InOutState
00012 {
00013     Q_OBJECT
00014 public:
00023     SH_QuestionState(QString question, QString name, QState *parent = 0);
00030     bool checkValidity();
00031
00038     virtual QVariant givenAnswer() const;
```

```
00045     virtual void setGivenAnswer(const QVariant &givenAnswer);
00052     virtual void setInput(const QVariant &input);
00053
00061     virtual bool isAnswerValid(const QVariant &givenAnswer) = 0;
00062
00063 signals:
00067     void answerValid();
00071     void answerInvalid();
00072
00073 public slots:
00074
00075
00076 private:
00080     QVariant m_givenAnswer;
00081 };
00082
00083 #endif /* QUESTIONSTATE_H*/
```

5.69 Référence du fichier logic/SH_RegExpQuestionState.cpp

```
#include "SH_RegExpQuestionState.h"
```

Graphe des dépendances par inclusion de SH_RegExpQuestionState.cpp :



5.70 SH_RegExpQuestionState.cpp

```

00001 #include "SH_RegExpQuestionState.h"
00002
00003 SH_RegExpQuestionState::SH_RegExpQuestionState(QString
00004     question, QString name, QRegularExpression regex, QState *parent) :
00005     SH_StringQuestionState(question, name, 0,-1,parent)
00006 {
00007 }
00008
00009     bool SH_RegExpQuestionState::isValid(const QVariant &givenAnswer
00010     )
00011 {
00012
  
```

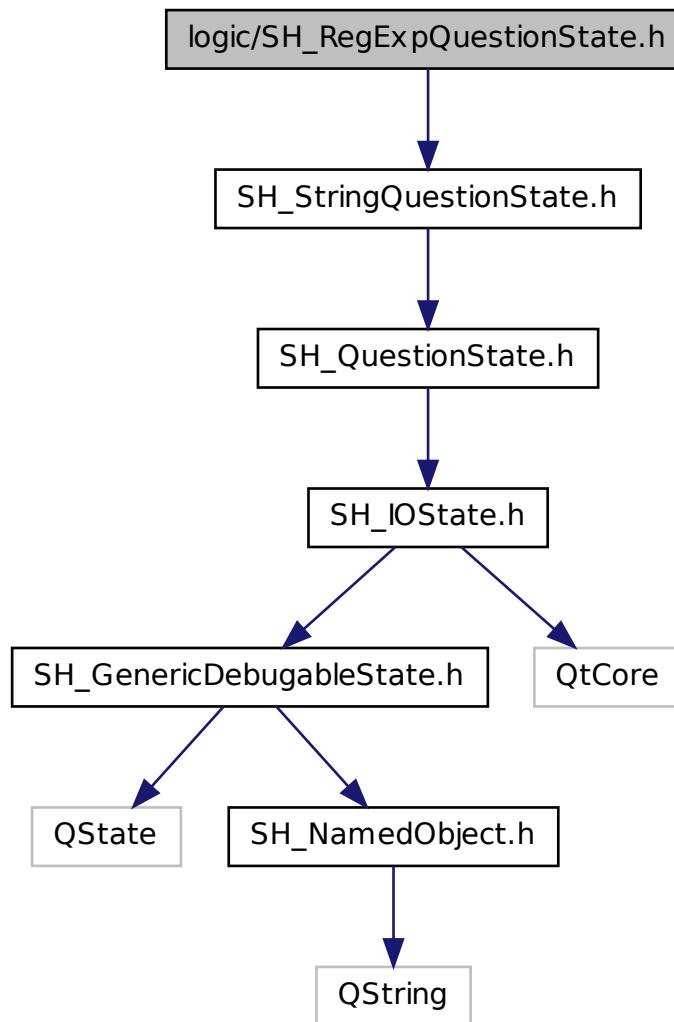
```

00020     QString answer = givenAnswer.toString();
00021     QRegularExpressionMatch found = m_regexp.match(answer);
00022     return (found.hasMatch() && (found.captured(0) == answer));
00023 }
00024
00025
00031 QRegularExpression SH_RegExpQuestionState::regexp() const
00032 {
00033     return m_regexp;
00034 }
00035
00041 void SH_RegExpQuestionState::setRegexp(const QRegularExpression &regexp)
00042 {
00043     m_regexp = regexp;
00044 }
```

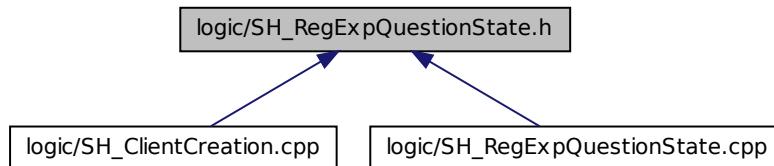
5.71 Référence du fichier logic/SH_RegExpQuestionState.h

#include "SH_StringQuestionState.h"

Graphe des dépendances par inclusion de SH_RegExpQuestionState.h :



Ce graphe montre quels fichiers incluent directement ou indirectement ce fichier :



Classes

- class [SH_RegExpQuestionState](#)
The SH_RegExpQuestionState class.

5.72 SH_RegExpQuestionState.h

```

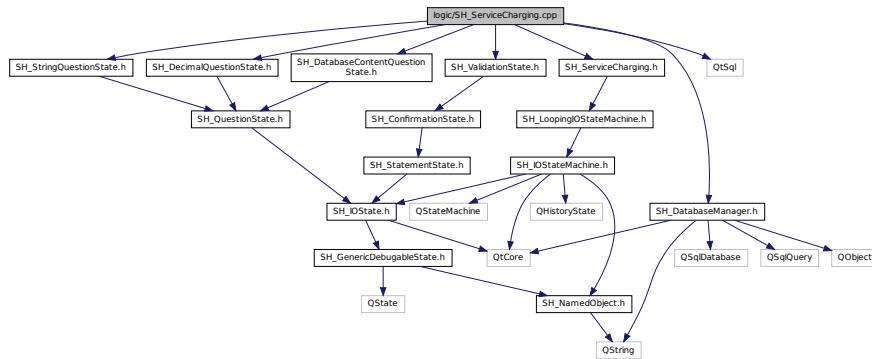
00001 #ifndef REGEXQUESTIONSTATE_H
00002 #define REGEXQUESTIONSTATE_H
00003 #include "SH_StringQuestionState.h"
00004
00008 class SH_RegExpQuestionState : public
00009     SH_StringQuestionState
00010 {
00011     Q_OBJECT
00012 public:
00020     SH_RegExpQuestionState(QString question, QString name, QRegularExpression
00021     regex = QRegularExpression(), QState *parent = 0);
00022
00028     virtual bool isAnswerValid(const QVariant &givenAnswer);
00034     QRegularExpression regexp() const;
00040     void setRegexp(const QRegularExpression &regexp);
00041
00042 signals:
00043
00044 public slots:
00045
00046
00047 private:
00051     QRegularExpression m_regexp;
00052 };
00053
00054 #endif /* REGEXQUESTIONSTATE_H */
  
```

5.73 Référence du fichier logic/SH_ServiceCharging.cpp

```

#include "SH_ServiceCharging.h"
#include "SH_ValidationState.h"
#include "SH_DatabaseContentQuestionState.h"
#include "SH_StringQuestionState.h"
#include "SH.DecimalQuestionState.h"
#include "SH_DatabaseManager.h"
#include <QtSql>
  
```

Graphe des dépendances par inclusion de SH_ServiceCharging.cpp :



5.74 SH_ServiceCharging.cpp

```

00001 #include "SH_ServiceCharging.h"
00002 #include "SH_ValidationState.h"
00003 #include "SH_DatabaseContentQuestionState.h"
00004 #include "SH_StringQuestionState.h"
00005 #include "SH_DecimalQuestionState.h"
00006 #include "SH_DatabaseManager.h"
00007 #include <QtSql>
00008
00012 SH_ServiceCharging::SH_ServiceCharging(QString name,
    QObject *parent) :
00013     Sh_LoopingInOutStateMachine("CHARGEDSERVICES",name, 0, parent), m_priceMin(0
    .0)
00014 {
00015     SH_DatabaseContentQuestionState* service = new
    SH_DatabaseContentQuestionState("Veuillez sélectionner une prestation ou
        appuyer sur la touche \"VALIDER\" pour cesser d'ajouter des prestations", "choose service in service charging",
    "SERVICES","CODE");
00016     SH_InOutState*serviceId = new SH_InOutState("", "service id in service
        charging");
00017     serviceId->setVisibility(false);
00018     SH_StringQuestionState* serviceName = new
    SH_StringQuestionState("Veuillez entrer ce qui sera affiché sur la facture", "service
        name in service charging",1);
00019     SH_DecimalQuestionState* price = new
    SH_DecimalQuestionState("", "price in service charging",-Q_INFINITY,Q_INFINITY);
00020     SH_DecimalQuestionState* quantity = new
    SH_DecimalQuestionState("", "quantity in service charging",1);
00021     SH_DatabaseContentQuestionState* vat = new
    SH_DatabaseContentQuestionState("", "vat in service charging","TAXES",
    "PERCENTAGE","ENABLED='1'");
00022     QFinalState* final = new QFinalState();
00023
00024
00025     connect(service, &SH_QuestionState::answerInvalid, [=]() {
00026         int in = service->rawInput().toInt();
00027         if(in == -1 || in == 0) {
00028             emit service->next();
00029         }
00030     });
00031     connect(service, &SH_QuestionState::answerValid, [=]() {
00032         if(service->rawInput().toInt() > -1) {
00033             QString name;
00034             QStringList list;
00035             list.append("PRINTEDNAME");
00036             list.append("PRICEMIN");
00037             list.append("PRICEMAX");
00038             list.append("VAT_PERCENTAGE");
00039             list.append("ID");
00040             QSqlQuery result = SH_DatabaseManager::getInstance()->
    execSelectQuery("SERVICESINFOS", list, QString("CODE=%1").arg(service->
    rawInput().toString()));
00041             result.next();
00042             QSqlRecord record = result.record();
00043             name= record.value(0).toString();
00044             m_priceMin =record.value(1).toDouble();
00045             m_vat =record.value(3).toDouble();
00046             serviceId->setInput(record.value(4).toInt());

```

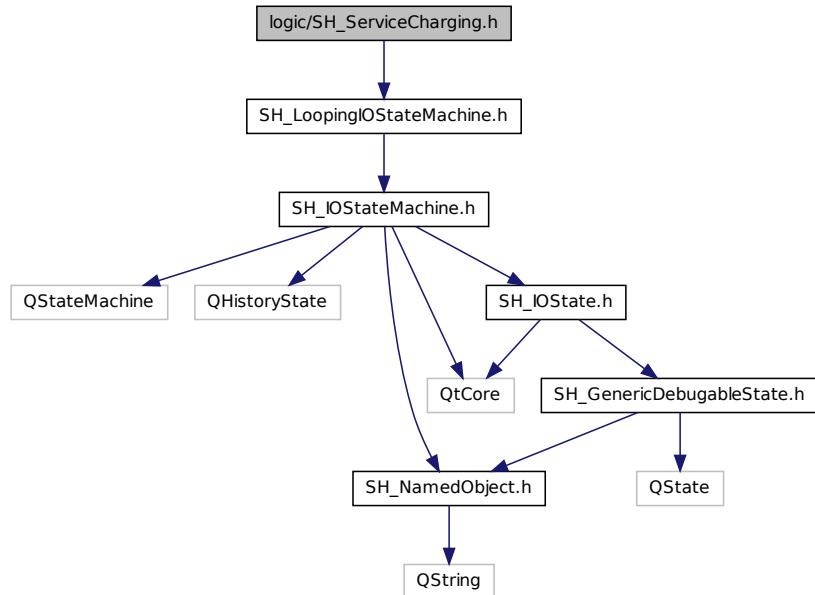
```

00047         serviceName->setInput(name);
00048         price->setOutput(QString("Le prix proposé pour cette prestation est : %1. Son prix
00049             minimum est %1 et son prix maximum %2.\nVeuillez entrer un nouveau prix ou appuyer sur la touche \"CONFIRMER\""
00050             ).arg(record.value(1).toString()).arg(record.value(2).toString()));
00051         vat->setOutput(QString("Cette prestation est associée à une TVA de %1%.\nVeuillez
00052             entrer une autre TVA à appliquer ou appuyer sur la touche \"CONFIRMER\"").arg(record.value(3).toString()));
00053         serviceName->setVisibility(false);
00054     }
00055     });
00056     connect(quantity, &QState::entered, [=]() {
00057         connect(this, &SH_InOutStateMachine::receiveInput, [=](QString in
00058         ) {
00059             QString newInput;
00060             if(in.right(in.length() - 1).toInt() != 0) {
00061                 newInput = in.right(in.length() - 1);
00062             }
00063             emit receiveInput(newInput);
00064         });
00065     });
00066     connect(price, &QState::entered, [=]() {
00067         connect(this, &SH_InOutStateMachine::confirmInput, [=]() {
00068             price->setInput(m_priceMin);
00069         });
00070     });
00071 };
00072
00073
00074 this->addState(final);
00075 this->addIOState(service, "");
00076 this->addIOState(serviceId, "SERVICE_ID");
00077 this->addIOState(serviceName, "PRINTEDNAME");
00078 this->addIOState(price, "CHARGEDUNITPRICE");
00079 this->addIOState(quantity, "QUANTITY");
00080 this->addIOState(vat, "CHARGEDVAT");
00081 this->addChildsNextTransition(service, serviceId);
00082 this->addChildsNextTransition(serviceId, serviceName);
00083 this->addChildsNextTransition(serviceName, quantity);
00084 this->addChildsNextTransition(quantity, price);
00085 this->addChildsNextTransition(price, vat);
00086 this->addChildsNextTransition(vat, final);
00087 this->setInitialState(service);
00088 connect(this, &SH_InOutStateMachine::validateInput, this, &
00089     Sh_LoopingInOutStateMachine::stopLooping);
00090 }
```

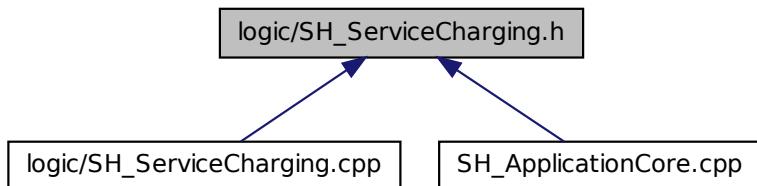
5.75 Référence du fichier logic/SH_ServiceCharging.h

```
#include "SH_LoopingIOStateMachine.h"
```

Graphe des dépendances par inclusion de SH_ServiceCharging.h :



Ce graphe montre quels fichiers incluent directement ou indirectement ce fichier :



Classes

- class `SH_ServiceCharging`
The `SH_ServiceCharging` class.

5.76 SH_ServiceCharging.h

```

00001 #ifndef SERVICECHARGING_H
00002 #define SERVICECHARGING_H
00003 #include "SH_LoopingIOStateMachine.h"
00004
00008 class SH_ServiceCharging : public Sh_LoopingInOutStateMachine
00009 {
00010     Q_OBJECT
00011 public:
00017     SH_ServiceCharging(QString name, QObject *parent = 0);
00018
  
```

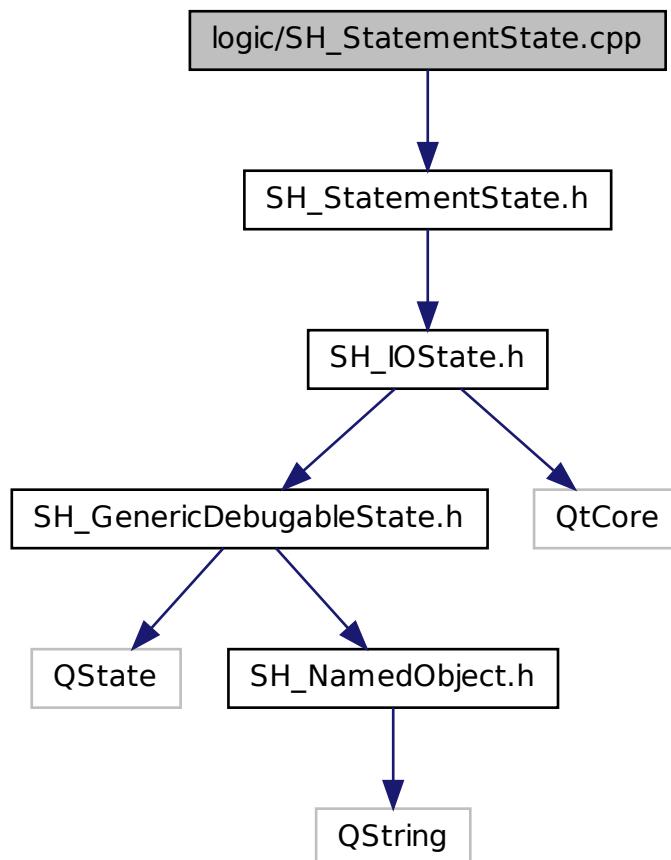
```

00019 signals:
00020
00021 public slots:
00022
00023 private:
00027     qreal m_priceMin;
00031     qreal m_vat;
00032
00033 };
00034
00035 #endif /* SERVICECHARGING_H*/

```

5.77 Référence du fichier logic/SH_StatementState.cpp

#include "SH_StatementState.h"
Graphe des dépendances par inclusion de SH_StatementState.cpp :



5.78 SH_StatementState.cpp

```

00001 #include "SH_StatementState.h"
00002
00009 SH_StatementState::SH_StatementState(QString output, QString name,
    QState *parent) :
00010     SH_InOutState(output, name, parent)
00011 {

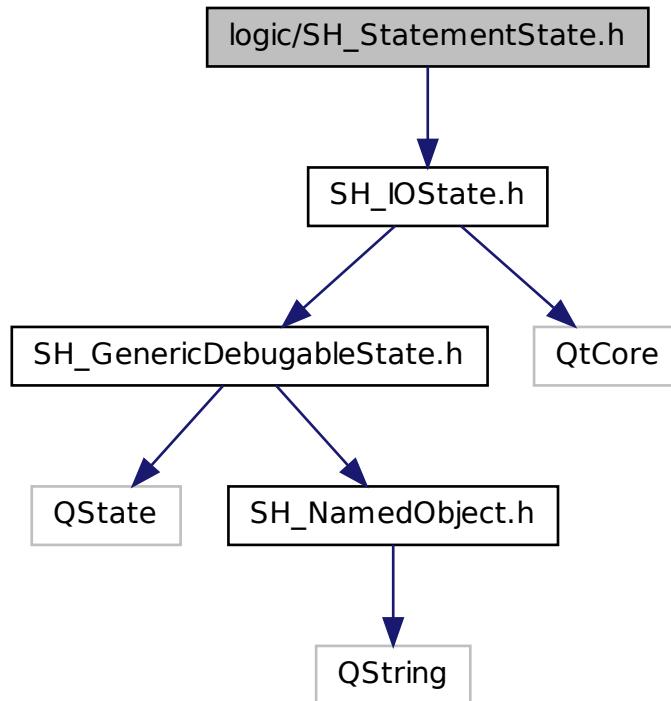
```

```
00012     qDebug() << "salut ! " << output;
00013 }
00014
00021 void SH_StatementState::setInput(const QVariant &input)
00022 {
00023     Q_UNUSED(input);
00024     /*DO NOTHING*/
00025 }
00026
00033 void SH_StatementState::onEntry(QEvent *event)
00034 {
00035     SH_GenericState::onEntry(event);
00036     display(true);
00037     emit next();
00038 }
```

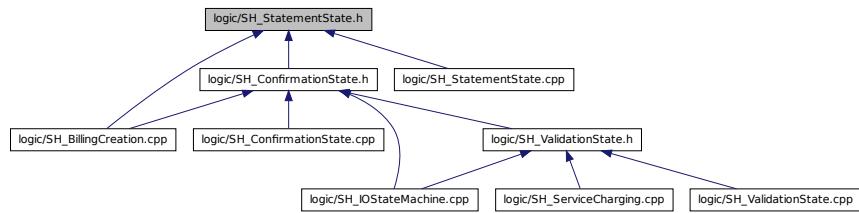
5.79 Référence du fichier logic/SH_StatementState.h

#include "SH_IOSState.h"

Graphe des dépendances par inclusion de SH_StatementState.h :



Ce graphe montre quels fichiers incluent directement ou indirectement ce fichier :



Classes

- class [SH_StatementState](#)

5.80 SH_StatementState.h

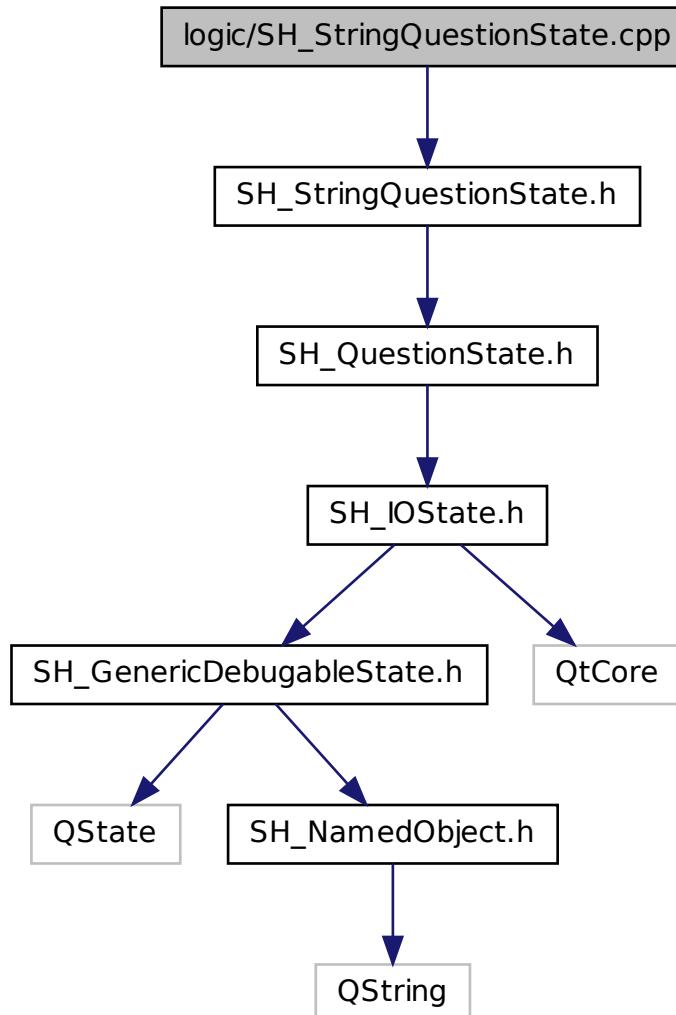
```

00001 #ifndef STATEMENTSTATE_H
00002 #define STATEMENTSTATE_H
00003 #include "SH_IState.h"
00004
00010 class SH_StatementState : public SH_InOutState
00011 {
00012     Q_OBJECT
00013 public:
00022     SH_StatementState(QString output, QString name,
00023     QState *parent = 0);
00023
00030     void setInput(const QVariant &input);
00037     void onEntry(QEvent *event);
00038
00039 signals:
00040
00041 public slots:
00042
00043 };
00044
00045 #endif /* STATEMENTSTATE_H */
  
```

5.81 Référence du fichier logic/SH_StringQuestionState.cpp

```
#include "SH_StringQuestionState.h"
```

Graphe des dépendances par inclusion de SH_StringQuestionState.cpp :



5.82 SH_StringQuestionState.cpp

```

00001 #include "SH_StringQuestionState.h"
00002
00003
00010 SH_StringQuestionState::SH_StringQuestionState(QString
    question, QString name, int minLength, int maxLength, QState *parent) :
00011     SH_QuestionState(question, name, parent), m_minLen(minLength), m_maxLen(maxLength)
00012 {
00013
00014 }
00015
00022 bool SH_StringQuestionState::isValid(const QVariant &givenAnswer
    )
00023 {
00024     QString answer = givenAnswer.toString();
00025     if(!answer.isEmpty()) {
00026         int answerLength= answer.length();
00027         return ((m_maxLen <= m_minLen || answerLength <=
    m_maxLen) && answerLength >= m_minLen);
  
```

```

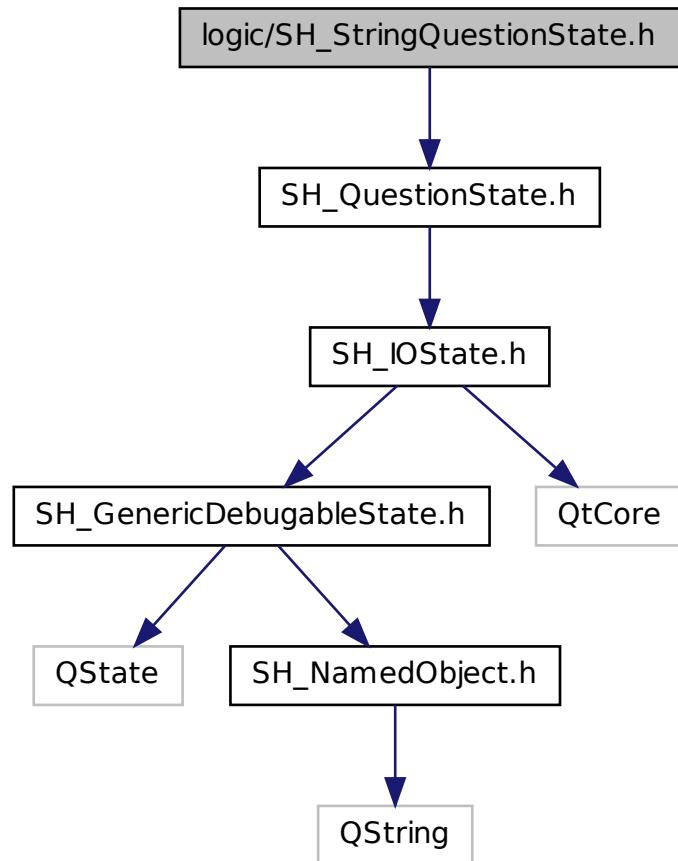
00028     } else {
00029         return false;
00030     }
00031 }
00032
00039 int SH_StringQuestionState::maxLen() const
00040 {
00041     return m_maxLen;
00042 }
00043
00050 void SH_StringQuestionState::setMaxLen(int maxLen)
00051 {
00052     m_maxLen = maxLen;
00053 }
00054
00061 int SH_StringQuestionState::minLen() const
00062 {
00063     return m_minLen;
00064 }
00065
00072 void SH_StringQuestionState::setMinLen(int minLen)
00073 {
00074     m_minLen = minLen;
00075 }

```

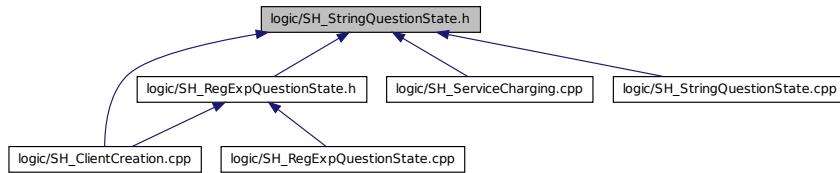
5.83 Référence du fichier logic/SH_StringQuestionState.h

#include "SH_QuestionState.h"

Graphe des dépendances par inclusion de SH_StringQuestionState.h :



Ce graphe montre quels fichiers incluent directement ou indirectement ce fichier :



Classes

- class [SH_StringQuestionState](#)

5.84 SH_StringQuestionState.h

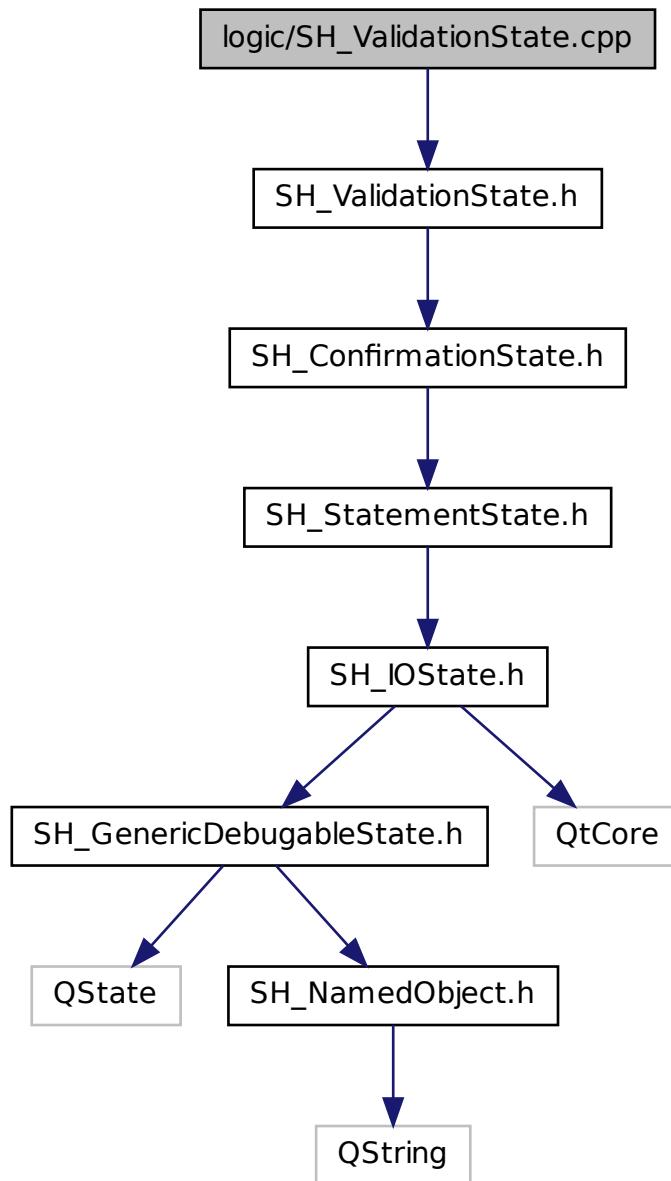
```

00001 #ifndef STRINGQUESTIONSTATE_H
00002 #define STRINGQUESTIONSTATE_H
00003 #include "SH_QuestionState.h"
00004
00010 class SH_StringQuestionState : public SH_QuestionState
00011 {
00012     Q_OBJECT
00013 public:
00024     SH_StringQuestionState(QString question, QString name, int minLength = 0, int
00025     maxLength = -1, QState *parent = 0);
00031     virtual bool isAnswerValid(const QVariant &givenAnswer);
00032
00039     int maxLen() const;
00046     void setMaxLen(int maxLen);
00047
00054     int minLen() const;
00061     void setMinLen(int minLen);
00062
00063     signals:
00064
00065     public slots:
00066
00067     private:
00071     int m_minLen;
00075     int m_maxLen;
00076
00077 };
00078
00079 #endif /* STRINGQUESTIONSTATE_H */
  
```

5.85 Référence du fichier logic/SH_ValidationState.cpp

```
#include "SH_ValidationState.h"
```

Graphe des dépendances par inclusion de SH_ValidationState.cpp :



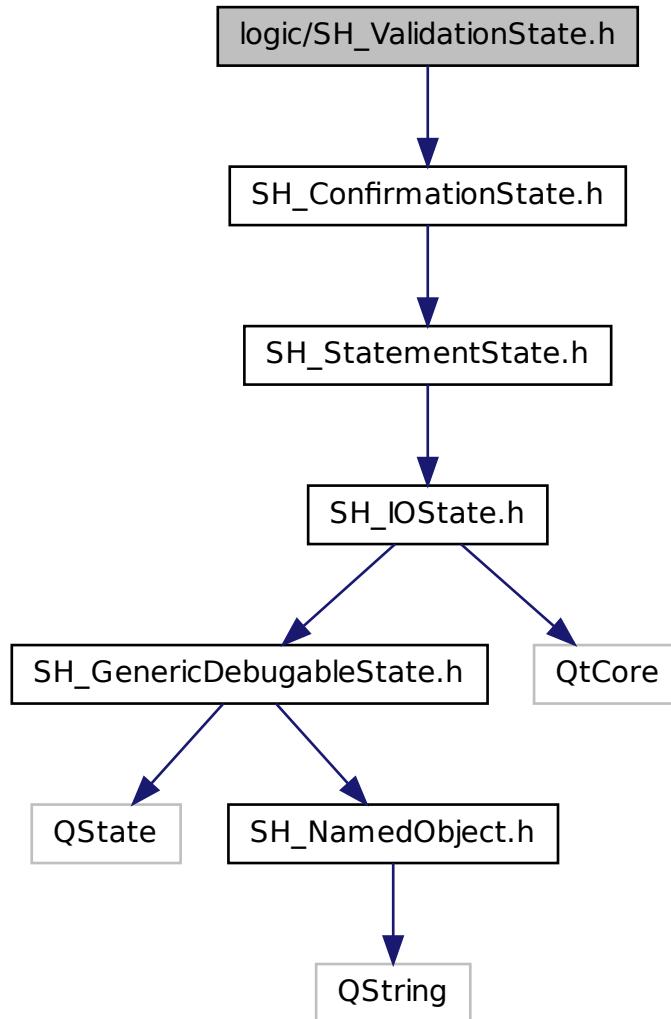
5.86 SH_ValidationState.cpp

```
00001 #include "SH_ValidationState.h"
00002
00003
00004
00011 SH_ValidationState::SH_ValidationState(QString output, QString name,
    QState *parent) :
00012     SH_ConfirmationState(output, name, parent)
00013 {
00014
00015 }
```

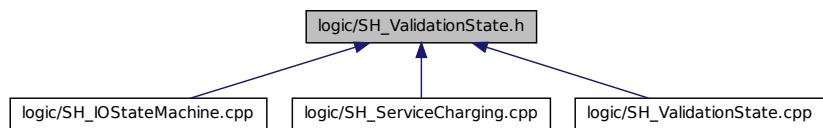
5.87 Référence du fichier logic/SH_ValidationState.h

```
#include "SH_ConfirmationState.h"
```

Graphe des dépendances par inclusion de SH_ValidationState.h :



Ce graphe montre quels fichiers incluent directement ou indirectement ce fichier :



Classes

- class [SH_ValidationState](#)

5.88 SH_ValidationState.h

```

00001 #ifndef VALIDATIONSTATE_H
00002 #define VALIDATIONSTATE_H
00003 #include "SH_ConfirmationState.h"
00004
0010 class SH_ValidationState : public SH_ConfirmationState
0011 {
0012     Q_OBJECT
0013 public:
0022     SH_ValidationState(QString output, QString name,
0023     QState *parent = 0);
0024 signals:
0025
0026 public slots:
0027
0028 };
0029
0030 #endif /* VALIDATIONSTATE_H*/

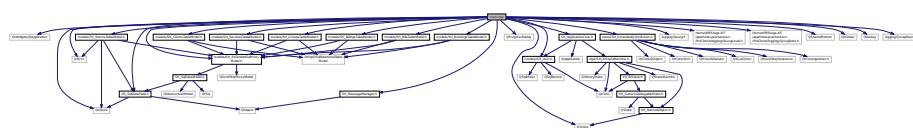
```

5.89 Référence du fichier main.cpp

```

#include <QtWidgets/QApplication>
#include <QtQml>
#include <QtQuick>
#include <QProgressDialog>
#include "SH_MessageManager.h"
#include "SH_ApplicationCore.h"
#include "models/SH_ExtendedSqlProxyModel.h"
#include "models/SH_RoomsTableModel.h"
#include "models/SH_SqlDataField.h"
#include "models/SH_BillingsTableModel.h"
#include "models/SH_BillsTableModel.h"
#include "models/SH_BookingsTableModel.h"
#include "models/SH_ClientsTableModel.h"
#include "models/SH_ServicesTableModel.h"
#include "models/SH_GroupsTableModel.h"
#include "models/SH_User.h"
#include "views/SH_ExtendedQQmlAction.h"
#include "logging/QsLog.h"
#include "logging/QsLogDest.h"
Graphe des dépendances par inclusion de main.cpp :

```



Fonctions

- void [enableLogging](#) (const QString sLogPath)
- void [exportlog](#) (QtMsgType type, const QMessageLogContext &context, const QString &msg)
- int [main](#) (int argc, char **argv)
- void [spin](#) (int &iteration)
- void [statusChanged](#) (QQmlComponent *component, QQmlComponent :Status status)

Variables

- const int **iterations** = 20

5.89.1 Documentation des fonctions

5.89.1.1 void enableLogging (const QString *sLogPath*)

enableLogging

Paramètres

<i>sLogPath</i>

Définition à la ligne 71 du fichier [main.cpp](#).

Référencé par [main\(\)](#).

```

00072 {
00073     /* init the logging mechanism*/
00074     QsLogging::Logger& logger = QsLogging::Logger::instance();
00075     logger.setLoggingLevel(QsLogging::TraceLevel);
00076
00077     QsLogging::DestinationPtr fileDestination(
00078         QsLogging::DestinationFactory::MakeFileDestination(sLogPath, true, 512, 2) );
00079     QsLogging::DestinationPtr debugDestination(
00080         QsLogging::DestinationFactory::MakeDebugOutputDestination() );
00081     logger.addDestination(debugDestination);
00082     logger.addDestination(fileDestination);
00083
00084     logger.setLoggingLevel(QsLogging::OffLevel); /*truning logging off*/
00085
00086
00087     /*qInstallMessageHandler(exportlog);*/
00088
00089     /* QLOG_INFO() << "Here is some information";
00090
00091         QLOG_TRACE() << "Here's a" << QString::fromUtf8("trace") << "message";
00092
00093         QLOG_DEBUG() << "Here's a" << static_cast<int>(QsLogging::DebugLevel) << "message";
00094
00095         QLOG_WARN() << "Uh-oh!";
00096
00097         qDebug() << "This message won't be picked up by the logger";
00098
00099         QLOG_ERROR() << "An example error has occurred";
00100
00101         qWarning() << "Neither will this one";
00102
00103         QLOG_FATAL() << "Fatal error example!";*/
00104 }
```

Voici le graphe des appels de cette fonction :



5.89.1.2 void exportlog (QtMsgType *type*, const QMessageLogContext & *context*, const QString & *msg*)

Définition à la ligne 46 du fichier [main.cpp](#).

```

00046
00047     QFile file(QApplication::applicationDirPath() + "/" + qAppName() + ".log");
00048     file.open(QIODevice::WriteOnly | QIODevice::Append);
00049     file.write(QString("[").toUtf8() + QDateTime::currentDateTime().toString().toUtf8() + QString("] ").toUtf8(
00050     ));
00050     QString typeName;
00051     switch (type) {
00052     case QtDebugMsg:
00053         typeName = QObject::tr("Debug");
00054         break;
00055     case QtWarningMsg:
00056         typeName = QObject::tr("Warning");
00057         break;
00058     }
00059     file.write(QObject::tr("%1: %2\r\n").arg(typeName).arg(msg).toUtf8());
00060 }
```

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

Définition à la ligne 125 du fichier [main.cpp](#).

Références [enableLogging\(\)](#), et [SH_MessageManager :errorMessage\(\)](#).

```

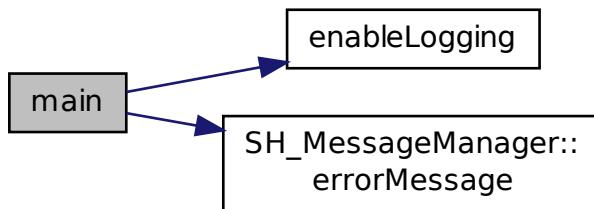
00126 {
00127     try
00128     {
00129         qDebug(); /* Un simple retour à la ligne pour un affichage propre dans la console*/
00130
00131         QApplication app(argc, argv);
00132
00133         const QString sLogPath(QDir::cleanPath(app.applicationDirPath() +
00134             "/../../../../src/PreCheck/debugLog.txt"));
00134         qDebug() << app.applicationDirPath();
00135         qDebug() << QDir::cleanPath(app.applicationDirPath() + "../../../../src/PreCheck/debugLog.txt");
00136         enableLogging(sLogPath);
00137
00138         QString appName = QString(QObject::tr("precheck"));
00139         QString locale = QLocale::system().name();
00140         QTranslator translator;
00141         if (! QFile::exists(appName + "_" + locale + ".qm"))
00142         {
00143             locale = locale.section('_', 0, 0);
00144         }
00145         if ( QFile::exists(appName + "_" + locale + ".qm"))
00146         {
00147             translator.load(appName + "_" + locale);
00148             app.installTranslator(&translator);
00149         }
00150
00151         QQmlEngine engine;
00152
00153         qmlRegisterUncreatableType<SH_ApplicationCore>("PreCheck", 1, 0, "AppMode", "pour enum AppMode");
00154         qmlRegisterType<SH_User>("PreCheck", 1, 0, "User");
00155         SH_ApplicationCore* appManager = new
00156             SH_ApplicationCore();
00157             engine.rootContext()->setContextProperty("SH_App", appManager);
00158
00159             qmlRegisterType<SH_RoomsTableModel>("PreCheck", 1, 0, "SH_RoomsModel");
00160             qmlRegisterType<SH_BillingsTableModel>("PreCheck", 1, 0, "SH_BillingsModel");
00161             qmlRegisterType<SH_BillsTableModel>("PreCheck", 1, 0, "SH_BillsModel");
00162             qmlRegisterType<SH_BookingsTableModel>("PreCheck", 1, 0, "SH_BookingsModel");
00163             qmlRegisterType<SH_ServicesTableModel>("PreCheck", 1, 0, "SH_ServicesModel");
00164             qmlRegisterType<SH_ClientsTableModel>("PreCheck", 1, 0, "SH_ClientsModel");
00165             qmlRegisterType<SH_GroupsTableModel>("PreCheck", 1, 0, "SH_GroupsModel");
00166             qmlRegisterType<SH_SQLDataFields>("PreCheck", 1, 0, "SH_SQLDataField");
00167             qmlRegisterType<SH_ExtendedQQmlAction>("PreCheck", 1, 0, "SH_ComplexAction");
00168
00169             QQmlComponent component(&engine);
00170             component.loadUrl(QUrl("qrc:/qml/SH_app.qml"));
00171             if (!component.isReady())
00172             {
00173                 qWarning("%s", qPrintable(component.errorString()));
00174                 return -1;
00175             }
00176             QObject *topLevel = component.create();
00177             QQuickWindow *window = qobject_cast<QQuickWindow *>(topLevel);
00178             if (!window)
00179             {
00180                 qWarning("Error: Your root item has to be a Window.");
00181                 return -1;
00182             }
00183             QObject::connect(&engine, SIGNAL(quit()), &app, SLOT(quit()));

```

```

00183
00184
00185     QObject * commonPage = window->findChild<QObject *>("Common");
00186     QObject * tabsZone = commonPage->findChild<QObject *>("TabView");
00187     QObject * displayZone = commonPage->findChild<QObject *>("RightOutput");
00188
00189     QObject::connect(appManager, SIGNAL(openTab(QVariant)), tabsZone, SLOT(openTab(QVariant)),
00190         Qt::DirectConnection);
00190     QObject::connect(appManager, SIGNAL(sendText(QString)), displayZone, SIGNAL(displayNewFixed(QString
00191         )), Qt::DirectConnection);
00191     QObject::connect(appManager, SIGNAL(sendText(QString)), displayZone, SIGNAL(replace(QString)),
00192         Qt::DirectConnection);
00192     QObject::connect(appManager, SIGNAL(clearAll()), displayZone, SLOT(clearAll()),
00193         Qt::QueuedConnection);
00193     /*QObject::connect(appManager, SIGNAL(displayCalendar()), displayZone, SLOT(displayCalendar())),
00194         Qt::DirectConnection);*/
00194
00195     window->show();
00196     QLOG_INFO() << "Program built with Qt" << QT_VERSION_STR << "running on" << qVersion();
00197     return app.exec();
00198 }
00199
00200     catch (const std::exception &e)
00201 {
00202     SH_MessageManager::errorMessage(e.what());
00203 }
00204 }
```

Voici le graphe d'appel pour cette fonction :



5.89.1.4 void spin(int & iteration)

Définition à la ligne 108 du fichier [main.cpp](#).

```

00109 {
00110     const int work = 1000 * 1000 * 40;
00111     volatile int v = 0;
00112     for (int j = 0; j < work; ++j)
00113         ++v;
00114
00115     qDebug() << "iteration" << iteration << "in thread" << QThread::currentThreadId();
00116 }
```

5.89.1.5 void statusChanged(QQmlComponent * component, QQmlComponent::Status status)

Définition à la ligne 31 du fichier [main.cpp](#).

```

00031     if (status == QQmlComponent::Error) {
00032         foreach (const QQmlError &error, component->errors()) {
00033             const QByteArray file = error.url().toEncoded();
00034             QMessageLogger(file.constData(), error.line(), 0).debug() << error.description();
00035         }
00036     }
00037 }
00038 }
```

5.89.2 Documentation des variables

5.89.2.1 const int iterations = 20

Définition à la ligne 23 du fichier [main.cpp](#).

5.90 main.cpp

```

00001 #include <QtWidgets/QApplication>
00002 #include <QtQml>
00003 #include <QtQuick>
00004 #include <QProgressDialog>
00005 #include "SH_MessageManager.h"
00006 #include "SH_ApplicationCore.h"
00007 #include "models/SH_ExtendedSqlProxyModel.h"
00008 #include "models/SH_RoomsTableModel.h"
00009 #include "models/SH_SqlDataField.h"
00010 #include "models/SH_BillingsTableModel.h"
00011 #include "models/SH_BillsTableModel.h"
00012 #include "models/SH_BookingsTableModel.h"
00013 #include "models/SH_RoomsTableModel.h"
00014 #include "models/SH_ClientsTableModel.h"
00015 #include "models/SH_ServicesTableModel.h"
00016 #include "models/SH_GroupsTableModel.h"
00017 #include "models/SH_User.h"
00018 #include "views/SH_ExtendedQQmlAction.h"
00019 #include "logging/QsLog.h"
00020 #include "logging/QsLogDest.h"
00021
00022
00023 const int iterations = 20;
00024
00025
00031 void statusChanged(QQmlComponent* component, QQmlComponent::Status status) {
00032     if (status == QQmlComponent::Error) {
00033         foreach (const QQmlError &error, component->errors()) {
00034             const QByteArray file = error.url().toEncoded();
00035             QMessageLogger(file.constData(), error.line(), 0).debug() << error.description();
00036         }
00037     }
00038 }
00039
00046 void exportlog(QtMsgType type, const QMessageLogContext &context, const QString &msg) {
00047     QFile file(QApplication::applicationDirPath() + "/" + qAppName() + ".log");
00048     file.open(QIODevice::WriteOnly | QIODevice::Append);
00049     file.write(QString("[%").toUtf8() + QDateTime::currentDateTime().toString().toUtf8() + QString("]").toUtf8());
00050     QString typeName;
00051     switch (type) {
00052     case QtDebugMsg:
00053         typeName = QObject::tr("Debug");
00054         break;
00055     case QtWarningMsg:
00056         typeName = QObject::tr("Warning");
00057         break;
00058     }
00059     file.write(QObject::tr("%1: %2\r\n").arg(typeName).arg(msg).toUtf8());
00060 }
00061
00062
00071 void enableLogging(const QString sLogPath)
00072 {
00073     /* init the logging mechanism*/
00074     QsLogging::Logger& logger = QsLogging::Logger::instance();
00075     logger.setLoggingLevel(QsLogging::TraceLevel);
00076
00077     QsLogging::DestinationPtr fileDestination(
00078         QsLogging::DestinationFactory::MakeFileDestination(sLogPath, true, 512, 2));
00079     QsLogging::DestinationPtr debugDestination(
00080         QsLogging::DestinationFactory::MakeDebugOutputDestination());
00081     logger.addDestination(debugDestination);
00082     logger.addDestination(fileDestination);
00083
00084     logger.setLoggingLevel(QsLogging::OffLevel); /*truning logging off*/
00085
00086
00087     /*qInstallMessageHandler(exportlog);*/
00088
00089     /* QLOG_INFO() << "Here is some information";
00090      QLOG_TRACE() << "Here's a" << QString::fromUtf8("trace") << "message";
```

```

00092     QLOG_DEBUG() << "Here's a" << static_cast<int>(QsLogging::DebugLevel) << "message";
00093     QLOG_WARN() << "Uh-oh!";
00094     qDebug() << "This message won't be picked up by the logger";
00095     QLOG_ERROR() << "An example error has occurred";
00096     qWarning() << "Neither will this one";
00097     QLOG_FATAL() << "Fatal error example!";*/
00098 }
00099
00100
00101
00102
00103 void spin(int &iteration)
00104 {
00105     const int work = 1000 * 1000 * 40;
00106     volatile int v = 0;
00107     for (int j = 0; j < work; ++j)
00108         ++v;
00109
00110     qDebug() << "iteration" << iteration << "in thread" << QThread::currentThreadId();
00111 }
00112
00113 int main(int argc, char **argv)
00114 {
00115     try
00116     {
00117         qDebug(); /* Un simple retour à la ligne pour un affichage propre dans la console*/
00118
00119         QApplication app(argc, argv);
00120
00121         const QString sLogPath(QDir::cleanPath(app.applicationDirPath()+"../../../../src/PreCheck/debugLog.txt"));
00122         qDebug() << app.applicationDirPath();
00123         qDebug() << QDir::cleanPath(app.applicationDirPath()+"../../../../src/PreCheck/debugLog.txt");
00124         enableLogging(sLogPath);
00125
00126         QString appName = QString(QObject::tr("precheck"));
00127         QString locale = QLocale::system().name();
00128         QTranslator translator;
00129         if (! QFile::exists(appName + "_" + locale + ".qm"))
00130         {
00131             locale = locale.section('_', 0, 0);
00132         }
00133         if ( QFile::exists(appName + "_" + locale + ".qm"))
00134         {
00135             translator.load(appName + "_" + locale);
00136             app.installTranslator(&translator);
00137         }
00138
00139         QQmlEngine engine;
00140
00141        qmlRegisterUncreatableType<SH_ApplicationCore>("PreCheck", 1, 0, "AppMode","pour enum AppMode");
00142         qmlRegisterType<SH_User>("PreCheck", 1, 0, "User");
00143         SH_ApplicationCore* appManager = new
00144             SH_ApplicationCore();
00145         engine.rootContext()->setContextProperty("SH_App", appManager);
00146
00147         qmlRegisterType<SH_RoomsTableModel>("PreCheck", 1, 0, "SH_RoomsModel");
00148         qmlRegisterType<SH_BillingsTableModel>("PreCheck", 1, 0, "SH_BillingsModel");
00149         qmlRegisterType<SH_BillsTableModel>("PreCheck", 1, 0, "SH_BillsModel");
00150         qmlRegisterType<SH_BookingsTableModel>("PreCheck", 1, 0, "SH_BookingsModel");
00151         qmlRegisterType<SH_ServicesTableModel>("PreCheck", 1, 0, "SH_ServicesModel");
00152         qmlRegisterType<SH_ClientsTableModel>("PreCheck", 1, 0, "SH_ClientsModel");
00153         qmlRegisterType<SH_GroupsTableModel>("PreCheck", 1, 0, "SH_GroupsModel");
00154         qmlRegisterType<SH_SQLDataFields>("PreCheck", 1, 0, "SH_SQLDataField");
00155         qmlRegisterType<SH_ExtendedQQmlAction>("PreCheck", 1, 0, "SH_ComplexAction");
00156
00157         QQmlComponent component(&engine);
00158         component.loadUrl(QUrl("qrc:/qml/SH_app.qml"));
00159         if (!component.isReady())
00160         {
00161             qWarning("%s", qPrintable(component.errorString()));
00162             return -1;
00163         }
00164         QObject *topLevel = component.create();
00165         QQuickWindow *window = qobject_cast<QQuickWindow *>(topLevel);
00166         if (!window)
00167         {
00168             qWarning("Error: Your root item has to be a Window.");
00169             return -1;
00170         }
00171         QObject::connect(&engine, SIGNAL(quit()), &app, SLOT(quit()));
00172
00173         QObject * commonPage = window->findChild<QObject *>("Common");
00174         QObject * tabsZone = commonPage->findChild<QObject *>("TabView");
00175         QObject * displayZone = commonPage->findChild<QObject *>("RightOutput");
00176
00177
00178
00179
00180
00181
00182
00183
00184
00185
00186
00187
00188

```

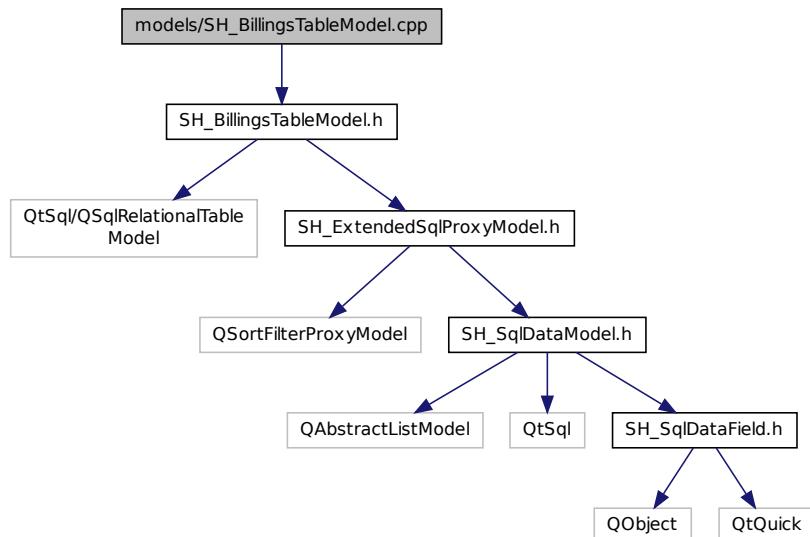
```

00189     QObject::connect(appManager, SIGNAL(openTab(QVariant)), tabsZone, SLOT(openTab(QVariant)),
00190     Qt::DirectConnection);
00190     QObject::connect(appManager, SIGNAL(sendText(QString)), displayZone, SIGNAL(displayNewFixed(QString
00191     )), Qt::DirectConnection);
00191     QObject::connect(appManager, SIGNAL(sendText(QString)), displayZone, SIGNAL(replace(QString)),
00192     Qt::DirectConnection);
00192     QObject::connect(appManager, SIGNAL(clearAll()), displayZone, SLOT(clearAll()),
00193     Qt::QueuedConnection);
00193     /*QObject::connect(appManager, SIGNAL(displayCalendar()), displayZone, SLOT(displayCalendar()),
00194     Qt::DirectConnection);*/
00194
00195     window->show();
00196     QLOG_INFO() << "Program built with Qt" << QT_VERSION_STR << "running on" << qVersion();
00197     return app.exec();
00198 }
00199
00200 catch (const std::exception &e)
00201 {
00202     SH_MessageManager::errorMessage(e.what());
00203 }
00204 }
```

5.91 Référence du fichier models/SH_BillingsTableModel.cpp

#include "SH_BillingsTableModel.h"

Graphe des dépendances par inclusion de SH_BillingsTableModel.cpp :



5.92 SH_BillingsTableModel.cpp

```

00001 #include "SH_BillingsTableModel.h"
00002
00003
00004
0010 SH_BillingsTableModel::SH_BillingsTableModel(
0011     QObject *parent):
0011     SH_ExtendedProxyModel(parent)
0012 {
0013     SH_ExtendedProxyModel::model->setTable("BILLINGSINFOS");
0014 }
0015
0016
0022 void SH_BillingsTableModel::fillModel()
0023 {
0024     SH_ExtendedProxyModel::model->setHeaderData(1, Qt::Horizontal,
  
```

```

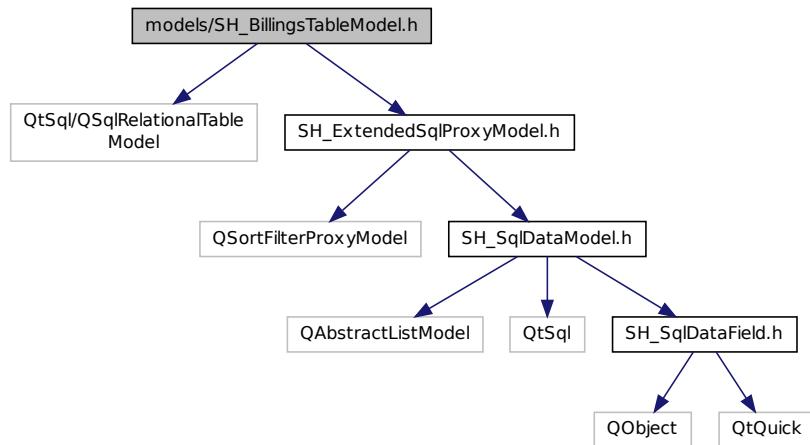
00025     SH_ExtendedProxyModel::model->setHeaderData(3, Qt::Horizontal,
00026         QObject::tr("Chambre"));
00027     SH_ExtendedProxyModel::model->setHeaderData(4, Qt::Horizontal,
00028         QObject::tr("Date arrivée"));
00029     SH_ExtendedProxyModel::model->setHeaderData(5, Qt::Horizontal,
00030         QObject::tr("Date départ prévue"));
00031     SH_ExtendedProxyModel::model->field(4)->
00032         setSortOrder(Qt::AscendingOrder);
00033 }

```

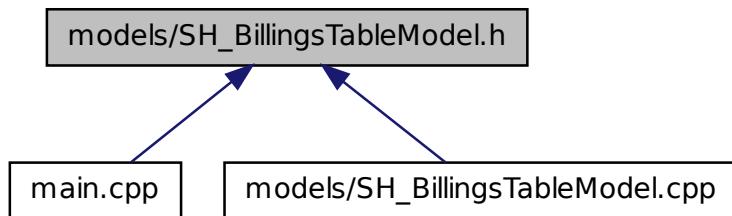
5.93 Référence du fichier models/SH_BillingsTableModel.h

```
#include <QtSql/ QSqlRelationalTableModel>
#include "SH_ExtendedSqlProxyModel.h"
```

Graphe des dépendances par inclusion de SH_BillingsTableModel.h :



Ce graphe montre quels fichiers incluent directement ou indirectement ce fichier :



Classes

- class [SH_BillingsTableModel](#)

5.94 SH_BillingsTableModel.h

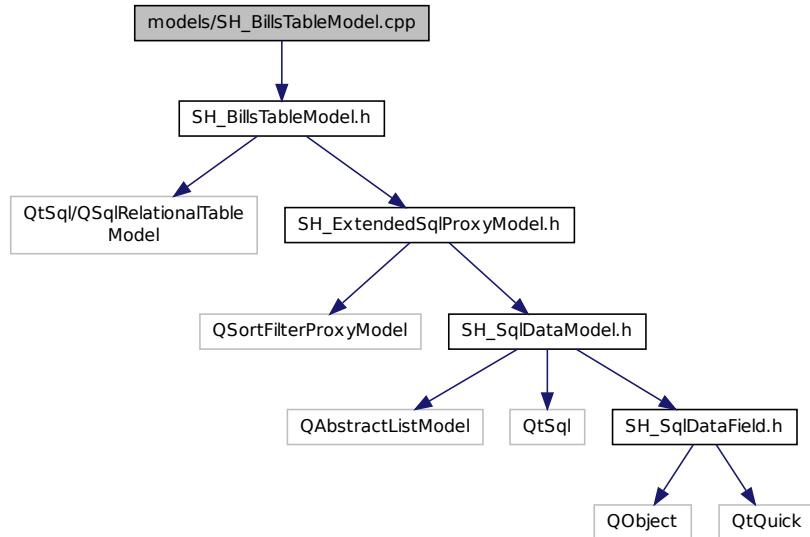
```

00001 #ifndef RENT_H
00002 #define RENT_H
00003 #include <QtSql/ QSqlRelationalTableModel>
00004 #include "SH_ExtendedSqlProxyModel.h"
00005
00006
00007
0013 class SH_BillingsTableModel : public SH_ExtendedProxyModel
0014 {
0015     Q_OBJECT
0016 public:
0017
0018
0025     SH_BillingsTableModel(QObject *parent = 0);
0026
0027 protected:
0028
0029
0035     void fillModel();
0036
0037 };
0038
0039 #endif /* RENT_H */

```

5.95 Référence du fichier models/SH_BillsTableModel.cpp

#include "SH_BillsTableModel.h"
Graphe des dépendances par inclusion de SH_BillsTableModel.cpp :



5.96 SH_BillsTableModel.cpp

```

00001 #include "SH_BillsTableModel.h"
00002
00003
00009 SH_BillsTableModel::SH_BillsTableModel(
    QObject *parent):
0010     SH_ExtendedProxyModel(parent)
0011 {
0012     SH_ExtendedProxyModel::model->setTable("BILLS");
0013 }
0014

```

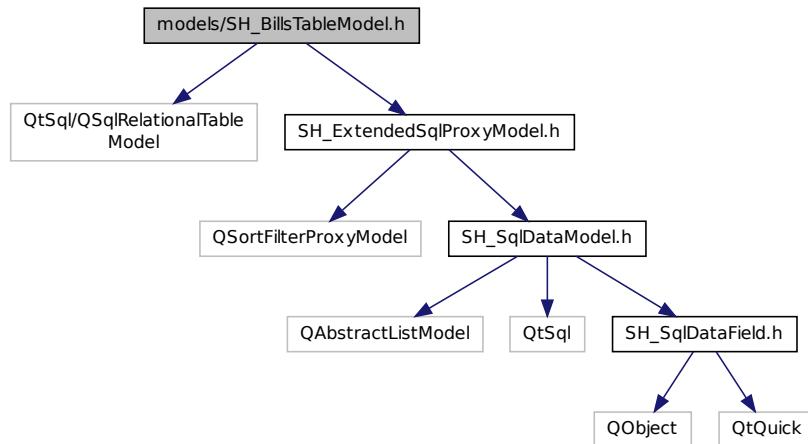
```

00015
00021 void SH_BillsTableModel::fillModel()
00022 {
00023 }
```

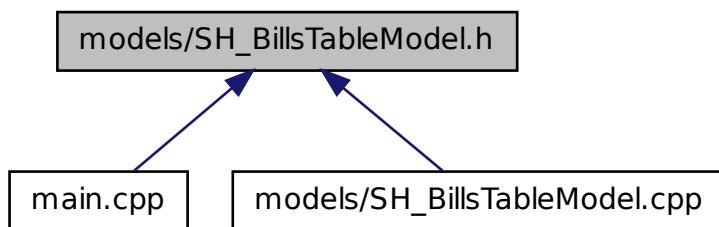
5.97 Référence du fichier models/SH_BillsTableModel.h

```
#include <QtSql/ QSqlRelationalTableModel>
#include "SH_ExtendedSqlProxyModel.h"
```

Graphe des dépendances par inclusion de SH_BillsTableModel.h :



Ce graphe montre quels fichiers incluent directement ou indirectement ce fichier :



Classes

- class `SH_BillsTableModel`

5.98 SH_BillsTableModel.h

```
00001 #ifndef BILLS_H
```

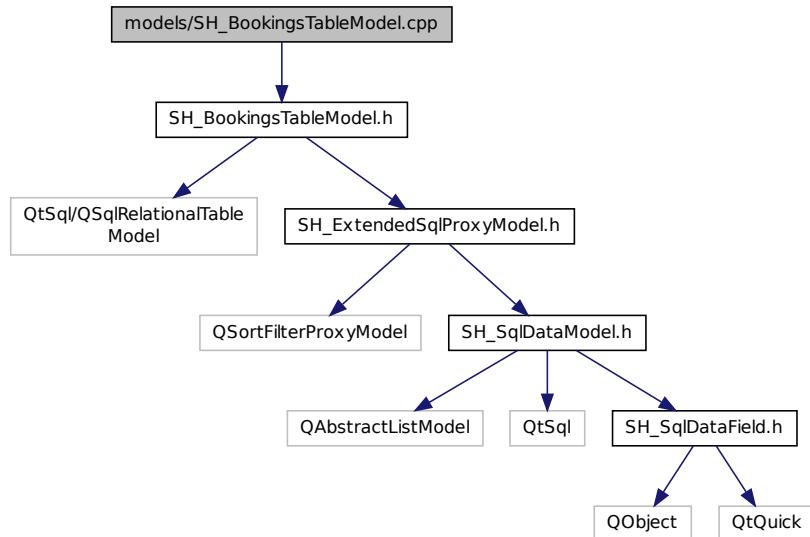
```

00002 #define BILLS_H
00003 #include <QtSql/ QSqlRelationalTableModel>
00004 #include "SH_ExtendedSqlProxyModel.h"
00005
00006
00007
00013 class SH_BillsTableModel : public SH_ExtendedProxyModel
00014 {
00015     Q_OBJECT
00016
00017     public:
00018         SH_BillsTableModel(QObject *parent = 0);
00026
00027     protected:
00028
00029         void fillModel();
00035
00036 };
00037
00038 #endif /* BILLS_H*/

```

5.99 Référence du fichier models/SH_BookingsTableModel.cpp

#include "SH_BookingsTableModel.h"
Graphe des dépendances par inclusion de SH_BookingsTableModel.cpp :



5.100 SH_BookingsTableModel.cpp

```

00001 #include "SH_BookingsTableModel.h"
00002
00003
00009 SH_BookingsTableModel::SH_BookingsTableModel(
    QObject *parent):
00010     SH_ExtendedProxyModel(parent)
00011 {
00012     SH_ExtendedProxyModel::model->setTable("BOOKINGS");
00013     SH_ExtendedProxyModel::model->setFilterCondition(
        QObject::tr("ISCONFIRMED") + "'='1'");
00014 }
00015
00016
00022 void SH_BookingsTableModel::fillModel()
00023 {

```

```

00024     SH_ExtendedProxyModel::model->setHeaderData(0, Qt::Horizontal,
00025         QObject::tr("Date réservation"));
00026     SH_ExtendedProxyModel::model->setHeaderData(1, Qt::Horizontal,
00027         QObject::tr("Nom client"));
00028     SH_ExtendedProxyModel::model->setHeaderData(2, Qt::Horizontal,
00029         QObject::tr("Date arrivée prévue"));
00030     SH_ExtendedProxyModel::model->setHeaderData(3, Qt::Horizontal,
00031         QObject::tr("Nb Personnes"));
00032
00029     SH_ExtendedProxyModel::model->field(0)->
00030         setSortOrder(Qt::AscendingOrder);
00030 }

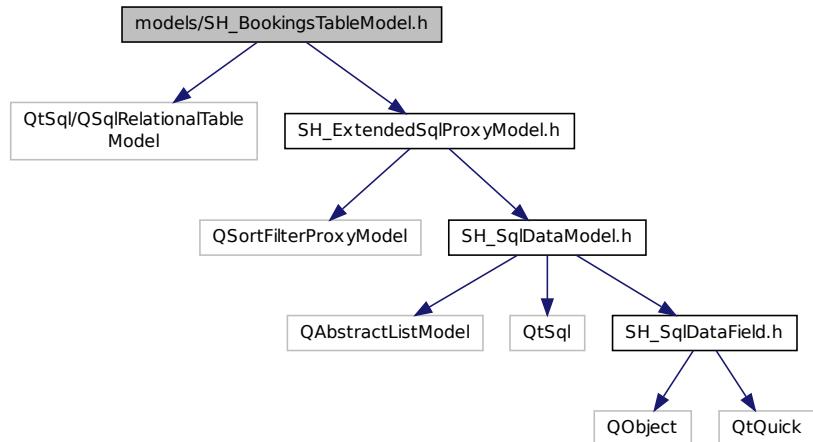
```

5.101 Référence du fichier models/SH_BookingsTableModel.h

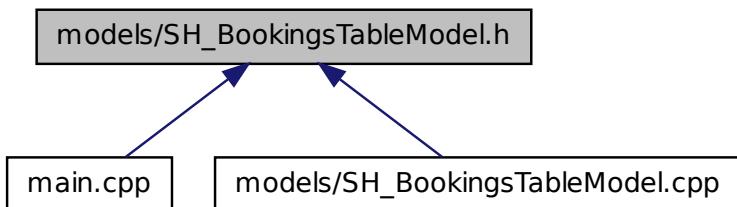
```
#include <QtSql/QSqlRelationalTableModel>
```

```
#include "SH_ExtendedSqlProxyModel.h"
```

Graphe des dépendances par inclusion de SH_BookingsTableModel.h :



Ce graphe montre quels fichiers incluent directement ou indirectement ce fichier :



Classes

- class `SH_BookingsTableModel`

5.102 SH_BookingsTableModel.h

```

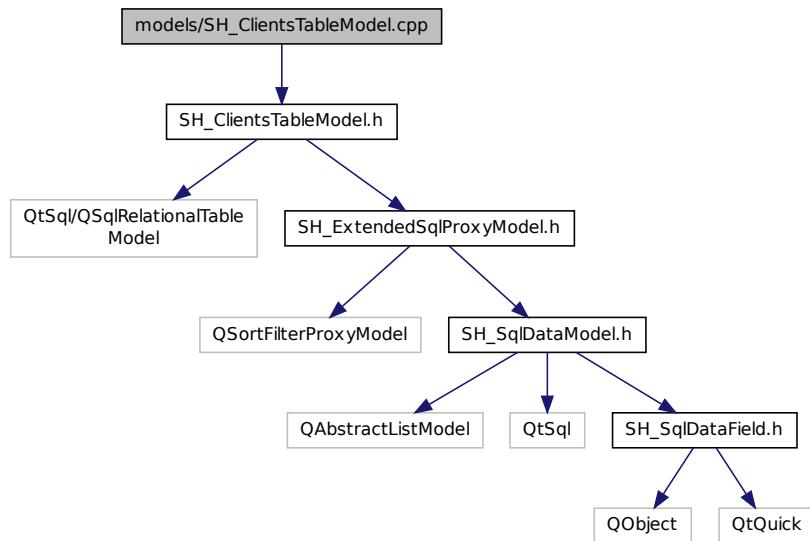
00001 #include <QtSql/QSqlRelationalTableModel>
00002 #ifndef BOOKING_H
00003 #define BOOKING_H
00004
00005 #include "SH_ExtendedSqlProxyModel.h"
00006
00007
00008
00014 class SH_BookingsTableModel : public SH_ExtendedProxyModel
00015 {
00016     Q_OBJECT
00017     public:
00018
00019         SH_BookingsTableModel(QObject *parent = 0);
00020
00021
00029     protected:
00030
00031         void fillModel();
00038 };
00039
00040 #endif /* BOOKING_H*/

```

5.103 Référence du fichier models/SH_ClientsTableModel.cpp

#include "SH_ClientsTableModel.h"

Graphe des dépendances par inclusion de SH_ClientsTableModel.cpp :



5.104 SH_ClientsTableModel.cpp

```

00001 #include "SH_ClientsTableModel.h"
00002
00003
00009 SH_ClientsTableModel::SH_ClientsTableModel(
00010     QObject *parent):
00011     SH_ExtendedProxyModel(parent)
00012 {
00013     SH_ExtendedProxyModel::model->setTable("CLIENTS");
00014 }

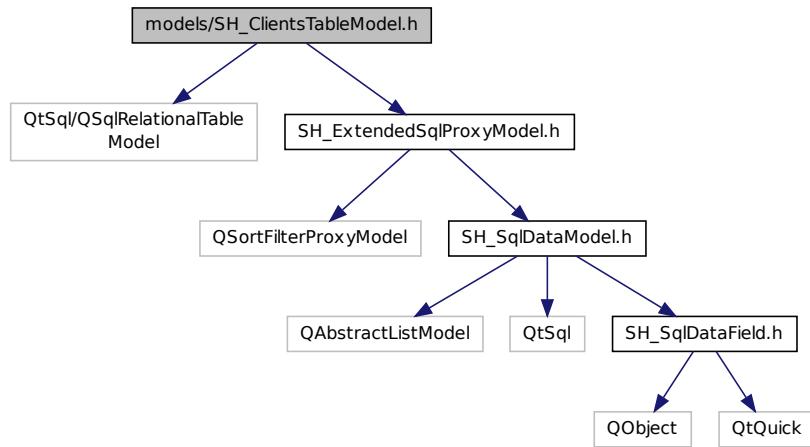
```

```

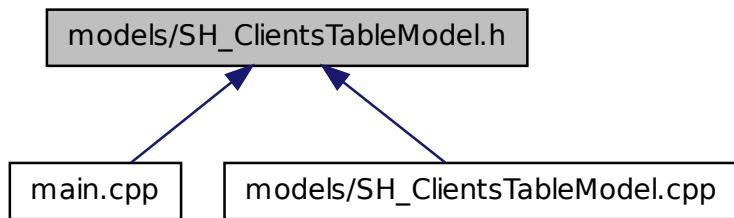
00014
00015
00021 void SH_ClientsTableModel::fillModel()
00022 {
00023 }
```

5.105 Référence du fichier models/SH_ClientsTableModel.h

```
#include <QtSql/QSqlRelationalTableModel>
#include "SH_ExtendedSqlProxyModel.h"
Graphe des dépendances par inclusion de SH_ClientsTableModel.h :
```



Ce graphe montre quels fichiers incluent directement ou indirectement ce fichier :



Classes

- class `SH_ClientsTableModel`

5.106 SH_ClientsTableModel.h

```
00001 #include <QtSql/QSqlRelationalTableModel>
```

```

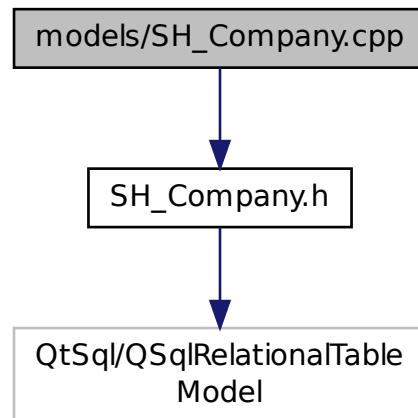
00002 #ifndef CLIENT_H
00003 #define CLIENT_H
00004
00005 #include "SH_ExtendedSqlProxyModel.h"
00006
00007
00008
0014 class SH_ClientsTableModel : public SH_ExtendedProxyModel
0015 {
0016     Q_OBJECT
0017     public:
0018
0019
0026         SH_ClientsTableModel(QObject *parent = 0);
0027
0028
0029     protected:
0030
0031         void fillModel();
0038 };
0039
0040 #endif /* CLIENT_H*/

```

5.107 Référence du fichier models/SH_Company.cpp

#include "SH_Company.h"

Graphe des dépendances par inclusion de SH_Company.cpp :



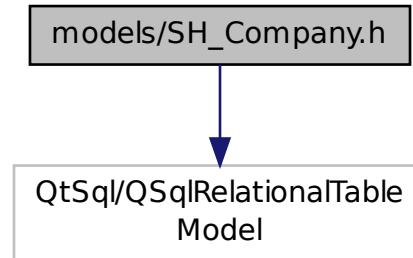
5.108 SH_Company.cpp

```
00001 #include "SH_Company.h"
```

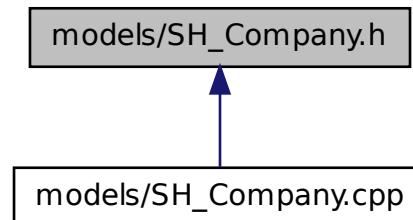
5.109 Référence du fichier models/SH_Company.h

```
#include <QtSql/QSqlRelationalTableModel>
```

Graphe des dépendances par inclusion de SH_Company.h :



Ce graphe montre quels fichiers incluent directement ou indirectement ce fichier :



Classes

- class [SH_Company](#)

5.110 SH_Company.h

```

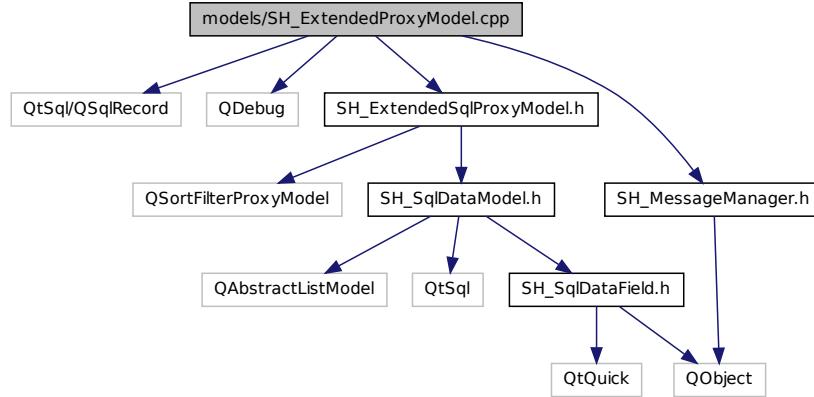
00001 #include <QtSql/QSqlRelationalTableModel>
00002 #ifndef COMPANY_H
00003 #define COMPANY_H
00004
00005
00012 class SH_Company : public QObject
00013 {
00014     Q_OBJECT
00015 };
00016
00017 #endif /* COMPANY_H*/
  
```

5.111 Référence du fichier models/SH_ExtendedProxyModel.cpp

```
#include <QtSql/QSqlRecord>
```

```
#include <QDebug>
#include "SH_ExtendedSqlProxyModel.h"
#include "SH_MessageManager.h"
```

Graphe des dépendances par inclusion de SH_ExtendedProxyModel.cpp :



5.112 SH_ExtendedProxyModel.cpp

```

00001 #include <QtSql/QSqlRecord>
00002 #include <QDebug>
00003 #include "SH_ExtendedSqlProxyModel.h"
00004 #include "SH_MessageManager.h"
00005
00006
00012 SH_ExtendedProxyModel::SH_ExtendedProxyModel(
    QObject *parent) :
00013     QSortFilterProxyModel(parent)
00014 {
00015     this->setDynamicSortFilter(false);
00016     this->model = new SH_SqlDataModel(parent);
00017     this->setSourceModel(this->model);
00018     this->sortIndex = 0;
00019     /*connect(this->model, tableChanged(), tableName());*/
00020 }
00021
00027 void SH_ExtendedProxyModel::replaceSet(QList<int>& originalSet, QList<int>
newSet) {
00028     originalSet.clear();
00029     foreach(int col, newSet) {
00030         if(!originalSet.contains(col)) {
00031             originalSet.append(col);
00032         }
00033     }
00034 }
00035
00041 void SH_ExtendedProxyModel::setBooleanColumns(QList<int> boolCols)
{
00042     replaceSet(this->booleanSet, boolCols);
00043 }
00044
00050 void SH_ExtendedProxyModel::setReadOnlyColumns(QList<int>
readonlyCols) {
00051     replaceSet(this->readonlySet, readonlyCols);
00052 }
00053
00059 void SH_ExtendedProxyModel::setPasswordColumns(QList<int>
passwordCols) {
00060     replaceSet(this->passwordSet, passwordCols);
00061 }
00062
00068 void SH_ExtendedProxyModel::setNullColumns(QList<int> nullCols) {
00069     if (sourceModel()->inherits(" QSqlQueryModel")) {
00070         replaceSet(this->nullSet, nullCols);
00071     }
00072 }
```

```

00073
00080 void SH_ExtendedProxyModel::setNotNullColumns(QList<int>
00081     notNullCols) {
00082     if (sourceModel() ->inherits("QSqlQueryModel")) {
00083         replaceSet(this->notNullSet, notNullCols);
00084     }
00085
00086
00092 bool SH_ExtendedProxyModel::filterAcceptsRow(int source_row, const
00093     QModelIndex &source_parent) const
00094 {
00095     Q_UNUSED(source_parent);
00096     if (!this->notNullSet.isEmpty())
00097     {
00098         QSqlQueryModel *m = static_cast<QSqlQueryModel *>(sourceModel());
00099         foreach(int column, this->notNullSet)
00100         {
00101             if (m->record(source_row).isNull(column))
00102             {
00103                 return false;
00104             }
00105         }
00106     }
00107
00108     if (!this->nullSet.isEmpty())
00109     {
00110         QSqlQueryModel *m = static_cast<QSqlQueryModel *>(sourceModel());
00111         foreach(int column, this->nullSet)
00112         {
00113             if (!m->record(source_row).isNull(column))
00114             {
00115                 return false;
00116             }
00117         }
00118     }
00119     return true;
00120 }
00121
00127 QVariant SH_ExtendedProxyModel::data(const QModelIndex &index, int role) const
00128 {
00129     if (index.isValid())
00130     {
00131         if (this->booleanSet.contains(role))
00132         {
00133             return index.data(Qt::EditRole).toBool() ? QVariant(Qt::Checked) : QVariant(Qt::Unchecked);
00134         }
00135         else if (this->passwordSet.contains(role))
00136         {
00137             return QVariant("****");
00138         }
00139         else if (!this->filters.contains(role))
00140         {
00141             QModelIndex source_index = QSortFilterProxyModel::mapToSource(index);
00142             if (source_index.isValid())
00143                 return this->model->data(source_index, role);
00144         }
00145     }
00146 }
00147 return QVariant();
00148 }
00149
00150
00156 bool SH_ExtendedProxyModel::setData(const QModelIndex &index, const QVariant
00157     &value, int role)
00158 {
00159     if (!index.isValid())
00160         return false;
00161
00162     if (this->booleanSet.contains(role))
00163     {
00164         QVariant data = (value.toInt() == Qt::Checked) ? QVariant(1) : QVariant(0);
00165         return QSortFilterProxyModel::setData(index, data, role);
00166     }
00167     else
00168     {
00169         return QSortFilterProxyModel::setData(index, value, role);
00170     }
00171 }
00172
00173
00179 Qt::ItemFlags SH_ExtendedProxyModel::flags(const QModelIndex &index) const
00180 {
00181     if (!index.isValid())
00182     {

```

```

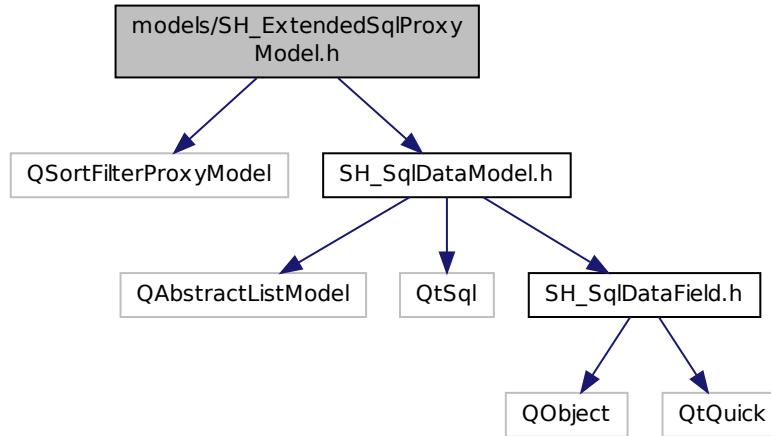
00183     return Qt::ItemIsEnabled;
00184 }
00185 if (!this->booleanSet.isEmpty())
00186 {
00187     return Qt::ItemIsUserCheckable | Qt::ItemIsSelectable | Qt::ItemIsEnabled;
00188 }
00189 else if (!this->readonlySet.isEmpty())
00190 {
00191     return Qt::ItemIsSelectable;
00192 }
00193 else
00194 {
00195     return QSortFilterProxyModel::flags(index);
00196 }
00197
00198 }
00199
00200 void SH_ExtendedProxyModel::invalidateFilter()
00201 {
00202     this->filters.clear();
00203 }
00204
00205 void SH_ExtendedProxyModel::removeFilterKeyColumn(int column)
00206 {
00207     this->filters.removeAt(this->filters.indexOf(column));
00208 }
00209
00210 bool SH_ExtendedProxyModel::containsFilterKeyColumn(int
00211 column)
00212 {
00213     return this->filters.contains(column);
00214 }
00215
00216 void SH_ExtendedProxyModel::sort(int column, Qt::SortOrder newOrder)
00217 {
00218     this->model->field(column)->setSortOrder(newOrder);
00219     SH_ExtendedProxyModel::setSortKeyColumn(column);
00220 }
00221
00222 void SH_ExtendedProxyModel::setSortKeyColumn(int column)
00223 {
00224     this->sortIndex = column;
00225     QSortFilterProxyModel::setSortRole(this->model->roleForField(column));
00226     QSortFilterProxyModel::sort(0, this->model->field(column)->
00227         sortOrder());
00228     emit sortChanged();
00229 }
00230
00231 void SH_ExtendedProxyModel::addFilterKeyColumn(int column)
00232 {
00233     this->filters.append(column);
00234 }
00235
00236 QVariant SH_ExtendedProxyModel::data(int row, int column) const
00237 {
00238     QModelIndex modelIndex = this->index(row, 0);
00239     return this->data(modelIndex, this->model->roleForField(column));
00240 }
00241
00242 bool SH_ExtendedProxyModel::fetch(QString tableName, QString filter, QString
00243 sort, QStringList fields)
00244 {
00245     bool fetched = this->model->fetch(tableName, filter, sort, fields);
00246     if (fetched)
00247     {
00248         this->fillModel();
00249     }
00250     this->setSourceModel(this->model);
00251     return fetched;
00252 }
00253
00254
00255
00256
00257
00258
00259
00260
00261
00262
00263
00264
00265
00266
00267
00268
00269
00270
00271
00272
00273
00274
00275
00276
00277
00278
00279
00280
00281
00282
00283
00284
00285
00286
00287
00288
00289
00290
00291
00292

```

5.113 Référence du fichier models/SH_ExtendedSqlProxyModel.h

```
#include <QSortFilterProxyModel>
#include "SH_SqlDataModel.h"
```

Graphe des dépendances par inclusion de SH_ExtendedSqlProxyModel.h :



Ce graphe montre quels fichiers incluent directement ou indirectement ce fichier :



Classes

- class [SH_ExtendedProxyModel](#)

5.114 SH_ExtendedSqlProxyModel.h

```

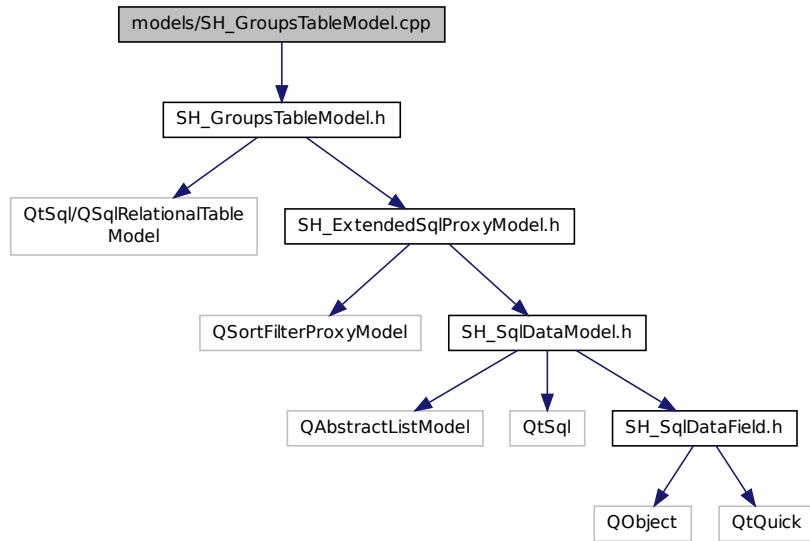
00001 #ifndef CHECKABLESORTFILTERPROXYMODEL_H
00002 #define CHECKABLESORTFILTERPROXYMODEL_H
00003
00004 #include <QSortFilterProxyModel>
00005 #include "SH_SqlDataModel.h"
00006
00013 class SH_ExtendedProxyModel : public QSortFilterProxyModel
00014 {
00015
00016     Q_OBJECT
00017     Q_PROPERTY(QString tableName READ tableName)
00018     Q_PROPERTY(QString fieldsList READ fields)
00019     Q_PROPERTY(QString lastError READ lastError)
00020     Q_PROPERTY(int sortKeyColumn READ currentSortKeyColumn WRITE
00021         setSortKeyColumn NOTIFY sortChanged)
00022     Q_PROPERTY(bool empty READ isEmpty)
00023
00024 public:
00025     SH_ExtendedProxyModel(QObject *parent = 0);
00026
00027     const int currentSortKeyColumn() const {return this->
00028         sortIndex;}
00029     const QString tableName() const { return this->model->
00030         tableName(); }
00031     const QString fields() const { if(this->model->fieldsList().isEmpty()) { return "*" }
00032     ;} else { return this->model->fieldsList().join(", "); } }
00033     const QString lastError() const { return this->model->
00034         lastError(); }
00035     const bool isEmpty() const { return this->model->isEmpty(); }
  
```

```
00073     void setSortKeyColumn(int column);
00074
00082     Q_INVOKABLE SH_SqlDataFields *field(int i) const { return this->
00083         model->field(i); }
00089     Q_INVOKABLE int fieldsCount() const { return this->model->
00090         fieldsCount(); }
00100     Q_INVOKABLE bool fetch(QString tableName = "", QString filter = "", QString
00101         sort = "", QStringList fields = QStringList());
00108     Q_INVOKABLE void sort(int column, Qt::SortOrder newOrder = Qt::AscendingOrder);
00115     Q_INVOKABLE void addFilterKeyColumn(int column);
00122     Q_INVOKABLE void removeFilterKeyColumn(int column);
00130     Q_INVOKABLE bool containsFilterKeyColumn(int column);
00139     Q_INVOKABLE QVariant data(int row, int column) const;
00148     Q_INVOKABLE QVariant data(const QModelIndex &index, int role = Qt::DisplayRole) const;
00158     Q_INVOKABLE bool setData(const QModelIndex &index, const QVariant &value, int role =
00159         Qt::EditRole);
00165     Q_INVOKABLE virtual QHash<int, QByteArray> roleNames() const { return this->
00166         model->roleNames(); }
00173     Q_INVOKABLE Qt::ItemFlags flags(const QModelIndex &index) const;
00174
00180     void invalidateFilter();
00187     void setBooleanColumns(QList<int> boolCols);
00194     void setReadOnlyColumns(QList<int> readonlyCols);
00201     void setPasswordColumns(QList<int> passwordCols);
00208     void setNullColumns(QList<int> nullCols);
00215     void setNotNullColumns(QList<int> notNullCols);
00216
00217 signals:
00223     void sortChanged();
00224
00225 protected:
00231     virtual void fillModel() = 0;
00240     bool filterAcceptsRow(int source_row, const QModelIndex &source_parent) const;
00241     SH_SqlDataModel *model;
00242
00243 private:
00251     void replaceSet(QList<int>& originalSet, QList<int> newSet);
00255     QList<int> booleanSet;
00259     QList<int> passwordSet;
00263     QList<int> readonlySet;
00267     QList<int> notNullSet;
00271     QList<int> nullSet;
00275     QList<int> filters;
00279     int sortIndex;
00280 };
00281 #endif
```

5.115 Référence du fichier models/SH_GroupsTableModel.cpp

```
#include "SH_GroupsTableModel.h"
```

Graphe des dépendances par inclusion de SH_GroupsTableModel.cpp :



5.116 SH_GroupsTableModel.cpp

```

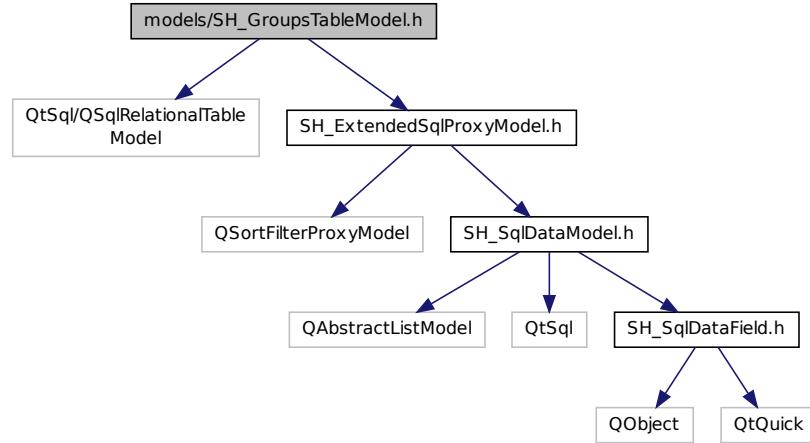
00001 #include "SH_GroupsTableModel.h"
00002
00003
00009 SH_GroupsTableModel::SH_GroupsTableModel(
    QObject *parent):
00010     SH_ExtendedProxyModel(parent)
00011 {
00012     SH_ExtendedProxyModel::model->setTable("GROUPS");
00013 }
00014
00020 void SH_GroupsTableModel::fillModel()
00021 {
00022 }
  
```

5.117 Référence du fichier models/SH_GroupsTableModel.h

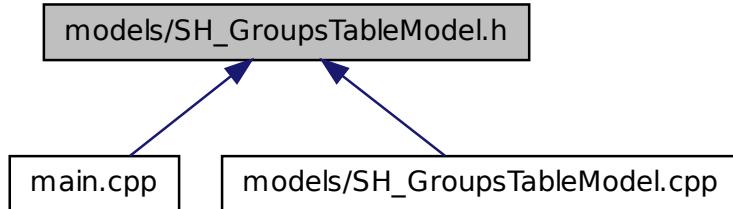
```

#include <QtSql/ QSqlRelationalTableModel>
#include "SH_ExtendedSqlProxyModel.h"
  
```

Graphe des dépendances par inclusion de SH_GroupsTableModel.h :



Ce graphe montre quels fichiers incluent directement ou indirectement ce fichier :



Classes

- class [SH_GroupsTableModel](#)

5.118 SH_GroupsTableModel.h

```

00001 #include <QtSql/QSqlRelationalTableModel>
00002
00003 #ifndef GROUP_H
00004 #define GROUP_H
00005
00006 #include "SH_ExtendedSqlProxyModel.h"
00007
0014 class SH_GroupsTableModel : public SH_ExtendedProxyModel
00015 {
00016     Q_OBJECT
00017     public:
00018
00019
00026     SH_GroupsTableModel(QObject *parent = 0);
00027
00028
  
```

```

00029     protected:
00030
00031
00037     void fillModel();
00038 }
00039
00040 #endif /* GROUP_H */

```

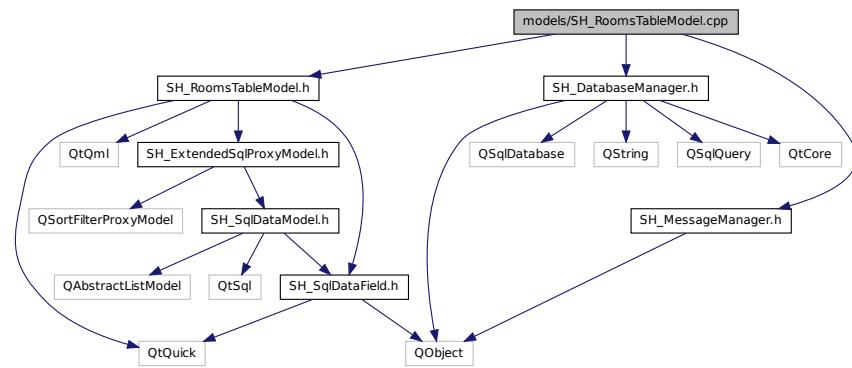
5.119 Référence du fichier models/SH_RoomsTableModel.cpp

```

#include "SH_RoomsTableModel.h"
#include "SH_DatabaseManager.h"
#include "SH_MessageManager.h"

```

Graphe des dépendances par inclusion de SH_RoomsTableModel.cpp :



5.120 SH_RoomsTableModel.cpp

```

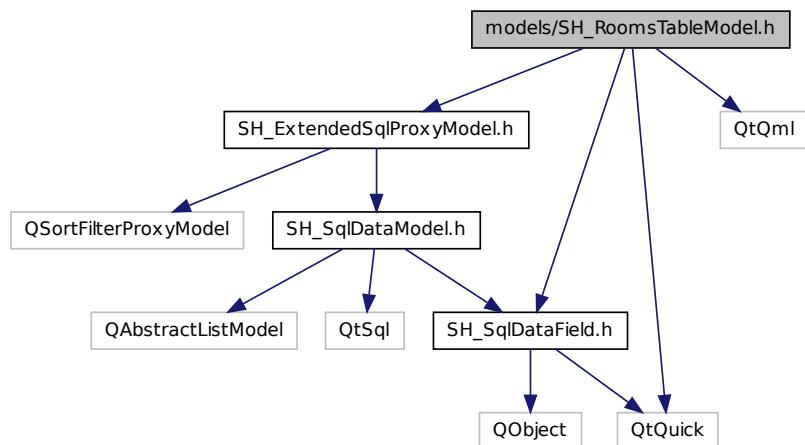
00001 #include "SH_RoomsTableModel.h"
00002 #include "SH_DatabaseManager.h"
00003 #include "SH_MessageManager.h"
00004
00010 SH_RoomsTableModel::SH_RoomsTableModel(
    QObject *parent):
00011     SH_ExtendedProxyModel(parent)
00012 {
00013     SH_ExtendedProxyModel::model->setTable("ROOMSINFOS");
00014     SH_ExtendedProxyModel::model->setOrderBy("FLOOR ASC, NUMBER ASC")
00015 }
00016
00022 void SH_RoomsTableModel::fillModel()
00023 {
00024     SH_ExtendedProxyModel::model->setHeaderData(0, Qt::Horizontal,
00025         QObject::tr("ID"));
00026     SH_ExtendedProxyModel::model->setHeaderData(1, Qt::Horizontal,
00027         QObject::tr("Numéro de chambre"));
00028     SH_ExtendedProxyModel::model->setHeaderData(2, Qt::Horizontal,
00029         QObject::tr("Étage"));
00030     SH_ExtendedProxyModel::model->setHeaderData(3, Qt::Horizontal,
00031         QObject::tr("Type de chambre"));
00032     SH_ExtendedProxyModel::model->setHeaderData(4, Qt::Horizontal,
00033         QObject::tr("Détail"));
00034     SH_ExtendedProxyModel::model->setHeaderData(5, Qt::Horizontal,
00035         QObject::tr("Prix minimum"));
00036     SH_ExtendedProxyModel::model->setHeaderData(6, Qt::Horizontal,
00037         QObject::tr("Prix maximum"));
00038     SH_ExtendedProxyModel::sort(2,Qt::AscendingOrder);
00039     SH_ExtendedProxyModel::addFilterKeyColumn(0);
00040     SH_ExtendedProxyModel::addFilterKeyColumn(7);
00041 }

```

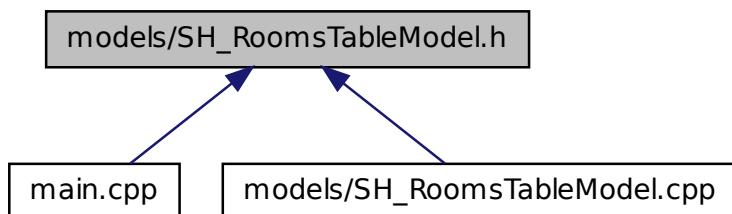
5.121 Référence du fichier models/SH_RoomsTableModel.h

```
#include "SH_ExtendedSqlProxyModel.h"
#include "SH_SqlDataField.h"
#include <QtQml>
#include <QtQuick>
```

Graphe des dépendances par inclusion de SH_RoomsTableModel.h :



Ce graphe montre quels fichiers incluent directement ou indirectement ce fichier :



Classes

- class [SH_RoomsTableModel](#)

5.122 SH_RoomsTableModel.h

```
00001 #ifndef ROOM_H
00002 #define ROOM_H
00003
00004 #include "SH_ExtendedSqlProxyModel.h"
00005 #include "SH_SqlDataField.h"
00006 #include <QtQml>
```

```

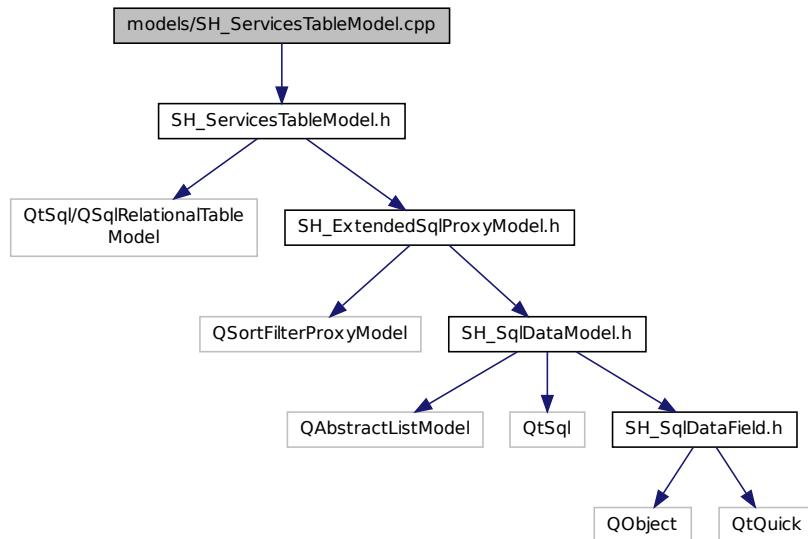
00007 #include <QtQuick>
00008
00009
00016 class SH_RoomsTableModel : public SH_ExtendedProxyModel
00017 {
00018     Q_OBJECT
00019 public:
00020     SH_RoomsTableModel(QObject *parent = 0);
00028
00029 protected:
00035     virtual void fillModel();
00036 private:
00037
00038 };
00039
00040 #endif /* ROOM_H*/

```

5.123 Référence du fichier models/SH_ServicesTableModel.cpp

#include "SH_ServicesTableModel.h"

Graphe des dépendances par inclusion de SH_ServicesTableModel.cpp :



5.124 SH_ServicesTableModel.cpp

```

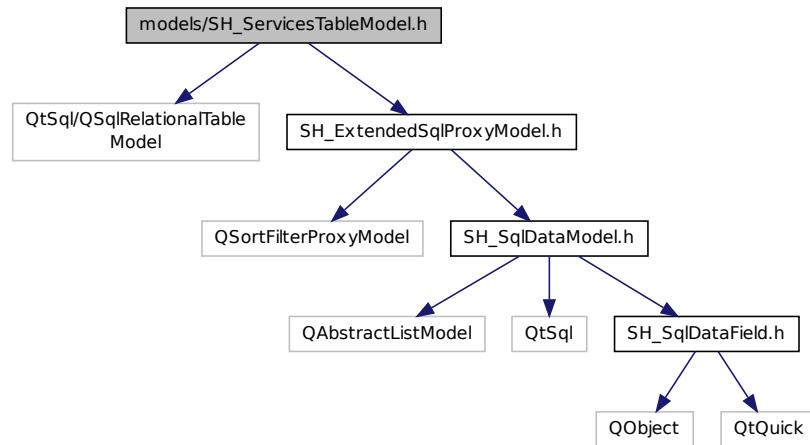
00001 #include "SH_ServicesTableModel.h"
00002
00003
00009 SH_ServicesTableModel::SH_ServicesTableModel(
00010     QObject *parent):
00011     SH_ExtendedProxyModel(parent)
00012     SH_ExtendedProxyModel::model->setTable("SERVICESINFOS");
00013 }
00014
00015
00021 void SH_ServicesTableModel::fillModel()
00022 {
00023     SH_ExtendedProxyModel::sort(1, Qt::AscendingOrder);
00024 }

```

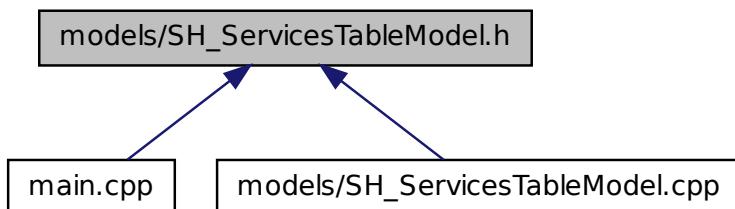
5.125 Référence du fichier models/SH_ServicesTableModel.h

```
#include <QtSql/ QSqlRelationalTableModel>
#include "SH_ExtendedSqlProxyModel.h"
```

Graphe des dépendances par inclusion de SH_ServicesTableModel.h :



Ce graphe montre quels fichiers incluent directement ou indirectement ce fichier :



Classes

- class [SH_ServicesTableModel](#)

5.126 SH_ServicesTableModel.h

```

00001 #include <QtSql/ QSqlRelationalTableModel>
00002 #ifndef SERVICE_H
00003 #define SERVICE_H
00004
00005 #include "SH_ExtendedSqlProxyModel.h"
00006
00013 class SH_ServicesTableModel : public SH_ExtendedProxyModel
00014 {
00015     Q_OBJECT
00016 public:
```

```

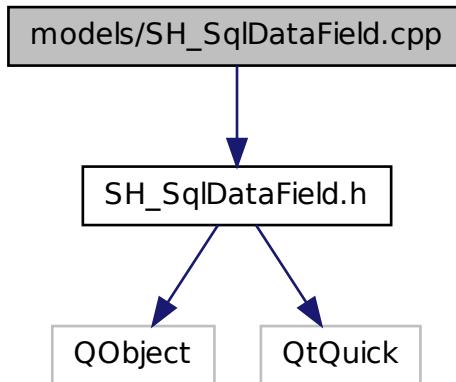
00017
00018     SH_ServicesTableModel (QObject *parent = 0);
00026
00027
00028 protected:
00029
00035     void fillModel ();
00036 };
00037
00038 #endif /* SERVICE_H*/

```

5.127 Référence du fichier models/SH_SqlDataField.cpp

#include "SH_SqlDataField.h"

Graphe des dépendances par inclusion de SH_SqlDataField.cpp :



5.128 SH_SqlDataField.cpp

```

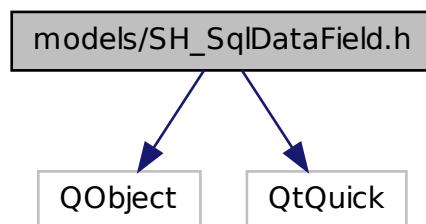
00001 #include "SH_SqlDataField.h"
00002
00008 SH_SqlDataFields::SH_SqlDataFields(QQuickItem *parent) :
00009     QQuickItem(parent)
00010 {
00011     this->setHeight(15);
00012 }
00013
00019 void SH_SqlDataFields::setText(QString newText)
00020 {
00021     m_text = newText;
00022     if (m_name == "")
00023     {
00024         this->setName(m_text.toUpper());
00025     }
00026     emit textChanged();
00027 }
00028
00035 void SH_SqlDataFields::setName(QString newName)
00036 {
00037     m_name = newName;
00038     this->setSortOrder(Qt::AscendingOrder);
00039     if (m_text == "")
00040     {
00041         this->setText(m_name);
00042     }
00043     emit nameChanged();
00044     emit roleChanged();

```

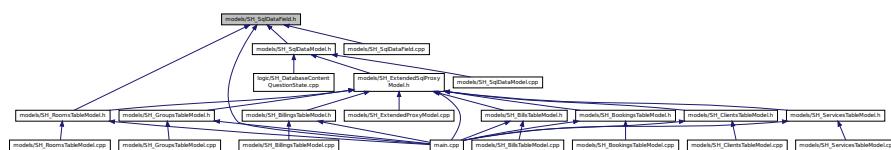
```
00045 }
00046
00047
00054 void SH_SqlDataFields::setSortOrder(Qt::SortOrder newSortOrder)
00055 {
00056     m_sortOrder = newSortOrder;
00057     emit sortOrderChanged();
00058 }
```

5.129 Référence du fichier models/SH_SqlDataField.h

```
#include <QObject>
#include <QtQuick>
Graphe des dépendances par inclusion de SH_SqlDataField.h :
```



Ce graphe montre quels fichiers incluent directement ou indirectement ce fichier :



Classes

- class SH.SqlDataFields

5.130 SH_SqlDataField.h

```
00001 #ifndef SQLDATAFIELDS_H
00002 #define SQLDATAFIELDS_H
00003
00004 #include <QObject>
00005 #include <QtQuick>
00006
00012 class SH_SQLDataFields : public QQQuickItem
00013 {
00014     Q_OBJECT
00015     Q_PROPERTY(QString name READ name WRITE setName NOTIFY
nameChanged)
00016     Q_PROPERTY(QString text READ text WRITE setText NOTIFY
textChanged)
00017     Q_PROPERTY(QByteArray role READ role NOTIFY roleChanged)
00018     Q_PROPERTY(Qt::SortOrder sortOrder READ sortOrder WRITE
```

```

00019     setSortOrder NOTIFY sortOrderChanged)
00020     public:
00021
00022         explicit SH_SqlDataFields(QQuickItem *parent = 0);
00023
00024         QString text() const { return m_text; }
00025         QString name() const { return m_name; }
00026         QByteArray role() const { return QByteArray(m_name.toUpper().toStdString().c_str()); }
00027         Qt::SortOrder sortOrder() const { return m_sortOrder; }
00028
00029         void setText(QString newText);
00030         void setName(QString newName);
00031         void setSortOrder(Qt::SortOrder newSortOrder);
00032
00033     private:
00034         QString m_text;
00035         QString m_name;
00036         Qt::SortOrder m_sortOrder;
00037
00038     signals:
00039         void textChanged();
00040         void nameChanged();
00041         void roleChanged();
00042         void sortOrderChanged();
00043
00044     public slots:
00045
00046 };
00047
00048 #endif /* SQLDATAFIELDS_H */

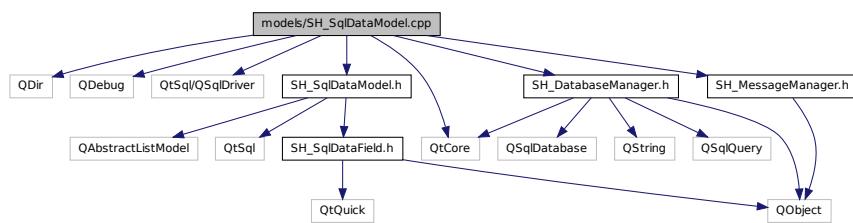
```

5.131 Référence du fichier models/SH_SqlDataManager.cpp

```

#include <QDir>
#include <QDebug>
#include <QtSql/QSqlDriver>
#include <QtCore>
#include "SH_SqlDataManager.h"
#include "SH_DatabaseManager.h"
#include "SH_MessageManager.h"
Graphe des dépendances par inclusion de SH_SqlDataManager.cpp :

```



5.132 SH_SqlDataManager.cpp

```

00001 #include <QDir>
00002 #include <QDebug>
00003 #include <QtSql/QSqlDriver>
00004 #include <QtCore>
00005 #include "SH_SqlDataManager.h"
00006 #include "SH_DatabaseManager.h"
00007 #include "SH_MessageManager.h"
00008
00009
00010 SH_SqlDataManager::SH_SqlDataManager(QObject *parent) :
00011     QAbstractListModel(parent)
00012 {
00013 }
00014
00015

```

```

00027 int SH_SqlDataModel::rowCount (const QModelIndex &parent) const
00028 {
00029     return mRecords.count();
00030 }
00031
00032
00033 QVariant SH_SqlDataModel::data (const QModelIndex &index, int role) const
00034 {
00035     if (this->mRecords.count() > 0)
00036     {
00037         int row = index.row();
00038         int column = this->fieldFromRole(role);
00039         int nbCols = this->mRoles.count();
00040         if(column >= 0 && column < nbCols) {
00041             SH_MessageManager::infoMessage(QString("row : %1, column : %2,
00042 field: %3 (%4), value : %5\n").arg(index.row()).arg(index.column()).arg(column).arg(QString(this->
00043 mDataFields.at(column)->role())).arg(this->mRecords.at(row).value(column).toString()));
00044             return this->mRecords.at(row).value(column);
00045         } else{
00046             SH_MessageManager::errorMessage(QString("rien à retourner pour
00047 %1x%2x%3 (%4>=%5)").arg(index.row()).arg(index.column()).arg(role).arg(column).arg(nbCols));
00048         }
00049     }
00050     SH_MessageManager::errorMessage("modèle vide");
00051     return QVariant();
00052 }
00053
00054
00055 }
00056
00057 QVariantMap SH_SqlDataModel::datas() const
00058 {
00059     qDebug() << "datas";
00060     QVariantMap result;
00061     if (this->mRecords.count() > 0)
00062     {
00063         qDebug() << "datas ok";
00064         for(int column = 0; column < this->mRoles.count(); column++) {
00065             for(int row = 0; row < this->mRecords.count();row++) {
00066                 qDebug() << "data inserted";
00067                 result.insertMulti(this->mRoles.value(column),this->
00068 mRecords.at(row).value(column));
00069             }
00070         }
00071     }
00072     return result;
00073 }
00074
00075
00076
00077 }
00078
00079 bool SH_SqlDataModel::setHeaderData(int section, Qt::Orientation orientation,
00080                                     const QVariant &value, int role)
00081 {
00082     Q_UNUSED(role);
00083     if (orientation == Qt::Horizontal)
00084     {
00085         this->mDataFields.at(section)->setText(value.toString());
00086         return (this->mDataFields.at(section)->text() == value.toString());
00087     }
00088     return false;
00089 }
00090
00091 const QString &SH_SqlDataModel::query() const
00092 {
00093     return mSqlQuery.lastQuery();
00094 }
00095
00096
00097 const QString &SH_SqlDataModel::tableName() const
00098 {
00099     return mTable;
00100 }
00101
00102
00103 const QString &SH_SqlDataModel::filter() const
00104 {
00105     return mFilter;
00106 }
00107
00108 const QStringList SH_SqlDataModel::fieldsList() const
00109 {
00110     QStringList fields;
00111     if(!this->mDataFields.isEmpty()) {
00112         int c = mDataFields.count();
00113         for (int i = 0; i < c; i++)
00114         {
00115             fields.append(this->mDataFields.at(i)->name());
00116         }
00117     }
00118     return fields;
00119 }
00120
00121
00122 void SH_SqlDataModel::setTable(const QString &tableName)
00123 {
00124     if (mTable.toUpper() != tableName.toUpper() && tableName != "")
00125
00126
00127
00128
00129
00130
00131
00132
00133
00134
00135
00136
00137
00138
00139
00140
00141
00142
00143
00144
00145
00146
00147
00148
00149
00150
00151
00152
00153
00154
00155
00156
00157
00158
00159
00160
00161
00162
00163
00164
00165
00166
00167
00168
00169
00170
00171
00172
00173
00174
00175
00176
00177
00178
00179
00180
00181
00182
00183
00184
00185
00186
00187
00188
00189
00190
00191
00192
00193
00194
00195
00196
00197
00198
00199
00200
00201
00202
00203
00204
00205
00206
00207
00208
00209
00210
00211
00212
00213
00214
00215
00216
00217
00218
00219
00220
00221
00222
00223
00224
00225
00226
00227
00228
00229
00230
00231
00232
00233
00234
00235
00236
00237
00238
00239
00240
00241
00242
00243
00244
00245
00246
00247
00248
00249
00250
00251
00252
00253
00254
00255
00256
00257
00258
00259
00260
00261
00262
00263
00264
00265
00266
00267
00268
00269
00270
00271
00272
00273
00274
00275
00276
00277
00278
00279
00280
00281
00282
00283
00284
00285
00286
00287
00288
00289
00290
00291
00292
00293
00294
00295
00296
00297
00298
00299
00300
00301
00302
00303
00304
00305
00306
00307
00308
00309
00310
00311
00312
00313
00314
00315
00316
00317
00318
00319
00320
00321
00322
00323
00324
00325
00326
00327
00328
00329
00330
00331
00332
00333
00334
00335
00336
00337
00338
00339
00340
00341
00342
00343
00344
00345
00346
00347
00348
00349
00350
00351
00352
00353
00354
00355
00356
00357
00358
00359
00360
00361
00362
00363
00364
00365
00366
00367
00368
00369
00370
00371
00372
00373
00374
00375
00376
00377
00378
00379
00380
00381
00382
00383
00384
00385
00386
00387
00388
00389
00390
00391
00392
00393
00394
00395
00396
00397
00398
00399
00400
00401
00402
00403
00404
00405
00406
00407
00408
00409
00410
00411
00412
00413
00414
00415
00416
00417
00418
00419
00420
00421
00422
00423
00424
00425
00426
00427
00428
00429
00430
00431
00432
00433
00434
00435
00436
00437
00438
00439
00440
00441
00442
00443
00444
00445
00446
00447
00448
00449
00450
00451
00452
00453
00454
00455
00456
00457
00458
00459
00460
00461
00462
00463
00464
00465
00466
00467
00468
00469
00470
00471
00472
00473
00474
00475
00476
00477
00478
00479
00480
00481
00482
00483
00484
00485
00486
00487
00488
00489
00490
00491
00492
00493
00494
00495
00496
00497
00498
00499
00500
00501
00502
00503
00504
00505
00506
00507
00508
00509
00510
00511
00512
00513
00514
00515
00516
00517
00518
00519
00520
00521
00522
00523
00524
00525
00526
00527
00528
00529
00530
00531
00532
00533
00534
00535
00536
00537
00538
00539
00540
00541
00542
00543
00544
00545
00546
00547
00548
00549
00550
00551
00552
00553
00554
00555
00556
00557
00558
00559
00560
00561
00562
00563
00564
00565
00566
00567
00568
00569
00570
00571
00572
00573
00574
00575
00576
00577
00578
00579
00580
00581
00582
00583
00584
00585
00586
00587
00588
00589
00590
00591
00592
00593
00594
00595
00596
00597
00598
00599
00600
00601
00602
00603
00604
00605
00606
00607
00608
00609
00610
00611
00612
00613
00614
00615
00616
00617
00618
00619
00620
00621
00622
00623
00624
00625
00626
00627
00628
00629
00630
00631
00632
00633
00634
00635
00636
00637
00638
00639
00640
00641
00642
00643
00644
00645
00646
00647
00648
00649
00650
00651
00652
00653
00654
00655
00656
00657
00658
00659
00660
00661
00662
00663
00664
00665
00666
00667
00668
00669
00670
00671
00672
00673
00674
00675
00676
00677
00678
00679
00680
00681
00682
00683
00684
00685
00686
00687
00688
00689
00690
00691
00692
00693
00694
00695
00696
00697
00698
00699
00700
00701
00702
00703
00704
00705
00706
00707
00708
00709
00710
00711
00712
00713
00714
00715
00716
00717
00718
00719
00720
00721
00722
00723
00724
00725
00726
00727
00728
00729
00730
00731
00732
00733
00734
00735
00736
00737
00738
00739
00740
00741
00742
00743
00744
00745
00746
00747
00748
00749
00750
00751
00752
00753
00754
00755
00756
00757
00758
00759
00760
00761
00762
00763
00764
00765
00766
00767
00768
00769
00770
00771
00772
00773
00774
00775
00776
00777
00778
00779
00780
00781
00782
00783
00784
00785
00786
00787
00788
00789
00790
00791
00792
00793
00794
00795
00796
00797
00798
00799
00800
00801
00802
00803
00804
00805
00806
00807
00808
00809
00810
00811
00812
00813
00814
00815
00816
00817
00818
00819
00820
00821
00822
00823
00824
00825
00826
00827
00828
00829
00830
00831
00832
00833
00834
00835
00836
00837
00838
00839
00840
00841
00842
00843
00844
00845
00846
00847
00848
00849
00850
00851
00852
00853
00854
00855
00856
00857
00858
00859
00860
00861
00862
00863
00864
00865
00866
00867
00868
00869
00870
00871
00872
00873
00874
00875
00876
00877
00878
00879
00880
00881
00882
00883
00884
00885
00886
00887
00888
00889
00890
00891
00892
00893
00894
00895
00896
00897
00898
00899
00900
00901
00902
00903
00904
00905
00906
00907
00908
00909
00910
00911
00912
00913
00914
00915
00916
00917
00918
00919
00920
00921
00922
00923
00924
00925
00926
00927
00928
00929
00930
00931
00932
00933
00934
00935
00936
00937
00938
00939
00940
00941
00942
00943
00944
00945
00946
00947
00948
00949
00950
00951
00952
00953
00954
00955
00956
00957
00958
00959
00960
00961
00962
00963
00964
00965
00966
00967
00968
00969
00970
00971
00972
00973
00974
00975
00976
00977
00978
00979
00980
00981
00982
00983
00984
00985
00986
00987
00988
00989
00990
00991
00992
00993
00994
00995
00996
00997
00998
00999
01000
01001
01002
01003
01004
01005
01006
01007
01008
01009
01010
01011
01012
01013
01014
01015
01016
01017
01018
01019
01020
01021
01022
01023
01024
01025
01026
01027
01028
01029
01030
01031
01032
01033
01034
01035
01036
01037
01038
01039
01040
01041
01042
01043
01044
01045
01046
01047
01048
01049
01050
01051
01052
01053
01054
01055
01056
01057
01058
01059
01060
01061
01062
01063
01064
01065
01066
01067
01068
01069
01070
01071
01072
01073
01074
01075
01076
01077
01078
01079
01080
01081
01082
01083
01084
01085
01086
01087
01088
01089
01090
01091
01092
01093
01094
01095
01096
01097
01098
01099
01100
01101
01102
01103
01104
01105
01106
01107
01108
01109
01110
01111
01112
01113
01114
01115
01116
01117
01118
01119
01120
01121
01122
01123
01124
01125
01126
01127
01128
01129
01130
01131
01132
01133
01134
01135
01136
01137
01138
01139
01140
01141
01142
01143
01144
01145
01146
01147
01148
01149
01150
01151
01152
01153
01154
01155
01156
01157
01158
01159
01160
01161
01162
01163
01164
01165
01166
01167
01168
01169
01170
01171
01172
01173
01174
01175
01176
01177
01178
01179
01180
01181
01182
01183
01184
01185
01186
01187
01188
01189
01190
01191
01192
01193
01194
01195
01196
01197
01198
01199
01200
01201
01202
01203
01204
01205
01206
01207
01208
01209
01210
01211
01212
01213
01214
01215
01216
01217
01218
01219
01220
01221
01222
01223
01224
01225
01226
01227
01228
01229
01230
01231
01232
01233
01234
01235
01236
01237
01238
01239
01240
01241
01242
01243
01244
01245
01246
01247
01248
01249
01250
01251
01252
01253
01254
01255
01256
01257
01258
01259
01260
01261
01262
01263
01264
01265
01266
01267
01268
01269
01270
01271
01272
01273
01274
01275
01276
01277
01278
01279
01280
01281
01282
01283
01284
01285
01286
01287
01288
01289
01290
01291
01292
01293
01294
01295
01296
01297
01298
01299
01300
01301
01302
01303
01304
01305
01306
01307
01308
01309
01310
01311
01312
01313
01314
01315
01316
01317
01318
01319
01320
01321
01322
01323
01324
01325
01326
01327
01328
01329
01330
01331
01332
01333
01334
01335
01336
01337
01338
01339
01340
01341
01342
01343
01344
01345
01346
01347
01348
01349
01350
01351
01352
01353
01354
01355
01356
01357
01358
01359
01360
01361
01362
01363
01364
01365
01366
01367
01368
01369
01370
01371
01372
01373
01374
01375
01376
01377
01378
01379
01380
01381
01382
01383
01384
01385
01386
01387
01388
01389
01390
01391
01392
01393
01394
01395
01396
01397
01398
01399
01400
01401
01402
01403
01404
01405
01406
01407
01408
01409
01410
01411
01412
01413
01414
01415
01416
01417
01418
01419
01420
01421
01422
01423
01424
01425
01426
01427
01428
01429
01430
01431
01432
01433
01434
01435
01436
01437
01438
01439
01440
01441
01442
01443
01444
01445
01446
01447
01448
01449
01450
01451
01452
01453
01454
01455
01456
01457
01458
01459
01460
01461
01462
01463
01464
01465
01466
01467
01468
01469
01470
01471
01472
01473
01474
01475
01476
01477
01478
01479
01480
01481
01482
01483
01484
01485
01486
01487
01488
01489
01490
01491
01492
01493
01494
01495
01496
01497
01498
01499
01500
01501
01502
01503
01504
01505
01506
01507
01508
01509
01510
01511
01512
01513
01514
01515
01516
01517
01518
01519
01520
01521
01522
01523
01524
01525
01526
01527
01528
01529
01530
01531
01532
01533
01534
01535
01536
01537
01538
01539
01540
01541
01542
01543
01544
01545
01546
01547
01548
01549
01550
01551
01552
01553
01554
01555
01556
01557
01558
01559
01560
01561
01562
01563
01564
01565
01566
01567
01568
01569
01570
01571
01572
01573
01574
01575
01576
01577
01578
01579
01580
01581
01582
01583
01584
01585
01586
01587
01588
01589
01590
01591
01592
01593
01594
01595
01596
01597
01598
01599
01600
01601
01602
01603
01604
01605
01606
01607
01608
01609
01610
01611
01612
01613
01614
01615
01616
01617
01618
01619
01620
01621
01622
01623
01624
01625
01626
01627
01628
01629
01630
01631
01632
01633
01634
01635
01636
01637
01638
01639
01640
01641
01642
01643
01644
01645
01646
01647
01648
01649
01650
01651
01652
01653
01654
01655
01656
01657
01658
01659
01660
01661
01662
01663
01664
01665
01666
01667
01668
01669
01670
01671
01672
01673
01674
01675
01676
01677
01678
01679
01680
01681
01682
01683
01684
01685
01686
01687
01688
01689
01690
01691
01692
01693
01694
01695
01696
01697
01698
01699
01700
01701
01702
01703
01704
01705
01706
01707
01708
01709
01710
01711
01712
01713
01714
01715
01716
01717
01718
01719
01720
01721
01722
01723
01724
01725
01726
01727
01728
01729
01730
01731
01732
01733
01734
01735
01736
01737
01738
01739
01740
01741
01742
01743
01744
01745
01746
01747
01748
01749
01750
01751
01752
01753
01754
01755
01756
01757
01758
01759
01760
01761
01762
01763
01764
01765
01766
01767
01768
0
```

```

00156      {
00157          mTable = tableName.toUpper();
00158          emit tableChanged();
00159      }
00160  }
00161
00168 void SH_SqlDataModel::setFilterCondition(const QString &filter)
00169 {
00170     if (mFilter != filter && filter != "")
00171     {
00172         mFilter = filter;
00173         emit filterChanged();
00174     }
00175 }
00176
00182 void SH_SqlDataModel::resetFilterCondition()
00183 {
00184     mFilter = "";
00185     emit filterChanged();
00186 }
00187
00194 bool SH_SqlDataModel::fetch(QString tableName, QString filter, QString sort,
00195     QStringList fieldsList)
00196 {
00197     if (!mTable.isEmpty() || !tableName.isEmpty())
00198     {
00199         SH_MessageManager::infoMessage("Bienvenue dans fetch");
00200         qDebug() << mTable << " " << this->fieldsList().join(", ") << " " <<
00201         mFilter << " " << mSort;
00202         this->setFields(fieldsList);
00203         this->setTable(tableName);
00204         this->setFilterCondition(filter);
00205         this->setOrderBy(sort);
00206         qDebug() << tableName << " " << filter << " " << sort << " " << fieldsList.join(", ");
00207         try
00208         {
00209             beginResetModel();
00210             mRecords.clear();
00211             endResetModel();
00212             qDebug() << mTable << " " << this->fieldsList() << " " <<
00213             mFilter << " " << mSort;
00214             mSqlQuery = SH_DatabaseManager::getInstance()->
00215             execSelectQuery(mTable, this->fieldsList(),
00216             mFilter, mSort);
00217             qDebug() << mSqlQuery.executedQuery();
00218             bool next = mSqlQuery.next();
00219             if(next)
00220                 qDebug() << "next ok";
00221             while (next) /* && mSqlQuery.isActive() */
00222             {
00223                 QSqlRecord record = mSqlQuery.record();
00224                 qDebug() << "\n";
00225                 SH_MessageManager::infoMessage("Nouvelle ligne récupérée");
00226                 SH_MessageManager::infoMessage(QString("%1 champs").arg(
00227                     record.count()));
00228                 if (mSqlQuery.isValid() && (!record.isEmpty()) && (record.count() > 0))
00229                 {
00230                     beginInsertRows(QModelIndex(), 0, 0);
00231                     mRecords.append(record);
00232                     int nbFields = record.count();
00233                     for (int i = 0; i < nbFields; i++)
00234                     {
00235                         SH_MessageManager::infoMessage(QString("%1 : %2").arg(
00236                             record.fieldName(i)).arg(record.value(i).toString()));
00237                         field->setName(record.fieldName(i));
00238                         SH_MessageManager::infoMessage("nouveau
00239                         champ (le n°%1): %2").arg(i).arg(field->name());
00240                         mDataFields.append(field);
00241                     }
00242                     this->applyRoles();
00243                     emit fieldsChanged();
00244                     endInsertRows();
00245                 }
00246                 next = mSqlQuery.next();
00247             }
00248         }
00249     catch (const std::exception &e)
00250     {

```

```

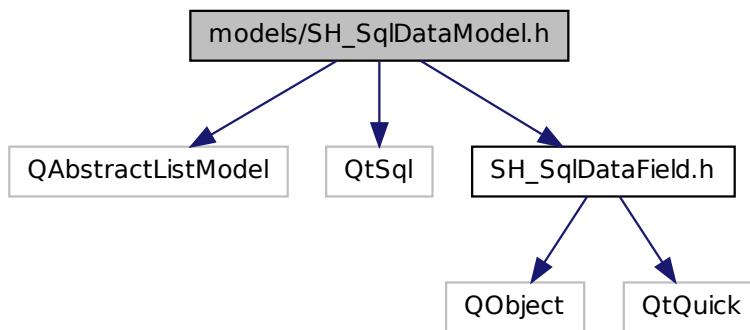
00251         SH_MessageManager::errorMessage(e.what(), "exception");
00252         if (this->lastError().isEmpty())
00253         {
00254             SH_MessageManager::errorMessage(this->
00255                 lastError(), "erreur SQL");
00256         }
00257         if (this->lastError().isEmpty())
00258         {
00259             SH_MessageManager::errorMessage(this->
00260                 lastError(), "erreur SQL");
00261         }
00262     return (!this->isEmpty());
00263 }
00264
00271 SH_SqlDataFields *SH_SqlDataModel::field(int i) const
00272 {
00273     i = qMin(i, this->fieldsCount()-1);
00274     i = qMax(i, 0);
00275     return this->mDataFields.at(i);
00276 }
00277
00284 void SH_SqlDataModel::setFields(QStringList fields)
00285 {
00286     fields.removeDuplicates();
00287     int nbFields = fields.count();
00288     if (nbFields > 0)
00289     {
00290         for (int i = 0; i < nbFields; i++)
00291         {
00292             SH_SqlDataFields *field = new SH_SqlDataFields();
00293             field->setName(fields.at(i));
00294             mDataFields.append(field);
00295         }
00296         this->applyRoles();
00297         emit fieldsChanged();
00298     }
00299 }
00300
00306 void SH_SqlDataModel::resetFieldsToAll()
00307 {
00308     mDataFields.clear();
00309     this->applyRoles();
00310     emit fieldsChanged();
00311 }
00312
00319 const QString &SH_SqlDataModel::lastError()
00320 {
00321     QSqlError error = mSqlQuery.lastError();
00322     if (error.isValid() && !error.text().isEmpty())
00323     {
00324         emit lastErrorChanged();
00325         return error.text();
00326     }
00327     else
00328     {
00329         return "";
00330     }
00331 }
00332
00333
00339 void SH_SqlDataModel::applyRoles()
00340 {
00341     this->mRoles.clear();
00342     int nbFields = this->mDataFields.count();
00343     for (int i = 0; i < nbFields; i++)
00344     {
00345         /*MessageManager::infoMessage(QString("nouveau rôle :
00346             %1").arg(QString(this->mDataFields.at(i)->role())));*/
00347         this->mRoles.insert(this->roleForField(i), this->
00348             mDataFields.at(i)->role());
00349     }
00350
00351
00358 int SH_SqlDataModel::fieldsCount() const
00359 {
00360     return mDataFields.count();
00361 }
00362
00369 void SH_SqlDataModel::setOrderBy(QString sort)
00370 {
00371     this->mSort = sort;
00372 }
00373

```

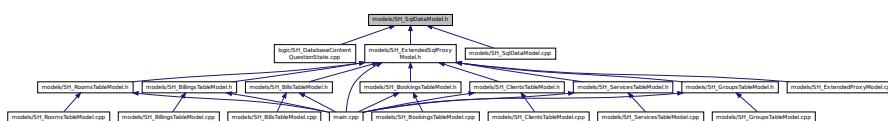
```
00380 bool SH_SqlDataModel::isEmpty() const
00381 {
00382     return mRecords.empty();
00383 }
```

5.133 Référence du fichier models/SH_SqlDataModel.h

```
#include <QAbstractListModel>
#include <QtSql>
#include "SH_SqlDataField.h"
Graphe des dépendances par inclusion de SH_SqlDataModel.h :
```



Ce graphe montre quels fichiers incluent directement ou indirectement ce fichier :



Classes

- class `SH_SqlDataModel`

5.134 SH_SqlDataModel.h

```
00001 #ifndef SQLDATAMODEL_H
00002 #define SQLDATAMODEL_H
00003
00004 #include <QAbstractListModel>
00005 #include <QtSql>
00006 #include "SH_SqlDataField.h"
00007
00008
00014 class SH_SqlDataModel : public QAbstractListModel
00015 {
00016     Q_OBJECT
00017     Q_PROPERTY(QString table READ tableName WRITE setTable NOTIFY
00018     tableChanged)
00019     Q_PROPERTY(QString filter READ filter WRITE setFilterCondition NOTIFY
00019     filterChanged)
00019     Q_PROPERTY(QString lastError READ lastError NOTIFY
```

```

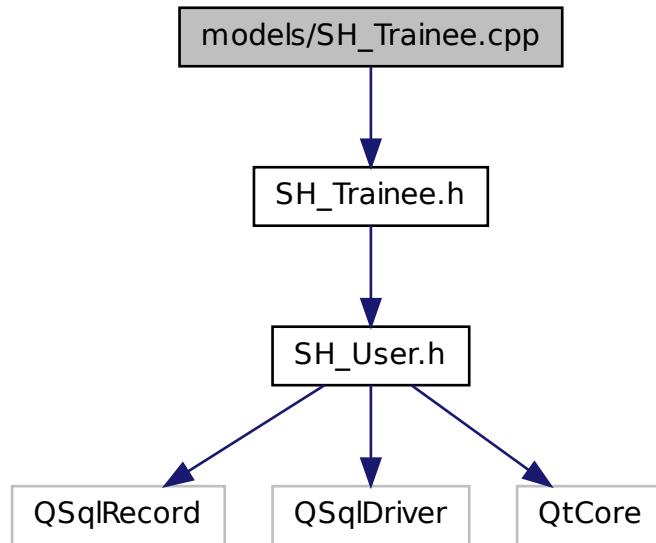
    lastErrorChanged)
00020
00021 public:
00022     explicit SH_SqlDataModel(QObject *parent = 0);
00023     int rowCount(const QModelIndex &parent) const;
00024     QVariant data(const QModelIndex &index, int role) const;
00025     QVariantMap datas() const;
00026     bool setHeaderData(int section, Qt::Orientation orientation, const QVariant &value, int
00027         role = Qt::EditRole);
00028     int roleForField(int fieldIndex) const { return UserRole + fieldIndex; }
00029     int fieldFromRole(int role) const { return role - UserRole; }
00030     const QString &tableName() const;
00031     const QString &lastError();
00032     const QString &filter() const;
00033
00034     void setTable(const QString &tableName);
00035     void setFilterCondition(const QString &filter);
00036     void resetFilterCondition();
00037     void setFields(QStringList fieldList);
00038     void resetFieldsToAll();
00039
00040     bool fetch(QString tableName = "", QString filter = "", QString sort = "",
00041     QStringList fields = QStringList());
00042     SH_SqlDataFields *field(int i) const;
00043     int fieldsCount() const;
00044     void setOrderBy(QString sort);
00045     virtual QHash<int, QByteArray> roleNames() const { return this->
00046         mRoles; }
00047
00048     bool isEmpty() const;
00049
00050     const QString &query() const;
00051     const QStringList fieldsList() const;
00052
00053 signals:
00054     void fieldsChanged();
00055     void tableChanged();
00056     void lastErrorChanged();
00057     void filterChanged();
00058     void rolesChanged();
00059
00060 protected:
00061     void applyRoles();
00062
00063 private:
00064     QString mTable;
00065     QString mFilter;
00066     QString mSort;
00067     QList<SH_SqlDataFields *> mDataFields;
00068     QHash<int, QByteArray> mRoles;
00069     QSqlQuery mSqlQuery;
00070     QList<QSqlRecord> mRecords;
00071 };
00072
00073 #endif /* SQLDATAMODEL_H */
00074

```

5.135 Référence du fichier models/SH_Trainee.cpp

```
#include "SH_Trainee.h"
```

Graphe des dépendances par inclusion de SH_Trainee.cpp :



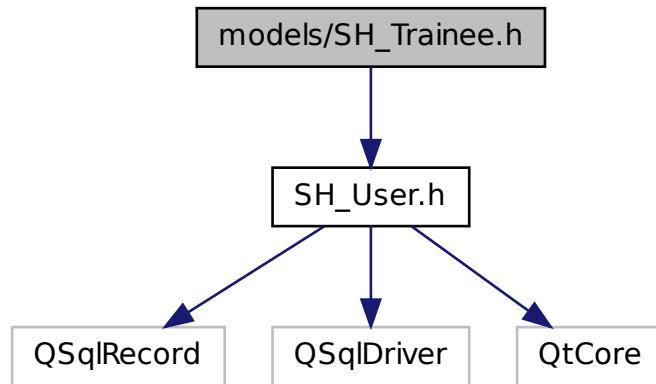
5.136 SH_Trainee.cpp

```
00001 #include "SH_Trainee.h"
00002
00009 SH_Trainee::SH_Trainee(QString name, int id, QObject *parent) :
00010     SH_User(name, id, true, false, false, parent)
00011 {
00012 }
00013
00014
```

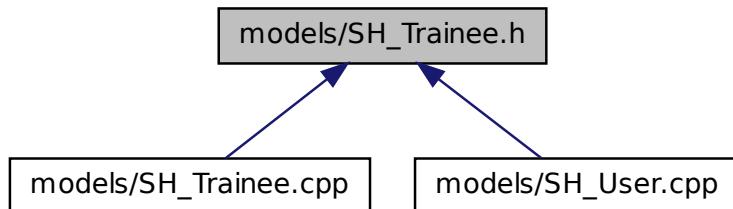
5.137 Référence du fichier models/SH_Trainee.h

```
#include "SH_User.h"
```

Graphe des dépendances par inclusion de SH_Trainee.h :



Ce graphe montre quels fichiers incluent directement ou indirectement ce fichier :



Classes

- class `SH_Trainee`

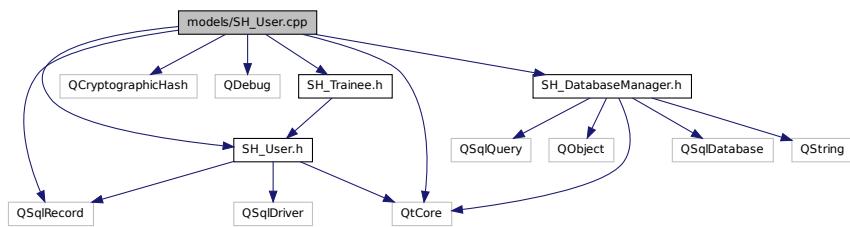
5.138 SH_Trainee.h

```

00001 #include "SH_User.h"
00002
00003 #ifndef TRAINEE_H
00004 #define TRAINEE_H
00005
00006
00007
00014 class SH_Trainee : public SH_User
00015 {
00016     Q_OBJECT
00017 public:
00026     SH_Trainee(QString name, int id, QObject *parent = 0);
00027 };
00028
00029 #endif /* TRAINEE_H*/
  
```

5.139 Référence du fichier models/SH_User.cpp

```
#include <QSqlRecord>
#include <QtCore>
#include <QCryptographicHash>
#include <QDebug>
#include "SH_User.h"
#include "SH_Trainee.h"
#include "SH_DatabaseManager.h"
Graphe des dépendances par inclusion de SH_User.cpp :
```



5.140 SH_User.cpp

```
00001 #include <QSqlRecord>
00002 #include <QtCore>
00003 #include <QCryptographicHash>
00004 #include <QDebug>
00005 #include "SH_User.h"
00006 #include "SH_Trainee.h"
00007 #include "SH_DatabaseManager.h"
00008
00015 SH_User::SH_User(QString name, int id, bool isReceptionist, bool isManagerX, bool
00016     isManagerZ, bool isAdministrator, QObject *parent)
00017     : QObject(parent)
00018 {
00019     this->setName(name);
00020     this->setID(id);
00021     this->m_receptionist = isReceptionist;
00022     this->m_managerX = isManagerX;
00023     this->m_managerZ = isManagerZ;
00024     this->m_administrator = isAdministrator;
00025 }
00026
00032 bool SH_User::isValid() const {
00033     return (!this->m_name.isEmpty()) && (this->m_id != 0));
00034 }
00035
00042 void SH_User::setName(QString name)
00043 {
00044     m_name = name;
00045 }
00046
00047
00053 QString SH_User::name() const
00054 {
00055     return m_name;
00056 }
00057
00064 bool SH_User::isReceptionist() const
00065 {
00066     return this->m_receptionist;
00067 }
00068
00075 int SH_User::roles() const
00076 {
00077     int nb = 0;
00078     if(this->isReceptionist()) {
00079         nb++;
00080     }
00081     if(this->isManagerX()) {
00082         nb++;
```

```

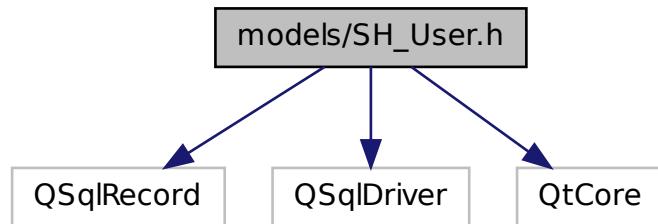
00083     }
00084     if(this->isManagerZ()) {
00085         nb++;
00086     }
00087     if(this->isAdministrator()) {
00088         nb++;
00089     }
00090     return nb;
00091 }
00092
00093 void SH_User::setID(int id)
00094 {
00095     m_id = id;
00096 }
00097
00098 bool SH_User::userExists(QString login) {
00099     qDebug() << "user exists";
00100     return (SH_DatabaseManager::getInstance()->dataCount("USERS", "LOGIN='"+login+"'") == 1);
00101 }
00102
00103 bool SH_User::traineeExists(QString login) {
00104     qDebug() << "trainee exists";
00105     return (SH_DatabaseManager::getInstance()->dataCount("TRAINEES", "LOGIN='"+login+"'") == 1);
00106 }
00107
00108 SH_User *SH_User::logIn(QString login, QString pass)
00109 {
00110     qDebug() << "log in";
00111     bool isValid = false;
00112     QCryptographicHash encPass(QCryptographicHash::Sha512);
00113     encPass.addData(pass.toUtf8());
00114     bool trainee=false;
00115     QStringList fields;
00116     QString table;
00117     if(userExists(login)) {
00118         fields << "ID" << "LOGIN" << "ISRECEPTIONIST" << "ISMANAGERX" << "ISMANAGERZ" << "ISADMINISTRATOR";
00119         table ="USERS";
00120     } else if(traineeExists(login)) {
00121         fields << "ID" << "LOGIN";
00122         table ="TRAINEES";
00123         trainee=true;
00124     }
00125     QSqlQuery result = SH_DatabaseManager::getInstance()->
00126     execSelectQuery(table,fields,"LOGIN='"+login+"' AND ENCRYPTEDPASS='"+QString::fromLatin1(
00127     encPass.result().toHex().toUpper()+"')");
00128     if(result.next()) {
00129         QSqlRecord rec = result.record();
00130         if(rec.isEmpty() || !result.isValid()) {
00131             isValid = false;
00132         } else {
00133             isValid = (rec.value(rec.indexOf("LOGIN")).toString() == login);
00134         }
00135
00136         if(isValid) {
00137             if(trainee) {
00138                 return new SH_Trainee(rec.value(rec.indexOf("LOGIN")).toString(),rec.value(rec.
00139             indexOf("ID")).toInt());
00140             } else {
00141                 return new SH_User(rec.value(rec.indexOf("LOGIN")).toString(),rec.value(rec.indexOf(
00142                 "ID")).toInt(),(rec.value(rec.indexOf("ISRECEPTIONIST")).toString()=="1"),(rec.value(rec.indexOf("ISMANAGERX")
00143                 ).toString()=="1"),(rec.value(rec.indexOf("ISMANAGERZ")).toString()=="1"),(rec.value(rec.indexOf("ISADMINISTRATOR"))
00144                 ).toString()=="1"));
00145             }
00146         }
00147     }
00148     return new SH_User();
00149 }
00150
00151
00152
00153
00154
00155
00156
00157
00158
00159
00160
00161
00162
00163
00164
00165
00166
00167 }
00168
00169
00170

```

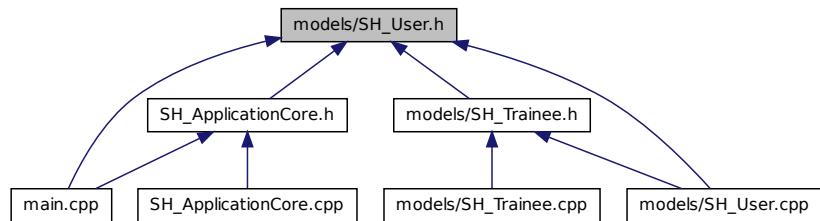
5.141 Référence du fichier models/SH_User.h

```
#include <QSqlRecord>
#include <QSqlDriver>
#include <QtCore>
```

Graphe des dépendances par inclusion de SH_User.h :



Ce graphe montre quels fichiers incluent directement ou indirectement ce fichier :



Classes

- class [SH_User](#)

5.142 SH_User.h

```

00001 #ifndef USER_H
00002 #define USER_H
00003 #include <QSqlRecord>
00004 #include <QSqlDriver>
00005 #include <QtCore>
00006
00007
00008
00015 class SH_User : public QObject
00016 {
00017     Q_OBJECT
00018     Q_PROPERTY(int id READ id)
00019     Q_PROPERTY(QString name READ name NOTIFY nameChanged)
00020     Q_PROPERTY(bool receptionist READ isReceptionist NOTIFY
00021     rolesChanged)
00021     Q_PROPERTY(bool managerX READ isManagerX NOTIFY
00022     rolesChanged)
00022     Q_PROPERTY(bool managerZ READ isManagerZ NOTIFY
00023     rolesChanged)
00023     Q_PROPERTY(bool administrator READ isAdministrator NOTIFY
00024     rolesChanged)
00024     Q_PROPERTY(int roles READ roles NOTIFY rolesChanged)
00025     Q_PROPERTY(bool valid READ isValid NOTIFY validityChanged)
00026
00027 public:
00040     SH_User(QString name = "", int id = 0, bool isReceptionist = false, bool
  
```

```

isManagerX = false, bool isManagerZ = false, bool
isAdministrator = false, QObject *parent = 0);
00047     QString name() const;
00054     int id() const { return this->m_id; }
00061     bool isReceptionist() const;
00068     bool isManagerX() const { return this->m_managerX; }
00075     bool isManagerZ() const { return this->m_managerZ; }
00082     bool isAdministrator() const { return this->m_administrator; }
00089     int roles() const;
00096     bool isValid() const;
00097
00106     static SH_User *logIn(QString login, QString pass);
00114     static bool traineeExists(QString login);
00122     static bool userExists(QString login);
00123
00124 public slots:
00132     static QVariant exists(QVariant login) {return QVariant(
00133         SH_User::userExists(login.toString()) ||
00134         SH_User::traineeExists(login.toString()));}
00133 signals:
00139     void nameChanged();
00145     void rolesChanged();
00151     void validityChanged();
00152
00153 private:
00160     void setName(QString name);
00167     void setID(int id);
00168
00172     QString m_name;
00176     bool m_receptionist;
00180     bool m_managerX;
00184     bool m_managerZ;
00188     bool m_administrator;
00192     int m_id;
00193 };
00194
00195 #endif /* USER_H */

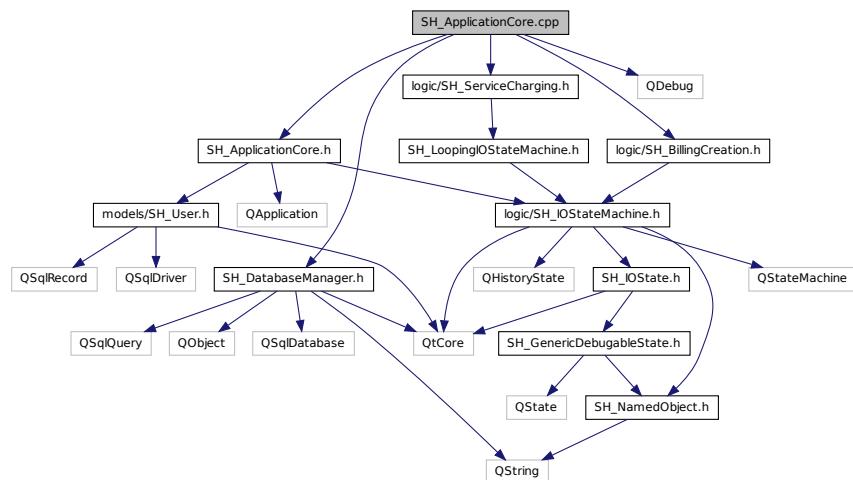
```

5.143 Référence du fichier SH_ApplicationCore.cpp

```

#include "SH_ApplicationCore.h"
#include <QDebug>
#include "SH_DatabaseManager.h"
#include "logic/SH_ServiceCharging.h"
#include "logic/SH_BillingCreation.h"
Graphe des dépendances par inclusion de SH_ApplicationCore.cpp :

```



5.144 SH_ApplicationCore.cpp

```

00001 #include "SH_ApplicationCore.h"
00002 #include <QDebug>
00003 #include "SH_DatabaseManager.h"
00004 #include "logic/SH_ServiceCharging.h"
00005 #include "logic/SH_BillingCreation.h"
00006
00007
00014 SH_ApplicationCore::SH_ApplicationCore(
    QObject* parent) :
00015     QObject(parent)
00016 {
00017     init();
00018 }
00019
00025 SH_ApplicationCore::AppMode SH_ApplicationCore::mode()
    const
00026 {
00027     return m_mode;
00028 }
00029
00035 void SH_ApplicationCore::init() {
00036     this->m_currentUser = new SH_User();
00037 }
00038
00044 void SH_ApplicationCore::setMode(
    SH_ApplicationCore::AppMode mode)
00045 {
00046     if(!this->m_currentUser || ! SH_User::exists(QVariant(this->
m_currentUser->name()).toBool())) {
00047         this->m_mode = CONNEXION;
00048     } else {
00049         if((mode == ADMINISTRATION) && (!this->m_currentUser->
isAdministrator()) ||
00050             ((mode == MANAGEMENT_X) && (!this->m_currentUser->
isManagerX()) ||
00051             ((mode == MANAGEMENT_Z) && (!this->m_currentUser->
isManagerZ()) ||
00052             ((mode == RECEPTION) && (!this->m_currentUser->
isReceptionist())))) {
00053             this->m_mode = ACCUEIL;
00054         } else {
00055             this->m_mode = mode;
00056         }
00057     }
00058 }
00059
00065 SH_User *SH_ApplicationCore::user() const
00066 {
00067     return this->m_currentUser;
00068 }
00069
00075 bool SH_ApplicationCore::userLogOut()
00076 {
00077     this->m_currentUser = new SH_User();
00078     return !this->m_currentUser->isValid();
00079 }
00080
00086 bool SH_ApplicationCore::setUser(QString login, QString pass)
00087 {
00088     this->m_currentUser = SH_User::logIn(login, pass);
00089     if(this->m_currentUser->isValid()) {
00090         emit userChanged(QVariant(this->m_currentUser->
name()));
00091         return true;
00092     }
00093     return false;
00094 }
00095
00096
00102 bool SH_ApplicationCore::userExists(QString login)
00103 {
00104     return SH_User::exists(login).toBool();
00105 }
00106
00107
00112 bool SH_ApplicationCore::balanceLogRoutine() {
00113     /*AppDatabase::getInstance()->getDbConnection().exec("execute procedure logPeriodicBalance(H)");
00114     AppDatabase::getInstance()->getDbConnection().exec("execute procedure logPeriodicBalance(D)");
00115     AppDatabase::getInstance()->getDbConnection().exec("execute procedure logPeriodicBalance(W)");
00116     AppDatabase::getInstance()->getDbConnection().exec("execute procedure logPeriodicBalance(M)");
00117     AppDatabase::getInstance()->getDbConnection().exec("execute procedure logPeriodicBalance(Y)");*/
00118 }
00119
00120

```

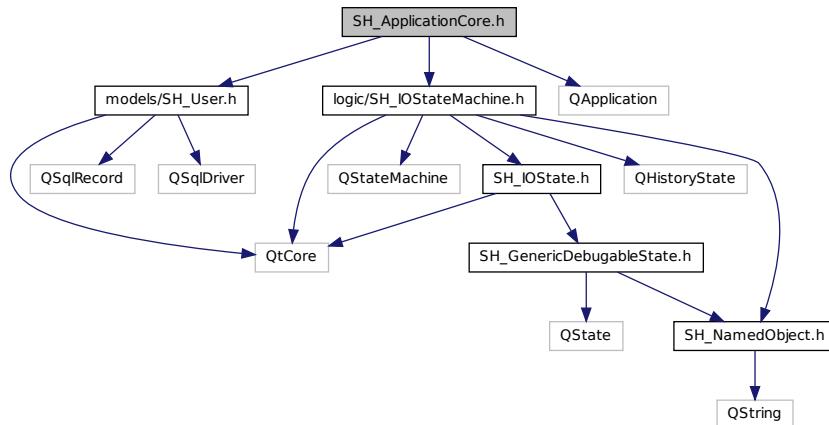
```
00125 void SH_ApplicationCore::receiveInput(QString in)
00126 {
00127     qDebug() << "input received "<<in;
00128     emit this->m_currentFSM->receiveInput(in);
00129 }
00130 }
00131
00136 void SH_ApplicationCore::receiveValidation()
00137 {
00138
00139     emit this->m_currentFSM->validateInput();
00140 }
00141 }
00142
00147 void SH_ApplicationCore::receiveConfirmation()
00148 {
00149
00150     emit this->m_currentFSM->confirmInput();
00151 }
00152 }
00153
00158 void SH_ApplicationCore::replaceInput(QString inputName)
00159 {
00160
00161     emit this->m_currentFSM->replaceInput(inputName);
00162 }
00164
00169 void SH_ApplicationCore::cancelReplacement()
00170 {
00171     if(this->m_currentFSM) {
00172         emit this->m_currentFSM->cancelReplacement();
00173     }
00174 }
00175
00176
00182 bool SH_ApplicationCore::launchBillingsThread()
00183 {
00184     qDebug() << "Hallo !";
00185     /*if(this->m_currentFSM) {
00186         return false;
00187     }*/
00188     qDebug() << "Hallo !";
00189     this->m_currentFSM= new SH_BillingCreationStateMachine("création facturation");
00190     this->m_currentFSM->start();
00191     qDebug() << this->m_currentFSM->toString() << " " << this->
00192     m_currentFSM->initialState();
00192     return this->connectRunningThread();
00193 }
00194 }
00195
00201 bool SH_ApplicationCore::launchBookingsThread()
00202 {
00203     /*if(this->m_currentFSM) {
00204         return false;
00205    >*/
00206     /*this->m_currentFSM= new BookingCreationStateMachine("création facturation");*/
00207     /*this->m_currentFSM->start();*/
00208     return this->connectRunningThread();
00209 }
00210
00216 bool SH_ApplicationCore::launchBillThread()
00217 {
00218     /*if(this->m_currentFSM) {
00219         return false;
00220    >*/
00221     this->m_currentFSM= new SH_ServiceCharging("facturation prestation");
00222     this->m_currentFSM->setContentValue(QVariant(this->
00223     m_currentUser->id()), "BILL_ID");
00223     this->m_currentFSM->start();
00224     return this->connectRunningThread();
00225 }
00226
00232 bool SH_ApplicationCore::cancelRunningThread()
00233 {
00234     /*if(!this->m_currentFSM) {
00235         return true;
00236    >*/
00237     this->m_currentFSM->stop();
00238     bool ok = !this->m_currentFSM->isRunning();
00239     this->m_currentFSM = NULL;
00240     return ok;
00241 }
00242
00243
00249 bool SH_ApplicationCore::connectRunningThread()
```

```

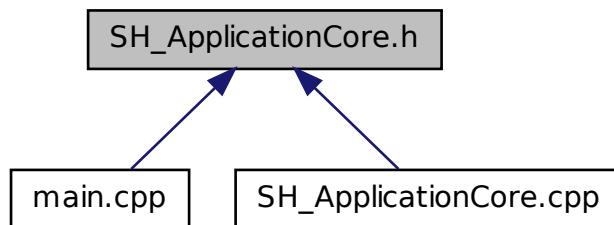
00250 {
00251     /*if(!this->m_currentFSM) {
00252         return false;
00253     }*/
00254     qDebug() << "coucou";
00255     QObject::connect(this->m_currentFSM, &
00256         SH_InOutStateMachine::sendText, this, &
00257         SH_ApplicationCore::sendText, Qt::DirectConnection);
00258     QObject::connect(this->m_currentFSM, &
00259         SH_InOutStateMachine::clearAll, this, &
00260         SH_ApplicationCore::clearAll, Qt::DirectConnection);
00261     QObject::connect(this->m_currentFSM, &
00262         SH_InOutStateMachine::resendText, this, &
00263         SH_ApplicationCore::resendText, Qt::DirectConnection);
00264     QObject::connect(this->m_currentFSM, &
00265         SH_InOutStateMachine::displayCalendar, this, &
00266         SH_ApplicationCore::displayCalendar, Qt::DirectConnection);
00267     return this->m_currentFSM->isRunning();
00268 }
```

5.145 Référence du fichier SH_ApplicationCore.h

```
#include "models/SH_User.h"
#include <QApplication>
#include "logic/SH_IOSMachine.h"
Graphe des dépendances par inclusion de SH_ApplicationCore.h :
```



Ce graphe montre quels fichiers incluent directement ou indirectement ce fichier :



Classes

- class [SH_ApplicationCore](#)

5.146 SH_ApplicationCore.h

```

00001 #ifndef RESTRICTIVEAPPLICATION_H
00002 #define RESTRICTIVEAPPLICATION_H
00003 #include "models/SH_User.h"
00004 #include <QApplication>
00005 #include "logic/SH_IOSStateMachine.h"
00011 class SH_ApplicationCore : public QObject
00012 {
00013     Q_OBJECT
00014     Q_PROPERTY(SH_User* currentUser READ user NOTIFY
00015         userChanged)
00015     Q_PROPERTY(AppMode currentMode READ mode WRITE setMode NOTIFY
00016         modeChanged)
00017     Q_ENUMS(AppMode)
00018
00019 public:
00025     enum AppMode { CONNEXION, ACCUEIL, RECEPTION,
00026         MANAGEMENT_X, MANAGEMENT_Z, ADMINISTRATION };
00032     SH_ApplicationCore(QObject* parent=0);
00039     AppMode mode() const;
00046     SH_User* user() const;
00047     /*Q_INVOKABLE User* currentUser() const;*/
00053     void init();
00060     void setMode(AppMode mode);
00061
00062
00063 public slots:
00069     bool balanceLogRoutine();
00077     bool userExists(QString login);
00086     bool setUser(QString login, QString pass);
00093     bool userLogOut();
00094
00101     bool launchBookingsThread();
00108     bool launchBillThread();
00115     Q_INVOKABLE bool launchBillingsThread();
00122     bool cancelRunningThread();
00123
00129     void receiveInput(QString in);
00130
00135     void receiveValidation();
00136
00141     void receiveConfirmation();
00142
00149     void replaceInput(QString inputName);
00155     void cancelReplacement();
00156
00157 signals:
00163     void clearAll();
00170     void userChanged(QVariant name);
00177     void modeChanged(QVariant mode);
00178
00184     void currentFSMchanged();
00191     void sendText(QString text);
00198     void resendText(QString text);
00204     void displayCalendar();
00211     void openTab(QVariant tabPos);
00212
00213 protected:
00219     bool connectRunningThread();
00220
00221 private:
00225     SH_User* m_currentUser;
00229     AppMode m_mode;
00233     SH_InOutStateMachine* m_currentFSM;
00234 };
00235
00236 #endif /* RESTRICTIVEAPPLICATION_H*/

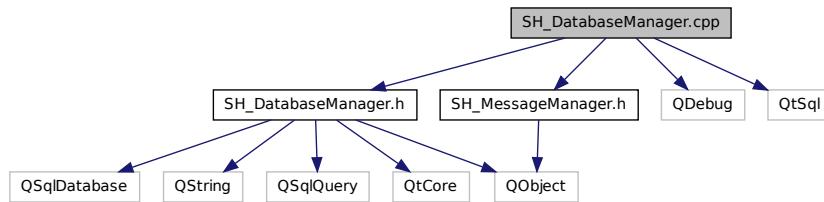
```

5.147 Référence du fichier SH_DatabaseManager.cpp

```
#include "SH_DatabaseManager.h"
```

```
#include "SH_MessageManager.h"
#include <QDebug>
#include <QtSql>
```

Graphe des dépendances par inclusion de SH_DatabaseManager.cpp :



5.148 SH_DatabaseManager.cpp

```

00001 #include "SH_DatabaseManager.h"
00002 #include "SH_MessageManager.h"
00003 #include <QDebug>
00004 #include <QtSql>
00005
00009 SH_DatabaseManager *SH_DatabaseManager::_instance = 0;
00010
00011
00017 SH_DatabaseManager *SH_DatabaseManager::getInstance()
00018 {
00019     if (_instance == 0)
00020     {
00021         _instance = new SH_DatabaseManager;
00022     }
00023     qDebug() << dbFilePathStr;
00024     return _instance;
00025 }
00026
00027
00033 SH_DatabaseManager::~SH_DatabaseManager()
00034 {
00035     dbDisconnect();
00036 }
00037
00038
00044 SH_DatabaseManager::SH_DatabaseManager()
00045 {
00046     /*
00047     *Check the existence of the database driver.
00048     */
00049     if (!QSqlDatabase::isDriverAvailable(dbDriverStr))
00050     {
00051         /*
00052         *Gui message that informs that the driver does not exist
00053         */
00054         SH_MessageManager::errorMessage(
00055             dbDriverNotExistStr);
00056         qDebug() << dbConnection.lastError();
00057         for (int i = 0; i < dbConnection.drivers().count(); i++)
00058         {
00059             qDebug() << "AVAILABLE DRIVERS : " << dbConnection.drivers()[i] << endl;
00060         }
00061         exit(1);
00062     }
00063     /*
00064     *Connect to the database with the following driver.
00065     */
00066     dbConnection = QSqlDatabase::addDatabase(dbDriverStr);
00067     if (dbDriverStr == "QIBASE")
00068     {
00069         dbConnection.setDatabaseName(dbFilePathStr);
00070     } else {
00071         dbConnection.setDatabaseName(dbFileNameStr);
00072     }
00073     dbConnection.setUserName(dbUsernameStr);
```

```

00075     dbConnection.setPassword(dbPasswordStr);
00076     dbConnect();
00077
00078 }
00079 */
00080 /*connect to database
00081 */
00082
00083
00084 bool SH_DatabaseManager::dbConnect()
00090 {
00091     /*
00092         *Open database, if the database cannot open for
00093         *any reason print a warning.
00094     */
00095     if (!dbConnection.open())
00096     {
00097         /*
00098             *Gui message that informs that the database cannot open
00099             */
00100         SH_MessageManager::errorMessage(
00101             dbCannotOpenStr);
00102         qDebug() << dbConnection.lastError();
00103
00104         /*
00105             *@return false if database connection failed.
00106         */
00106         return false;
00107     }
00108
00109 /*
00110     *@return true if database connection successed
00111 */
00112     return dbConnection.isOpen();
00113 }
00114
00115 */
00116     *disconnects from a database
00117 */
00118
00119
00120 bool SH_DatabaseManager::dbDisconnect()
00125 {
00126     /*
00127         *close database
00128         */
00129     dbConnection.close();
00130     return (!dbConnection.isOpen());
00131 }
00132
00133
00134 bool SH_DatabaseManager::isConnected()
00140 {
00141     return dbConnection.isOpen();
00142 }
00143
00144
00145 QSqlDatabase SH_DatabaseManager::getDbConnection()
00151 {
00152     return dbConnection;
00153 }
00154
00155
00156 bool SH_DatabaseManager::tableExists(QString tableName)
00161 {
00162     return dbConnection.tables(SqlTables::Views).contains(tableName.toUpper(), Qt::CaseInsensitive)
00163 || dbConnection.tables(SqlTables::Tables).contains(tableName.toUpper(), Qt::CaseInsensitive);
00164
00165
00166 int SH_DatabaseManager::dataCount(QString tableName, QString filter) {
00167     if(!tableName.isEmpty() && !filter.isEmpty()) {
00168         QSqlQuery result = execSelectQuery(tableName, QStringList("COUNT(*) AS MATCH"),
00169         filter);
00170         if(dbConnection.driver()->hasFeature(SqlDriver::QuerySize)) {
00171             return result.size();
00172         } else {
00173             if(result.next()) {
00174                 QSqlRecord rec = result.record();
00175                 if(!rec.isEmpty() && result.isValid()) {
00176                     return rec.value(rec.indexOf("MATCH")).toInt();
00177                 }
00178             }
00179         }
00180     }
00181 }
00182
00183 }
00184
00185 }
00186
00187
00188 QSqlQuery SH_DatabaseManager::execSelectQuery(QString tableName,

```

```

00194     QStringList fields, QString condition, QString ordering) {
00195         if(fields.isEmpty()) {
00196             fields.append("*");
00197         }
00198         QString query;
00199         if(dbConnection.driverName() == "QIBASE") {
00200             query = QString("SELECT %1 FROM %2").arg(fields.join(", ")).arg(tableName);
00201             if(!condition.isEmpty()) {
00202                 query = QString("%1 WHERE %2").arg(query).arg(condition);
00203             }
00204             if(!ordering.isEmpty()) {
00205                 query = QString("%1 ORDER BY %2").arg(query).arg(ordering);
00206             }
00207         }
00208         qDebug() << query;
00209         QSqlQuery result;
00210         result.exec(query);
00211         qDebug() << result.executedQuery() << " > " << result.isValid() << " << result.isActive();
00212         return result;
00213     }
00214
00215
00220     bool SH_DatabaseManager::execReplaceQuery(QString tableName,
00221         QVariantMap values) {
00222         QString fields;
00223         QString vals;
00224         divideQVariantMap(values, fields, vals);
00225         QString query;
00226         if(dbConnection.driverName() == "QIBASE") {
00227             query = QString("UPDATE OR INSERT INTO %1(%2) VALUES(%3) MATCHING(ID)").arg(tableName).arg(fields).
00228                 arg(vals);
00229         }
00230         QSqlQuery result = dbConnection.exec(query);
00231         qDebug() << result.executedQuery() << " > " << result.isValid() << " << result.isActive();
00232         return (result.numRowsAffected() > 0);
00233     }
00234
00237     QVariant SH_DatabaseManager::execInsertReturningQuery(QString
00238         tableName, QVariantMap values, QString returningField) {
00239         QString fields;
00240         QString vals;
00241         divideQVariantMap(values, fields, vals);
00242         QString query;
00243         if(dbConnection.driverName() == "QIBASE") {
00244             query = QString("UPDATE OR INSERT INTO %1(%2) VALUES(%3) MATCHING(ID) RETURNING %4").arg(tableName)
00245                 .arg(fields).arg(vals).arg(returningField);
00246         }
00247         QSqlQuery result = dbConnection.exec(query);
00248         qDebug() << result.executedQuery() << " > " << result.isValid() << " << result.isActive();
00249         if(result.next()) {
00250             QSqlRecord rec = result.record();
00251             if(!rec.isEmpty() && result.isValid()) {
00252                 return rec.value(rec.indexOf(returningField));
00253             }
00254         }
00255     }
00260     void SH_DatabaseManager::divideQVariantMap(QVariantMap values, QString
00261         & fields, QString& vals) {
00262         for(auto field : values.keys())
00263         {
00264             fields += field+",";
00265             QVariant val = values.value(field);
00266             bool ok;
00267             int intValue = val.toInt(&ok);
00268             if(ok) {
00269                 vals += QString::number(intValue)+",";
00270             }
00271             double dblVal = val.toDouble(&ok);
00272             if(ok) {
00273                 vals += QString::number(dblVal)+",";
00274             }
00275             /*bool boolVal = val.toBool();
00276             if(boolVal) {
00277                 &vals += "'"+1+"'','";
00278             }*/
00279             QDate dateVal = valtoDate();
00280             if(dateVal.isValid()) {
00281                 vals += "'"+dateVal.toString()+"',"; /*FIXME adapt date format*/
00282             }
00283             QDateTime dateTimeVal = valtoDateTime();
00284             if(dateTimeVal.isValid()) {
00285                 vals += "'"+dateTimeVal.toString()+"',"; /*FIXME adapt datetime format*/
00286             }
00287             QString stringVal = val.toString();
00288         }
00289     }

```

```

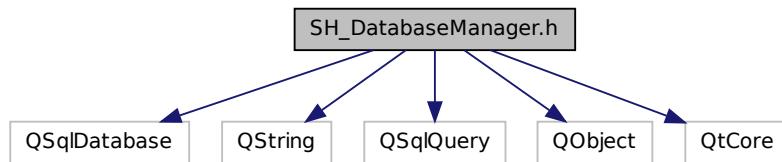
00287         vals += ""+stringVal+"', ";
00288     }
00289     fields = fields.left(fields.lastIndexOf(',')-1);
00290     vals = vals.left(vals.lastIndexOf(',')-1);
00291 }
00292

```

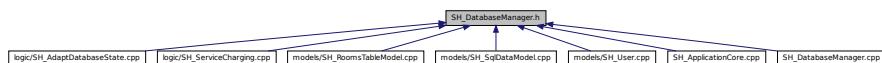
5.149 Référence du fichier SH_DatabaseManager.h

```
#include <QSqlDatabase>
#include <QString>
#include <QSqlQuery>
#include <QObject>
#include <QtCore>
```

Graphe des dépendances par inclusion de SH_DatabaseManager.h :



Ce graphe montre quels fichiers incluent directement ou indirectement ce fichier :



Classes

- class [SH_DatabaseManager](#)

Variables

- static const QString [dbAliasNameStr](#) = "precheck-hotel"
dbAliasNameStr
- static QString [dbCannotOpenStr](#) = QObject::tr("The database %1 cannot be opened.").arg([dbFilePathStr](#))
dbCannotOpenStr
- static QString [dbDriverNotExistStr](#) = QObject::tr("%1 database driver is not available.").arg([dbDriverStr](#))
dbDriverNotExistStr
- static const QString [dbDriverStr](#) = "QIBASE"
dbDriverStr
- static const QString [dbFileNameStr](#) = "PreCheckDB.fdb"
dbFileNameStr
- static const QString [dbFolderPathStr](#) = QString("%1/%2").arg([dbFolderPathStr](#)).arg([dbFileNameStr](#))
dbFolderPathStr
- static const QString [dbFolderPathStr](#) = QDir::cleanPath(QDir::currentPath() + "/../../src/Database/")
dbFolderPathStr
- static const QString [dbPasswordStr](#) = "masterkey"
dbPasswordStr
- static const QString [dbUsernameStr](#) = "SYSDBA"
dbUsernameStr

5.149.1 Documentation des variables

5.149.1.1 const QString dbAliasNameStr = "precheck-hotel" [static]

dbAliasNameStr

Définition à la ligne 23 du fichier [SH_DatabaseManager.h](#).

5.149.1.2 QString dbCannotOpenStr = QObject::tr("The database %1 cannot be opened.").arg(dbFilePathStr) [static]

dbCannotOpenStr

Définition à la ligne 53 du fichier [SH_DatabaseManager.h](#).

Référencé par [SH_DatabaseManager](#) : `:dbConnect()`.

5.149.1.3 QString dbDriverNotExistStr = QObject::tr("%1 database driver is not available.").arg(dbDriverStr) [static]

dbDriverNotExistStr

Définition à la ligne 49 du fichier [SH_DatabaseManager.h](#).

Référencé par [SH_DatabaseManager](#) : `:SH_DatabaseManager()`.

5.149.1.4 const QString dbDriverStr = "QIBASE" [static]

dbDriverStr

Définition à la ligne 15 du fichier [SH_DatabaseManager.h](#).

Référencé par [SH_DatabaseManager](#) : `:SH_DatabaseManager()`.

5.149.1.5 const QString dbFileNameStr = "PreCheckDB.fdb" [static]

dbFileNameStr

Définition à la ligne 19 du fichier [SH_DatabaseManager.h](#).

Référencé par [SH_DatabaseManager](#) : `:SH_DatabaseManager()`.

5.149.1.6 const QString dbFilePathStr = QString("%1/%2").arg(dbFolderPathStr).arg(dbFileNameStr) [static]

dbFilePathStr

Définition à la ligne 40 du fichier [SH_DatabaseManager.h](#).

Référencé par [SH_DatabaseManager](#) : `:getInstance()`, et [SH_DatabaseManager](#) : `:SH_DatabaseManager()`.

5.149.1.7 const QString dbFolderPathStr = QDir::cleanPath(QDir::currentPath() + "/../../src/Database/") [static]

dbFolderPathStr

Définition à la ligne 36 du fichier [SH_DatabaseManager.h](#).

5.149.1.8 const QString dbPasswordStr = "masterkey" [static]

dbPasswordStr

Définition à la ligne 31 du fichier [SH_DatabaseManager.h](#).

Référencé par [SH_DatabaseManager](#) : [:SH_DatabaseManager\(\)](#).

5.149.1.9 const QString dbUsernameStr = "SYSDBA" [static]

dbUsernameStr

Définition à la ligne 27 du fichier [SH_DatabaseManager.h](#).

Référencé par [SH_DatabaseManager](#) : [:SH_DatabaseManager\(\)](#).

5.150 SH_DatabaseManager.h

```

00001 #ifndef APPDATABASESINGLETON_H
00002 #define APPDATABASESINGLETON_H
00003 #include <QSqlDatabase>
00004 #include <QString>
00005 #include <QSqlQuery>
00006 #include <QObject>
00007 #include <QtCore>
00008 /*
00009  * declare DB driver and filename.
00010 */
00015 static const QString dbDriverStr = "QIBASE";
00019 static const QString dbFileNameStr = "PreCheckDB.fdb";
00023 static const QString dbAliasNameStr = "precheck-hotel";
00027 static const QString dbUsernameStr = "SYSDBA";
00031 static const QString dbPasswordStr = "masterkey";
00032
00036 static const QString dbFolderPathStr = QDir::cleanPath(QDir::currentPath() +"
/../../../../src/Database/");
00040 static const QString dbFilePathStr = QString("%1/%2").arg(
dbFolderPathStr).arg(dbFileNameStr);
00041
00042
00043 /*
00044  * GUI string messages.
00045 */
00049 static QString dbDriverNotExistStr = QObject::tr("%1 database driver is not available.")
.arg(dbDriverStr);
00053 static QString dbCannotOpenStr = QObject::tr("The database %1 cannot be opened.").arg(
dbFilePathStr);
00054
00055
00056
00062 class SH_DatabaseManager: public QObject
00063 {
00064     Q_OBJECT
00065     private:
00066         static SH_DatabaseManager *_instance;
00067
00075     void divideQVariantMap(QVariantMap values, QString &fields, QString &vals);
00076 protected:
00077
00083     SH_DatabaseManager();
00087     QSqlDatabase dbConnection;
00088
00089 public:
00090
00091     static SH_DatabaseManager *getInstance();
00099
00100
00101     bool dbConnect();
00109
00116     bool isConnected();
00117
00118
00125     bool dbDisconnect();
00126
00127
00134     QSqlDatabase getDbConnection();
00135
00136
00142     ~SH_DatabaseManager();
00150     bool tableExists(QString tableName);
00159     int dataCount(QString tableName, QString filter);
00167     QSqlQuery execSelectQuery(QString tableName, QStringList fields=QStringList("*"),
QString condition="", QString ordering="");

```

```

00174     bool execReplaceQuery(QString tableName, QVariantMap values);
00182     QVariant execInsertReturningQuery(QString tableName, QVariantMap values,
00183     QString returningField);
00184 };
00185 #endif /* APPDATABASESINGLETON_H */

```

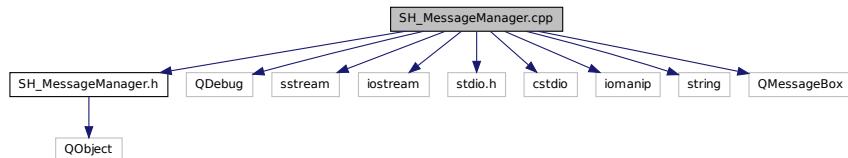
5.151 Référence du fichier SH_MessageManager.cpp

```

#include "SH_MessageManager.h"
#include <QDebug>
#include <sstream>
#include <iostream>
#include <stdio.h>
#include <cstdio>
#include <iomanip>
#include <string>
#include <QMessageBox>

```

Graphe des dépendances par inclusion de SH_MessageManager.cpp :



5.152 SH_MessageManager.cpp

```

00001 #include "SH_MessageManager.h"
00002 #include <QDebug>
00003 #include <sstream>
00004 #include <iostream>
00005 #include <stdio.h>
00006 #include <cstdio>
00007 #include <iomanip>
00008 #include <string>
00009 #include <QMessageBox>
00010
00016 void SH_MessageManager::errorMessage(QString message, QString title)
00017 {
00018     if(!message.isEmpty()) {
00019 #ifdef DEBUG
00020         qWarning() << QObject::tr("%L1 : %L2").arg(title).arg(message);
00021 #else
00022         QMessageBox::critical(0,QObject::tr("%L1").arg(title),QObject::tr("%L2").arg(message));
00023 #endif
00024     }
00025 }
00026
00027
00033 void SH_MessageManager::successMessage(QString message, QString title)
00034 {
00035     if(!message.isEmpty()) {
00036         /*switch(errorMode) {
00037             case DEBUG:*/
00038             qDebug() << QObject::tr("%L1 : %L2").arg(title).arg(message);
00039             /*break;
00040         case TEST:
00041         case RELEASE:
00042             QMessageBox::information(0,QObject::tr("%L1").arg(title),QObject::tr("%L2").arg(message));
00043         default:
00044             console.log(title+" : " + message);
00045         }*/
00046     }
00047 }
00048

```

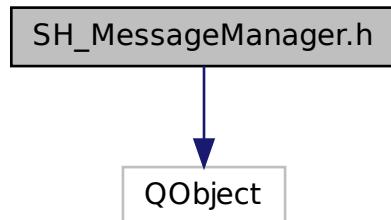
```

00054 void SH_MessageManager::infoMessage(QString message, QString title)
00055 {
00056     if(!message.isEmpty()) {
00057         /*switch(errorMode) {
00058             case DEBUG:/
00059                 qDebug() << QObject::tr("%L1 : %L2").arg(title).arg(message);
00060             /*break;
00061             case TEST:
00062             case RELEASE:
00063                 QMessageBox::information(0,QObject::tr("%L1").arg(title),QObject::tr("%L2").arg(message));
00064             default:
00065                 console.log(title+ " : " + message);
00066         } */
00067     }
00068 }

```

5.153 Référence du fichier SH_MessageManager.h

#include <QObject>
Graphe des dépendances par inclusion de SH_MessageManager.h :



Ce graphe montre quels fichiers incluent directement ou indirectement ce fichier :



Classes

- class [SH_MessageManager](#)

5.154 SH_MessageManager.h

```

00001 #ifndef ERRORMESSAGE_H
00002 #define ERRORMESSAGE_H
00003
00004 #include <QObject>
00005
00011 class SH_MessageManager: QObject
00012 {
00013     Q_OBJECT
00014
00015     Q_ENUMS (ErrorMode)
00016
00017 public:

```

```

00023     enum ErrorMode { ERROR, TEST, DEBUG, DEBUG_VERBOSE,
00024         RELEASE };
00024
00032     static void errorMessage(QString message, QString title = "Erreur");
00040     static void successMessage(QString message, QString title = "Réussite");
00048     static void infoMessage(QString message, QString title = "Info");
00049
00050 };
00051
00052 #endif /* ERRORMESSAGE_H*/

```

5.155 Référence du fichier views/qml/SH_app.qml

Classes

- class [SH_app](#)

5.156 SH_app.qml

```

00001 import QtQuick 2.1
00002 import QtQuick.Window 2.1
00003 import QtQuick.Controls 1.0
00004 import QtQuick.Controls.Styles 1.0
00005 import QtQuick.Layouts 1.0
00006 import PreCheck 1.0
00007
00012 ApplicationWindow {
00013     id: window
00014     visibility: Window.Windowed
00015     /*maximumHeight: 600*/
00016     /*maximumWidth: 800*/
00017     height: maximumHeight
00018     width: maximumWidth
00019     title: "Réception"
00020     property alias pages: stack
00021     signal reload()
00022     statusBar: StatusBar {
00023         Label {
00024             id: status
00025             text: ""
00026             anchors.centerIn: parent
00027         }
00028     }
00029     onReload: {
00030         status.text = Qt.binding(function() {return ((SH_App.currentMode === SH_AppMode.CONNEXION) ? qsTr("Connexion") : (SH_App.currentUser.valid ? SH_App.currentUser.name : ""));});
00031     }
00032     Component.onCompleted: {
00033         window.reload();
00034     }
00035
00036     StackView {
00037         id: stack
00038         signal cycle(Item c)
00039         anchors.fill: parent
00040         Component.onCompleted: {
00041             stack.push({item: commonPage, immediate:true});
00042             stack.push({item: welcomePage, immediate: true});
00043             stack.push({item: connexionPage, immediate: true});
00044         }
00045         onCycle: {
00046             window.reload();
00047             c.reload();
00048             stack.push({item: stack.get(c.Stack.index)});
00049         }
00050         delegate: StackViewDelegate {
00059             function getTransition(properties)
00060             {
00061                 /*return (properties.enterItem.Stack.index % 2) ? fading : bouncing*/
00062                 return fading;
00063             }
00064
00065             property Component fading: StackViewTransition {
00066                 PropertyAnimation {
00067                     target: exitItem
00068                     property: "opacity"
00069                     from: 100
00070                     to: 0

```

```

00071             duration: 50
00072         }
00073         PropertyAnimation {
00074             target: enterItem
00075             property: "opacity"
00076             from: 0
00077             to: 100
00078             duration: 600
00079         }
00080     }
00081 }
00082 }
00083 SH_WelcomePage {
00084     id:welcomePage
00085     objectName: "Welcome"
00086     onQuit: {
00087         Qt.quit();
00088     }
00089     onClicked : {
00090         stack.cycle(commonPage);
00091     }
00092     onLoggedOut: {
00093         stack.cycle(connexionPage);
00094     }
00095 }
00096 SH_ConexionPage {
00097     id: connexionPage
00098     objectName: "Connexion"
00099     onLoggedIn: {
00100         stack.cycle(welcomePage);
00101     }
00102 }
00103 }
00104 SH_CommonPage {
00105     id: commonPage
00106     objectName: "Common"
00107     onQuit: {
00108         stack.cycle(welcomePage);
00109     }
00110 }
00111 }
```

5.157 Référence du fichier views/qml/SH_BillingsDelegate.qml

Classes

- class [SH_BillingsDelegate](#)

5.158 SH_BillingsDelegate.qml

```

00001 import QtQuick 2.1
00002 import QtQuick.Window 2.1
00003 import QtQuick.Controls 1.0
00004 import QtQuick.Controls.Styles 1.0
00005 import QtQuick.Layouts 1.0
00006 import PreCheck 1.0
00007
00011 SH_DataDelegate {
00012     value: ID
00013     text: CLIENT_ID
00014
00015     enabled: dataView.enabled
00016
00017 }
```

5.159 Référence du fichier views/qml/SH_BookingsDelegate.qml

Classes

- class [SH_BookingsDelegate](#)

5.160 SH_BookingsDelegate.qml

```

00001 import QtQuick 2.1
00002 import QtQuick.Window 2.1
00003 import QtQuick.Controls 1.0
00004 import QtQuick.Controls.Styles 1.0
00005 import QtQuick.Layouts 1.0
00006 import PreCheck 1.0
00007
00011 SH_DataDelegate {
00012     value: ID
00013     text: CLIENT_ID
00014
00015     enabled: dataView.enabled
00016
00017 }
```

5.161 Référence du fichier views/qml/SH_CalendarDialog.qml

Classes

- class [SH_CalendarDialog](#)

5.162 SH_CalendarDialog.qml

```

00001 import QtQuick 2.1
00002 import QtQuick.Window 2.1
00003 import QtQuick.Controls 1.0
00004 import QtQuick.Controls.Styles 1.0
00005 import QtQuick.Layouts 1.0
00006 import PreCheck 1.0
00007
00011 Rectangle {
00012     color: "green"
00013     id: calendar
00014     property int firstWeekdayIndex: 1
00015     property string text
00016     readonly property alias currentDate: calendarBase.currentDate
00017     readonly property alias currentWeekday: calendarBase.currentWeekday
00018     readonly property alias currentDay: calendarBase.currentDay
00019     readonly property alias currentMonth: calendarBase.currentMonth
00020     readonly property alias currentYear: calendarBase.currentYear
00021     readonly property alias currentMonthLength: calendarBase.currentMonthLength
00022     readonly property variant monthsList: [qsTr("Janvier"),qsTr("Février"),qsTr("Mars"), qsTr("Avril"),qsTr("Mai"),
("Juin"),qsTr("Juillet"),qsTr("Août"),qsTr("Septembre"),qsTr("Octobre"),qsTr("Novembre"),qsTr("Décembre")]
00023     readonly property variant weekdaysList: [qsTr("Di"),qsTr("Lu"),qsTr("Ma"),qsTr("Me"), qsTr("Je"),qsTr("Ve"),
qsTr("Sa")]
00024     signal clicked(date selectedDate)
00025     signal refresh(date newDate)
00026     signal selected(string selectedDate)
00027     signal closed()
00028     signal opened()
00029     visible: true
00030     onClosed: {
00031         calendar.visible = false;
00032     }
00033     onOpened: {
00034         calendar.visible = true;
00035     }
00036     Component.onCompleted: {
00038         calendar.clicked(new Date());
00039         calendar.refresh(new Date());
00040     }
00041     onClicked: {
00042         /*console.log("selected date "+selectedDate);*/
00043         var year = selectedDate.getFullYear();
00044         var month = selectedDate.getMonth();
00045         var day = selectedDate.getDate();
00046         dateDay.text = day;
00047         dateMonth.text = monthsList[month];
00048         dateYear.text = year;
00049         calendar.text = qsTr("%1 - %2 - %3").arg(day).arg(monthsList[month]).arg(year);
00050     }
00051     onRefresh: {
```

```
00053     calendarBase.ready = false;
00054     /*console.log("refreshed with date "+newDate);*/
00055     calendarBase.currentDate = newDate;
00056     var year = newDate.getFullYear();
00057     var month = newDate.getMonth();
00058     var day = newDate.getDate();
00059     var monthLength = new Date(year, month+1, 0).getDate(); /*javascript Date trick since day count
00060      "starts" at 1 whereas month count starts at 0*/
00061     var firstWeekday = new Date(year, month, 1).getDay();
00062     calendarBase.currentWeekday = newDate.getDay();
00063     calendarBase.currentDay = day;
00064     calendarBase.currentMonth = month;
00065     calendarBase.currentYear = year;
00066     calendarBase.currentMonthLength = monthLength;
00067     calendarBase.currentMonthFirstWeekday = (calendar.firstWeekdayIndex >= 0 && calendar.
00068     firstWeekdayIndex < 7) ? ((firstWeekday - calendar.firstWeekdayIndex) % 7) : firstWeekday;
00069     /*calendarBase.currentMonthLastWeekday = new Date(year, month, new Date(year, month,
00070     0).getDate()).getDay();*/
00071     month.text = qsTr("%1 %2").arg(monthsList[month]).arg(year);
00072     gridRepeater.model = 7 * (Math.ceil(monthLength / 7) + 1);
00073     calendarBase.ready = true;
00074 }
00075 ColumnLayout {
00076     id: calendarBase
00077     property date currentDate: new Date()
00078     property bool ready: true
00079     property string currentWeekday
00080     property string currentDay
00081     property string currentMonth
00082     property string currentYear
00083     property int currentMonthLength
00084     property int currentMonthFirstWeekday
00085     property int currentMonthLastWeekday
00086     spacing: 0
00087     anchors.fill: calendar
00088     RowLayout {
00089         id: dateDisplay
00090         Layout.minimumWidth: childrenRect.width
00091         Layout.preferredHeight: Math.max(Math.max(dateDay.paintedHeight,dateMonth.paintedHeight),
00092         dateYear.paintedHeight)+20
00093         Layout.fillWidth: true
00094         Text {
00095             id: dateDay
00096             Layout.preferredWidth: calendar.width / 3 - 2
00097             horizontalAlignment: Text.AlignHCenter
00098         }
00099         Text {
00100             id: dateMonth
00101             Layout.preferredWidth: calendar.width / 3 - 2
00102             horizontalAlignment: Text.AlignHCenter
00103         }
00104         Text {
00105             id: dateYear
00106             Layout.preferredWidth: calendar.width / 3 - 2
00107             horizontalAlignment: Text.AlignHCenter
00108         }
00109         MouseArea {
00110             id: dateDisplayArea
00111             anchors.fill: dateDisplay
00112             onClicked: {
00113                 calendar.opened();
00114             }
00115         }
00116     }
00117     Rectangle {
00118         id: headRow
00119         color: "red"
00120         Layout.minimumWidth: childrenRect.width
00121         Layout.minimumHeight: month.paintedHeight
00122         Layout.maximumHeight: 3*month.paintedHeight/2
00123         Layout.fillWidth: true
00124         RowLayout {
00125             anchors.fill:headRow
00126             spacing: 0
00127             Image {
00128                 id: previousMonth
00129                 Layout.preferredHeight: headRow.height
00130                 fillMode: Image.PreserveAspectFit
00131                 source: "../img/left.png"
00132                 Layout.alignment: Qt.AlignLeft
00133                 Layout.columnSpan: 1
00134             }
00135         }
00136     }
00137 }
```

```

00136                     id: previousMonthArea
00137                     enabled: calendarBase.ready
00138                     anchors.fill: previousMonth
00139                     onClicked: {
00140                         calendar.refresh(new Date(calendarBase.currentYear, (calendarBase.currentMonth
00141                         - 1 % 12), 1));
00142                     }
00143                     onDoubleClicked: {
00144                         previousMonthArea.clicked(mouse)
00145                     }
00146                 }
00147             Text {
00148                 id: month
00149                 text: qsTr("%1 %L2").arg(monthsList[calendarBase.currentMonth]).arg(calendarBase.
00150                 currentYear);
00151                 Layout.preferredHeight: headRow.height
00152                 Layout.alignment: Qt.AlignHCenter
00153                 Layout.columnSpan: 6
00154             }
00155             Image {
00156                 id:nextMonth
00157                 Layout.alignment: Qt.AlignRight
00158                 Layout.columnSpan: 1
00159                 Layout.preferredHeight: headRow.height
00160                 fillMode: Image.PreserveAspectFit
00161                 source: "../img/right.png"
00162                 MouseArea {
00163                     id: nextMonthArea
00164                     enabled: calendarBase.ready
00165                     anchors.fill: nextMonth
00166                     onClicked: {
00167                         calendar.refresh(new Date(calendarBase.currentYear, (calendarBase.currentMonth
00168                         + 1 % 12), 1));
00169                     }
00170                     onDoubleClicked: {
00171                         nextMonthArea.clicked(mouse)
00172                     }
00173                 }
00174             }
00175             Rectangle {
00176                 id:grid
00177                 color: "pink"
00178                 Layout.fillHeight: true
00179                 Layout.fillWidth: true
00180                 Layout.preferredHeight: childrenRect.height+2
00181                 Layout.preferredWidth: childrenRect.width+2
00182                 GridLayout {
00183                     id: calendarGrid
00184                     anchors.fill:grid
00185                     columns: 7
00186                     flow: Qt.LeftToRight
00187                     /*rows: 1 + Math.ceil(calendarBase.currentMonthLength / 7) */ /*unnecessary because of the
00188                     horizontal flow*/
00189                     columnSpacing: 1
00190                     rowSpacing: 1
00191                     Repeater {
00192                         id: gridRepeater
00193                         /*Layout.fillWidth: true *//*no effect*/
00194                         /*Layout.fillHeight: true *//*no effect*/
00195                         model : 7 * (Math.ceil(calendarBase.currentMonthLength / 7) + 1)
00196                         delegate:
00197                             Rectangle {
00198                                 id: cell
00199                                 /*anchors.fill:parent */ /*hides the other cells*/
00200                                 readonly property real preferredW: (grid.width / 7) - calendarGrid.columnSpacing
00201                                 readonly property real preferredH: (grid.height / (Math.ceil(calendarBase.
00202                                     currentMonthLength / 7) + 1)) - calendarGrid.RowSpacing
00203                                 readonly property real minW: cellText.paintedWidth+3
00204                                 readonly property real minH: cellText.paintedHeight+5
00205                                 Layout.minimumHeight: Math.min(cell.preferredH, cell.minH)
00206                                 Layout.minimumWidth: Math.min(cell.preferredW, cell.minW)
00207                                 Layout.preferredWidth: Math.max(cell.preferredW, cell.minW)
00208                                 Layout.preferredHeight: Math.max(cell.preferredH, cell.minH)
00209                                 readonly property int previousMonthLength: new Date(currentYear, currentMonth, 0).
00210                                 getDate()
00211                                 readonly property int dayIndexInMonth : (index < 7) ? /*1ère ligne (liste des
00212                                     jours de la semaine) ?*/ 0 : /*pas de compteur*/
00213                                         (index - 7 - calendarBase.
00214                                         currentMonthFirstWeekday + 1) /* sinon, on soustrait du compteur les 7 cases sans compteurs, puis de nouveau le
00215                                         nombre de colonnes de décalage avec le début du mois*/
00216                                         property int dayIndex: (index < 7) ? /* 1ère ligne (liste des jours de la semaine)
00217                                         0 : ( /*pas de compteur*/
00218                                         */
00219

```

```

00213                               (dayIndexInMonth <= 0) ? /* avant le
00214                               début du mois ?*/
00215                               dayIndexInMonth + previousMonthLength) : ( /* date dans le mois précédent*/
00216                               (dayIndexInMonth > currentMonthLength) ? /* après la fin du mois ?*/
00217                               (dayIndexInMonth - currentMonthLength)
00218                               : /*date dans le mois suivant (nombre de jours du mois suivant)*/
00219                               dayIndexInMonth /* sinon : date dans le
00220                               mois*/
00221                               )
00222                               )
00223                               color: (index < 7) ? "white" :
00224                               (dayIndexInMonth <= 0 || dayIndexInMonth > currentMonthLength)
00225                               ? /* en dehors du mois ?*/
00226                               "lightgrey" :
00227                               ((cell.dayIndexInMonth === calendarBase.currentDay) ? "blue" : "lightsteelblue")
00228                               border.width: 1
00229                               border.color: "darkblue"
00230
00231                               Text {
00232                                   id: cellText
00233                                   text: (index < 7) ? /* 1ère ligne (liste des jours de la semaine) ?*/
00234                                   calendar.weekdaysList[(index+calendar.firstWeekdayIndex) %
00235                                   7] :
00236                                   cell.dayIndex.toString() /*sinon le numéro de jour*/
00237                                   horizontalAlignment: Text.AlignHCenter
00238                                   anchors.centerIn: cell
00239                               }
00240                               MouseArea {
00241                                   id: cellArea
00242                                   anchors.fill: cell
00243                                   onPressed: {
00244                                       cell.color = "lightblue";
00245                                   }
00246                                   onReleased: {
00247                                       cell.color = (index < 7) ? "white" : ((cell.dayIndexInMonth ===
00248                                       calendarBase.currentDay) ? "blue" : "lightsteelblue");
00249                                   }
00250
00251                                   onClicked: {
00252                                       var d = new Date(calendarBase.currentYear, calendarBase.currentMonth,
00253                                       cellText.text);
00254                                       calendar.clicked(d);
00255                                       calendar.selected(d.toLocaleDateString());
00256                                       calendar.closed();
00257                                   }
00258                               }
00259                           }
00260 }

```

5.163 Référence du fichier views/qml/SH_CommonPage.qml

Classes

- class [SH_CommonPage](#)

5.164 SH_CommonPage.qml

```

00001 import QtQuick 2.1
00002 import QtQuick.Window 2.1
00003 import QtQuick.Controls 1.0
00004 import QtQuick.Controls.Styles 1.0
00005 import QtQuick.Layouts 1.0
00006 import PreCheck 1.0
00007

```

```

00011 Item {
00012     id: commonPage
00013     signal quit();
00014     signal selected(string selectedItem)
00015     signal keySelected(string selectedKey)
00016     signal cancelProcess()
00017     signal confirm()
00018     signal validate()
00019     signal replace(string field)
00020     signal cancelReplace()
00021     signal reload()
00022     property QtObject streamBuffer: buff
00023     QObject {
00024         id: buff
00025         property string content: ""
00026         property bool upperCase: false
00027         signal buffer(string text)
00028         signal eraseLastChar()
00029         signal clearBuffer()
00030         onBuffer: {
00031             var value = (buff.upperCase) ? text.toLocaleUpperCase() : text.toLocaleLowerCase();
00032             if(buff.content === "") {
00033                 buff.content = value;
00034                 rightOutput.displayNew(value, (value.length === 1));
00035             } else {
00036                 buff.content += value;
00037                 rightOutput.display(value);
00038             }
00039         }
00040         onEraseLastChar: {
00041             buff.content = buff.content.slice(0,-1);
00042             rightOutput.replace(buff.content);
00043         }
00044         onClearBuffer: {
00045             commonPage.selected(buff.content);
00046             buff.content = "";
00047         }
00048     }
00049     onReload: {
00050         tabs.reload();
00051     }
00052     onSelected: {
00053         console.log("selection: "+selectedItem);
00054         SH_App.receiveInput(selectedItem);
00055     }
00056     onCancelProcess: {
00057         SH_App.cancelRunningThread();
00058         console.log("processus interrompu");
00059     }
00060     onValidate: {
00061         SH_App.receiveValidation();
00062     }
00063     onConfirm: {
00064         SH_App.receiveConfirmation();
00065     }
00066     onReplace: {
00067         SH_App.replaceInput(field);
00068     }
00069     onCancelReplace: {
00070         SH_App.cancelReplacement();
00071     }
00072
00073     onKeySelected: {
00074         commonPage.streamBuffer.buffer(selectedKey);
00075     }
00076     GridLayout {
00077         id:main
00078         columnSpacing: 2
00079         rowSpacing: 2
00080         width: parent.width
00081         height: parent.height
00082         anchors.fill: parent
00083         flow: GridLayout.TopToBottom
00084         rows: 2
00085         columns: 2
00086         /*la partie supérieure du panel de gauche est remplie par des onglets et leur contenu*/
00087         SH_TabZone {
00088             id: tabs
00089             objectName: "TabView"
00090             Layout.alignment: Qt.AlignTop
00091             Layout.minimumWidth: main.width/2-main.columnSpacing
00092             Layout.minimumHeight: main.height/2-main.rowSpacing
00093             Layout.fillWidth: true
00094             Layout.fillHeight: true
00095             enabled:true
00096             stdKeyboard: [
00097                 char10Action, char11Action, char12Action, char13Action, char14Action, char15Action,

```

```

00098     char16Action, char17Action, char18Action, char19Action, capsAction,
00099         char20Action, char21Action, char22Action, char23Action, char24Action, char25Action,
00100         char26Action, char27Action, char28Action, char29Action, dotAction,
00101         char30Action, char31Action, char32Action, char33Action, char34Action, char35Action,
00102         char36Action, char37Action, char38Action, char39Action, spaceAction,
00103         char40Action, char41Action, char42Action, char43Action, char44Action, char45Action,
00104         char46Action, char47Action, char48Action, char49Action, commaAction,
00105         char50Action, char51Action, char52Action, char53Action, char54Action, char55Action,
00106         char56Action, char57Action, char58Action, char59Action, quoteAction,
00107         char60Action, char61Action, char62Action, char63Action, char64Action, char65Action,
00108         char66Action, char67Action, char68Action, char69Action, quote2Action
00109     ]
00110
00111     onSelected: {
00112         commonPage.keySelected(selectedItem);
00113     }
00114     onSelectedForDetail: {
00115         rightOutput.displaySqlDetail(data);
00116     }
00117 }
00118 /*la partie inférieure du panel de gauche contient le clavier*/
00119 SH_Keyboard{
00120     id: keys
00121     columns: 5
00122     actionsList: [
00123         abandonAction, cancelAction, eraseAction, replaceAction, backAction,
00124         leavingRoomAction, digit7Action, digit8Action, digit9Action, plusAction,
00125         arrivingAction, digit4Action, digit5Action, digit6Action, timesAction,
00126         departureAction, digit1Action, digit2Action, digit3Action, dividesAction,
00127         vatAction, doubleNullAction, nullAction, decimalAction, minusAction,
00128         escapeAction, enterAction, confirmAction, quitAction, helpAction
00129     ]
00130     enabled:true
00131     Layout.alignment: Qt.AlignBottom
00132     Layout.minimumWidth: main.width/2-main.columnSpacing
00133     Layout.minimumHeight: main.height/2-main.rowSpacing
00134     Layout.fillWidth: true
00135     Layout.fillHeight: true
00136 }
00137 /*la zone d'affichage remplit toute la moitié de droite*/
00138 SH_OutputZone {
00139     id: rightOutput
00140     objectName: "RightOutput"
00141     Layout.rowSpan: 2
00142     Layout.minimumWidth: main.width/2-main.columnSpacing
00143     Layout.maximumWidth: main.width/2
00144     Layout.fillHeight: true
00145     onSelected: {
00146         commonPage.keySelected(selectedItem);
00147     }
00148 }
00149
00150 SH_ComplexAction {
00151     id: abandonAction
00152     text: qsTr("VALIDER")
00153     keyShortcut: Qt.Key_unknown
00154     onTriggered: commonPage.validate()
00155 }
00156 SH_ComplexAction {
00157     id: cancelAction
00158     text: qsTr("ANNULER")
00159     keyShortcut: Qt.Key_Cancel
00160     /*onTriggered: *//**TODO*/
00161 }
00162 SH_ComplexAction {
00163     id: eraseAction
00164     text: qsTr("EFFACER")
00165     keyShortcut: Qt.Key_Delete
00166     /*onTriggered: *//**TODO*/
00167 }
00168 SH_ComplexAction {
00169     id: replaceAction
00170     text: qsTr("VENDRE")
00171     keyShortcut: Qt.Key_unknown
00172     onTriggered: tabs.newSelling()
00173 }
00174 SH_ComplexAction {
00175     id: backAction
00176     text: qsTr("RETOUR")
00177     keyShortcut: Qt.Key_Backspace
00178     onTriggered: commonPage.streamBuffer.eraseLastChar();
00179 }
00180 SH_ComplexAction {

```

```
00179      id: leavingRoomAction
00180      text: qsTr("LIBÉRER")
00181      keyShortcut: Qt.Key_unknown
00182
00183      /*onTriggered: *//*TODO*/
00184  }
00185  SH_ComplexAction {
00186      id: digit7Action
00187      text: qsTr("7")
00188      keyShortcut: Qt.Key_7
00189
00190      onTriggered: commonPage.keySelected(text);
00191  }
00192  SH_ComplexAction {
00193      id: digit8Action
00194      text: qsTr("8")
00195      keyShortcut: Qt.Key_8
00196
00197      onTriggered: commonPage.keySelected(text);
00198  }
00199  SH_ComplexAction {
00200      id: digit9Action
00201      text: qsTr("9")
00202      keyShortcut: Qt.Key_9
00203
00204      onTriggered: commonPage.keySelected(text);
00205  }
00206  SH_ComplexAction {
00207      id: plusAction
00208      text: qsTr("+")
00209      keyShortcut: Qt.Key_Plus
00210
00211      onTriggered: commonPage.keySelected(text);
00212  }
00213
00214  SH_ComplexAction {
00215      id: arrivingAction
00216      text: qsTr("ARRIVÉE")
00217      keyShortcut: Qt.Key_unknown
00218      onTriggered: tabs.newBilling();
00219  }
00220  SH_ComplexAction {
00221      id: digit4Action
00222      text: qsTr("4")
00223      keyShortcut: Qt.Key_4
00224
00225      onTriggered: commonPage.keySelected(text);
00226  }
00227  SH_ComplexAction {
00228      id: digit5Action
00229      text: qsTr("5")
00230      keyShortcut: Qt.Key_5
00231
00232      onTriggered: commonPage.keySelected(text);
00233  }
00234  SH_ComplexAction {
00235      id: digit6Action
00236      text: qsTr("6")
00237      keyShortcut: Qt.Key_6
00238
00239      onTriggered: commonPage.keySelected(text);
00240  }
00241  SH_ComplexAction {
00242      id: timesAction
00243      text: qsTr("*")
00244      keyShortcut: Qt.Key_multiply
00245
00246      onTriggered: commonPage.keySelected(text);
00247  }
00248
00249  SH_ComplexAction {
00250      id: departureAction
00251      text: qsTr("DÉPART")
00252      keyShortcut: Qt.Key_unknown
00253      /*onTriggered: *//*TODO*/
00254  }
00255  SH_ComplexAction {
00256      id: digit1Action
00257      text: qsTr("1")
00258      keyShortcut: Qt.Key_1
00259
00260      onTriggered: commonPage.keySelected(text);
00261  }
00262  SH_ComplexAction {
00263      id: digit2Action
00264      text: qsTr("2")
00265      keyShortcut: Qt.Key_2
```

```
00266      onTriggered: commonPage.keySelected(text);
00267  }
00268  SH_ComplexAction {
00269      id: digit3Action
00270      text: qsTr("3")
00271      keyShortcut: Qt.Key_3
00272
00273      onTriggered: commonPage.keySelected(text);
00274  }
00275  SH_ComplexAction {
00276      id: dividesAction
00277      text: qsTr("/")
00278      keyShortcut: Qt.Key_division
00279
00280      onTriggered: commonPage.keySelected(text);
00281  }
00282  SH_ComplexAction {
00283      id: vatAction
00284      text: qsTr("TVA")
00285      keyShortcut: Qt.Key_unknown
00286
00287      /*onTriggered: */ /*TODO*/
00288  }
00289  SH_ComplexAction {
00290      id: doubleNullAction
00291      text: qsTr("00")
00292      keyShortcut: Qt.Key_unknown
00293
00294      onTriggered: commonPage.keySelected(text);
00295  }
00296  SH_ComplexAction {
00297      id: nullAction
00298      text: qsTr("0")
00299      keyShortcut: Qt.Key_0
00300
00301      onTriggered: commonPage.keySelected(text);
00302  }
00303  SH_ComplexAction {
00304      id: decimalAction
00305      text: qsTr(",")
00306      keyShortcut: Qt.Key_Period
00307
00308      onTriggered: commonPage.keySelected(text);
00309  }
00310  SH_ComplexAction {
00311      id: minusAction
00312      text: qsTr("-")
00313      keyShortcut: Qt.Key_Minus
00314
00315      onTriggered: commonPage.keySelected(text);
00316  }
00317
00318  SH_ComplexAction {
00319      id: escapeAction
00320      text: qsTr("ÉCHAP")
00321      keyShortcut: Qt.Key_Escape
00322      onTriggered: SH_CommonPage.cancelProcess()
00323  }
00324  SH_ComplexAction {
00325      id: enterAction
00326      text: qsTr("ENTRÉE")
00327      keyShortcut: Qt.Key_Enter
00328      onTriggered: SH_CommonPage.streamBuffer.clearBuffer();
00329  }
00330  SH_ComplexAction {
00331      id: confirmAction
00332      text: qsTr("CONFIRMER")
00333      keyShortcut: Qt.Key_Return
00334      onTriggered: {
00335          SH_CommonPage.confirm();
00336      }
00337  }
00338  SH_ComplexAction {
00339      id: quitAction
00340      text: qsTr("QUITTER")
00341      keyShortcut: Qt.Key_unknown
00342      onTriggered: {
00343          SH_CommonPage.cancelProcess();
00344          SH_CommonPage.quit();
00345      }
00346  }
00347
00348  SH_ComplexAction {
00349      id: helpAction
00350      text: qsTr("AIDE")
00351      keyShortcut: Qt.Key_unknown
00352      /*onTriggered: */ /*TODO*/
```

```
00353     }
00354
00355
00356     SH_ComplexAction {
00357         id: char10Action
00358         text: qsTr("Q")
00359         keyShortcut: Qt.Key_Q
00360
00361         onTriggered: commonPage.keySelected(text.toLocaleLowerCase());
00362     }
00363     SH_ComplexAction {
00364         id: char11Action
00365         text: qsTr("W")
00366         keyShortcut: Qt.Key_W
00367
00368         onTriggered: commonPage.keySelected(text.toLocaleLowerCase());
00369     }
00370     SH_ComplexAction {
00371         id: char12Action
00372         text: qsTr("E")
00373         keyShortcut: Qt.Key_E
00374
00375         onTriggered: commonPage.keySelected(text.toLocaleLowerCase());
00376     }
00377     SH_ComplexAction {
00378         id: char13Action
00379         text: qsTr("R")
00380         keyShortcut: Qt.Key_R
00381
00382         onTriggered: commonPage.keySelected(text.toLocaleLowerCase());
00383     }
00384     SH_ComplexAction {
00385         id: char14Action
00386         text: qsTr("T")
00387         keyShortcut: Qt.Key_T
00388
00389         onTriggered: commonPage.keySelected(text.toLocaleLowerCase());
00390     }
00391     SH_ComplexAction {
00392         id: char15Action
00393         text: qsTr("Z")
00394         keyShortcut: Qt.Key_Z
00395
00396         onTriggered: commonPage.keySelected(text.toLocaleLowerCase());
00397     }
00398     SH_ComplexAction {
00399         id: char16Action
00400         text: qsTr("U")
00401         keyShortcut: Qt.Key_U
00402
00403         onTriggered: commonPage.keySelected(text.toLocaleLowerCase());
00404     }
00405     SH_ComplexAction {
00406         id: char17Action
00407         text: qsTr("I")
00408         keyShortcut: Qt.Key_I
00409
00410         onTriggered: commonPage.keySelected(text.toLocaleLowerCase());
00411     }
00412     SH_ComplexAction {
00413         id: char18Action
00414         text: qsTr("O")
00415         keyShortcut: Qt.Key_O
00416
00417         onTriggered: commonPage.keySelected(text.toLocaleLowerCase());
00418     }
00419     SH_ComplexAction {
00420         id: char19Action
00421         text: qsTr("P")
00422         keyShortcut: Qt.Key_P
00423
00424         onTriggered: commonPage.keySelected(text.toLocaleLowerCase());
00425     }
00426
00427     SH_ComplexAction {
00428         id: char20Action
00429         text: qsTr("É")
00430         keyShortcut: Qt.Key_Eacute
00431
00432         onTriggered: commonPage.keySelected(text.toLocaleLowerCase());
00433     }
00434     SH_ComplexAction {
00435         id: char21Action
00436         text: qsTr("È")
00437         keyShortcut: Qt.Key_Egrave
00438
00439         onTriggered: commonPage.keySelected(text.toLocaleLowerCase());
```

```
00440 }
00441     SH_ComplexAction {
00442         id: char22Action
00443         text: qsTr("Ê")
00444         keyShortcut: Qt.Key_Ecircumflex
00445
00446         onTriggered: commonPage.keySelected(text.toLocaleLowerCase());
00447     }
00448     SH_ComplexAction {
00449         id: char23Action
00450         text: qsTr("Ë")
00451         keyShortcut: Qt.Key_Ediaeresis
00452
00453         onTriggered: commonPage.keySelected(text.toLocaleLowerCase());
00454     }
00455     SH_ComplexAction {
00456         id: char24Action
00457         text: qsTr("À")
00458         keyShortcut: Qt.Key_Agrave
00459
00460         onTriggered: commonPage.keySelected(text.toLocaleLowerCase());
00461     }
00462     SH_ComplexAction {
00463         id: char25Action
00464         text: qsTr("Ã")
00465         keyShortcut: Qt.Key_Acircumflex
00466
00467         onTriggered: commonPage.keySelected(text.toLocaleLowerCase());
00468     }
00469     SH_ComplexAction {
00470         id: char26Action
00471         text: qsTr("Ã")
00472         keyShortcut: Qt.Key_Adiaeresis
00473
00474         onTriggered: commonPage.keySelected(text.toLocaleLowerCase());
00475     }
00476     SH_ComplexAction {
00477         id: char27Action
00478         text: qsTr("Í")
00479         keyShortcut: Qt.Key_Idiaeresis
00480
00481         onTriggered: commonPage.keySelected(text.toLocaleLowerCase());
00482     }
00483     SH_ComplexAction {
00484         id: char28Action
00485         text: qsTr("Ӯ")
00486         keyShortcut: Qt.Key_Ugrave
00487
00488         onTriggered: commonPage.keySelected(text.toLocaleLowerCase());
00489     }
00490     SH_ComplexAction {
00491         id: char29Action
00492         text: qsTr("Ӱ")
00493         keyShortcut: Qt.Key_Ucircumflex
00494
00495         onTriggered: commonPage.keySelected(text.toLocaleLowerCase());
00496     }
00497
00498     SH_ComplexAction {
00499         id: char30Action
00500         text: qsTr("A")
00501         keyShortcut: Qt.Key_A
00502
00503         onTriggered: commonPage.keySelected(text.toLocaleLowerCase());
00504     }
00505     SH_ComplexAction {
00506         id: char31Action
00507         text: qsTr("S")
00508         keyShortcut: Qt.Key_S
00509
00510         onTriggered: commonPage.keySelected(text.toLocaleLowerCase());
00511     }
00512     SH_ComplexAction {
00513         id: char32Action
00514         text: qsTr("D")
00515         keyShortcut: Qt.Key_D
00516
00517         onTriggered: commonPage.keySelected(text.toLocaleLowerCase());
00518     }
00519     SH_ComplexAction {
00520         id: char33Action
00521         text: qsTr("F")
00522         keyShortcut: Qt.Key_F
00523
00524         onTriggered: commonPage.keySelected(text.toLocaleLowerCase());
00525     }
00526     SH_ComplexAction {
```

```
00527     id: char34Action
00528     text: qsTr("G")
00529     keyShortcut: Qt.Key_G
00530
00531     onTriggered: commonPage.keySelected(text.toLocaleLowerCase());
00532 }
00533 SH_ComplexAction {
00534     id: char35Action
00535     text: qsTr("H")
00536     keyShortcut: Qt.Key_H
00537
00538     onTriggered: commonPage.keySelected(text.toLocaleLowerCase());
00539 }
00540 SH_ComplexAction {
00541     id: char36Action
00542     text: qsTr("J")
00543     keyShortcut: Qt.Key_J
00544
00545     onTriggered: commonPage.keySelected(text.toLocaleLowerCase());
00546 }
00547 SH_ComplexAction {
00548     id: char37Action
00549     text: qsTr("K")
00550     keyShortcut: Qt.Key_K
00551
00552     onTriggered: commonPage.keySelected(text.toLocaleLowerCase());
00553 }
00554 SH_ComplexAction {
00555     id: char38Action
00556     text: qsTr("L")
00557     keyShortcut: Qt.Key_L
00558
00559     onTriggered: commonPage.keySelected(text.toLocaleLowerCase());
00560 }
00561 SH_ComplexAction {
00562     id: char39Action
00563     text: qsTr("Ӧ")
00564     keyShortcut: Qt.Key_Odiaeresis
00565
00566     onTriggered: commonPage.keySelected(text.toLocaleLowerCase());
00567 }
00568
00569 SH_ComplexAction {
00570     id: char40Action
00571     text: qsTr("Ӣ")
00572     keyShortcut: Qt.Key_Y
00573
00574     onTriggered: commonPage.keySelected(text.toLocaleLowerCase());
00575 }
00576 SH_ComplexAction {
00577     id: char41Action
00578     text: qsTr("X")
00579     keyShortcut: Qt.Key_X
00580
00581     onTriggered: commonPage.keySelected(text.toLocaleLowerCase());
00582 }
00583 SH_ComplexAction {
00584     id: char42Action
00585     text: qsTr("C")
00586     keyShortcut: Qt.Key_C
00587
00588     onTriggered: commonPage.keySelected(text.toLocaleLowerCase());
00589 }
00590 SH_ComplexAction {
00591     id: char43Action
00592     text: qsTr("V")
00593     keyShortcut: Qt.Key_V
00594
00595     onTriggered: commonPage.keySelected(text.toLocaleLowerCase());
00596 }
00597 SH_ComplexAction {
00598     id: char44Action
00599     text: qsTr("B")
00600     keyShortcut: Qt.Key_B
00601
00602     onTriggered: commonPage.keySelected(text.toLocaleLowerCase());
00603 }
00604 SH_ComplexAction {
00605     id: char45Action
00606     text: qsTr("N")
00607     keyShortcut: Qt.Key_N
00608
00609     onTriggered: commonPage.keySelected(text.toLocaleLowerCase());
00610 }
00611 SH_ComplexAction {
00612     id: char46Action
00613     text: qsTr("M")
```

```
00614     keyShortcut: Qt.Key_M
00615
00616     onTriggered: commonPage.keySelected(text.toLocaleLowerCase());
00617 }
00618 SH_ComplexAction {
00619     id: char47Action
00620     text: qsTr("í")
00621     keyShortcut: Qt.Key_Icircumflex
00622
00623     onTriggered: commonPage.keySelected(text.toLocaleLowerCase());
00624 }
00625 SH_ComplexAction {
00626     id: char48Action
00627     text: qsTr("ç")
00628     keyShortcut: Qt.Key_Ccedilla
00629
00630     onTriggered: commonPage.keySelected(text.toLocaleLowerCase());
00631 }
00632 SH_ComplexAction {
00633     id: char49Action
00634     text: qsTr("ô")
00635     keyShortcut: Qt.Key_Ocircumflex
00636
00637     onTriggered: commonPage.keySelected(text.toLocaleLowerCase());
00638 }
00639
00640 SH_ComplexAction {
00641     id: char50Action
00642     text: qsTr("ñ")
00643     keyShortcut: Qt.Key_Ntilde
00644
00645     onTriggered: commonPage.keySelected(text.toLocaleLowerCase());
00646 }
00647 SH_ComplexAction {
00648     id: char51Action
00649     text: qsTr("&")
00650     keyShortcut: Qt.Key_Ampersand
00651
00652     onTriggered: commonPage.keySelected(text.toLocaleLowerCase());
00653 }
00654 SH_ComplexAction {
00655     id: char52Action
00656     text: qsTr("@")
00657     keyShortcut: Qt.Key_At
00658
00659     onTriggered: commonPage.keySelected(text.toLocaleLowerCase());
00660 }
00661 SH_ComplexAction {
00662     id: char53Action
00663     text: qsTr("+")
00664     keyShortcut: Qt.Key_Plus
00665
00666     onTriggered: commonPage.keySelected(text.toLocaleLowerCase());
00667 }
00668 SH_ComplexAction {
00669     id: char54Action
00670     text: qsTr("_")
00671     keyShortcut: Qt.Key_Underscore
00672
00673     onTriggered: commonPage.keySelected(text.toLocaleLowerCase());
00674 }
00675 SH_ComplexAction {
00676     id: char55Action
00677     text: qsTr("(")
00678     keyShortcut: Qt.Key_BracketLeft
00679
00680     onTriggered: commonPage.keySelected(text.toLocaleLowerCase());
00681 }
00682 SH_ComplexAction {
00683     id: char56Action
00684     text: qsTr(")")
00685     keyShortcut: Qt.Key_BracketRight
00686
00687     onTriggered: commonPage.keySelected(text.toLocaleLowerCase());
00688 }
00689 SH_ComplexAction {
00690     id: char57Action
00691     text: qsTr("-")
00692     keyShortcut: Qt.Key_Hyphen
00693
00694     onTriggered: commonPage.keySelected(text.toLocaleLowerCase());
00695 }
00696 SH_ComplexAction {
00697     id: char58Action
00698     text: qsTr("?")
00699     keyShortcut: Qt.Key_Question
00700
```

```

00701     onTriggered: commonPage.keySelected(text.toLocaleLowerCase());
00702 }
00703 SH_ComplexAction {
00704     id: char59Action
00705     text: qsTr("!")
00706     keyShortcut: Qt.Key_Exclam
00707
00708     onTriggered: commonPage.keySelected(text.toLocaleLowerCase());
00709 }
00710
00711 SH_ComplexAction {
00712     id: char60Action
00713     text: qsTr("%")
00714     keyShortcut: Qt.Key_Percent
00715     onTriggered: commonPage.keySelected(text.toLocaleLowerCase());
00716 }
00717 SH_ComplexAction {
00718     id: char61Action
00719     text: qsTr(":")
00720     keyShortcut: Qt.Key_Semicolon
00721     onTriggered: commonPage.keySelected(text.toLocaleLowerCase());
00722 }
00723 SH_ComplexAction {
00724     id: char62Action
00725     text: qsTr("<")
00726     keyShortcut: Qt.Key_Less
00727     onTriggered: commonPage.keySelected(text.toLocaleLowerCase());
00728 }
00729 SH_ComplexAction {
00730     id: char63Action
00731     text: qsTr(">")
00732     keyShortcut: Qt.Key_Greater
00733     onTriggered: commonPage.keySelected(text.toLocaleLowerCase());
00734 }
00735 SH_ComplexAction {
00736     id: char64Action
00737     text: qsTr("\\")

00738     keyShortcut: Qt.Key_Backslash
00739     onTriggered: commonPage.keySelected(text.toLocaleLowerCase());
00740 }
00741 SH_ComplexAction {
00742     id: char65Action
00743     text: qsTr("/")
00744     keyShortcut: Qt.Key_Slash
00745     onTriggered: commonPage.keySelected(text.toLocaleLowerCase());
00746 }
00747 SH_ComplexAction {
00748     id: char66Action
00749     text: qsTr(">")
00750     keyShortcut: Qt.Key_Equal
00751     onTriggered: commonPage.keySelected(text.toLocaleLowerCase());
00752 }
00753 SH_ComplexAction {
00754     id: char67Action
00755     text: qsTr("*")
00756     keyShortcut: Qt.Key_Asterisk
00757     onTriggered: commonPage.keySelected(text.toLocaleLowerCase());
00758 }
00759 SH_ComplexAction {
00760     id: char68Action
00761     text: qsTr("°")
00762     keyShortcut: Qt.Key_degree
00763     onTriggered: commonPage.keySelected(text.toLocaleLowerCase());
00764 }
00765 SH_ComplexAction {
00766     id: char69Action
00767     text: qsTr(" ; ")
00768     keyShortcut: Qt.Key_Colon
00769     onTriggered: commonPage.keySelected(text.toLocaleLowerCase());
00770 }
00771 SH_ComplexAction {
00772     id: capsAction
00773     text: qsTr("MAJ .")
00774     keyShortcut: Qt.Key_CapsLock
00775     onTriggered: SH_CommonPage.streamBuffer.upperCase = !
00776     SH_CommonPage.streamBuffer.upperCase;
00777 }
00778 SH_ComplexAction {
00779     id: dotAction
00780     text: qsTr(".")
00781     keyShortcut: Qt.Key_Period
00782     onTriggered: commonPage.keySelected(text.toLocaleLowerCase());
00783 }
00784 SH_ComplexAction {
00785     id: commaAction
00786     text: qsTr(",")
00787     keyShortcut: Qt.Key_Comma

```

```

00787     onTriggered: commonPage.keySelected(text.toLocaleLowerCase());
00788 }
00789 SH_ComplexAction {
00790     id: quoteAction
00791     text: qsTr("''")
00792     keyShortcut: Qt.Key_Apostrophe
00793     onTriggered: commonPage.keySelected(text.toLocaleLowerCase());
00794 }
00795 SH_ComplexAction {
00796     id: quote2Action
00797     text: qsTr("\"")
00798     keyShortcut: Qt.Key_QuoteDbl
00799     onTriggered: commonPage.keySelected(text.toLocaleLowerCase());
00800 }
00801 SH_ComplexAction {
00802     id: spaceAction
00803     text: qsTr("ESPACE")
00804     keyShortcut: Qt.Key_Space
00805     onTriggered: commonPage.keySelected(" ");
00806 }
00807 }
00808

```

5.165 Référence du fichier views/qml/SH_ConexionPage.qml

Classes

- class [SH_ConexionPage](#)

5.166 SH_ConexionPage.qml

```

00001 import QtQuick 2.1
00002 import QtQuick.Window 2.1
00003 import QtQuick.Controls 1.0
00004 import QtQuick.Controls.Styles 1.0
00005 import QtQuick.Layouts 1.0
00006 import PreCheck 1.0
00007
00008 Item {
00009     id: connexionPage
00010     signal reload()
00011     signal checkUsername(string name)
00012     signal logIn(string login, string password)
00013     signal loggedIn
00014
00015     onCheckUsername: {
00016         if(!SH_App.userExists(name)) {
00017             errorLabel.text=qsTr("Cet utilisateur n'existe pas");
00018             errorLabel.visible = true;
00019             login.textColor="darkred";
00020         }
00021     }
00022
00023     onLogIn: {
00024         if(SH_App.setUser(login, password)) {
00025             errorLabel.visible=false;
00026             errorLabel.text="";
00027             SH_App.currentMode = SH_AppMode.ACCEUIL;
00028             connexionPage.loggedIn();
00029         } else {
00030             errorLabel.text=qsTr("Le mot de passe entré est incorrect");
00031             errorLabel.visible = true;
00032         }
00033     }
00034
00035     GridLayout {
00036         anchors.centerIn: parent
00037         columns: 2
00038         columnSpacing: 10
00039         rowSpacing: 3
00040         Label {
00041             id: errorLabel
00042             visible: false
00043             color: "red"
00044             font.pointSize: 15
00045             Layout.minimumHeight: paintedHeight+2
00046             Layout.minimumWidth: paintedWidth
00047             Layout.columnSpan: 2
00048
00049
00050

```

```

00051         }
00052     Label {
00053         text: qsTr("Nom d'utilisateur")
00054         Layout.minimumHeight: paintedHeight
00055     }
00056     TextField {
00057         id: login
00058         activeFocusOnPress: true
00059         onAccepted: {
00060             connexionPage.checkUsername(login.text);
00061             password.forceActiveFocus();
00062         }
00063         onFocusChanged: {
00064             if(!focus) {
00065                 accepted();
00066             } else {
00067                 textColor="black";
00068                 errorLabel.text="";
00069                 errorLabel.visible = false;
00070             }
00071         }
00072         Layout.maximumHeight: 20
00073     }
00074     Label {
00075         text: qsTr("Mot de passe")
00076         Layout.minimumHeight: paintedHeight
00077     }
00078     TextField {
00079         id:password
00080         echoMode: TextInput.Password
00081
00082         Layout.maximumHeight: 20
00083         onFocusChanged: {
00084             if(!focus) {
00085                 accepted();
00086             } else {
00087                 errorLabel.text="";
00088                 errorLabel.visible = false;
00089             }
00090         }
00091     }
00092 }
00093 Button {
00094     text: qsTr("Connexion")
00095     Layout.minimumHeight: childrenRect.height
00096     Layout.minimumWidth: childrenRect.width
00097     Layout.columnSpan: 2
00098     onClicked: {
00099         connexionPage.logIn(login.text, password.text);
00100         login.text = "";
00101         password.text = "";
00102     }
00103 }
00104 }
00105 }

```

5.167 Référence du fichier views/qml/SH_ContentView.qml

Classes

- class [SH_ContentView](#)

5.168 SH_ContentView.qml

```

00001 import QtQuick 2.1
00002 import QtQuick.Window 2.1
00003 import QtQuick.Controls 1.0
00004 import QtQuick.Controls.Styles 1.0
00005 import QtQuick.Layouts 1.0
00006 import PreCheck 1.0
00007
00011 GridLayout {
00012     id: dataView
00013     columns: 5
00014     columnSpacing: 1
00015     rowSpacing: 1
00016     property string SH_DataDelegate
00017     property string emptyDelegate

```

```

00018     property string sectionDelegate
00019     property variant activeFilterIndicatorIndexes
00020     property alias sectionIndex : repeater.sectionIndex
00021     property alias model: repeater.model
00022     signal selected(string selectedItem)
00023     function removeFilterIndex(index) {
00024         if(!dataView.activeFilterIndicatorIndexes) {
00025             dataView.activeFilterIndicatorIndexes = [];
00026         }
00027         var tmp = dataView.activeFilterIndicatorIndexes
00028         var pos = tmp.indexOf(index);
00029         tmp.splice(pos, 1);
00030         return tmp;
00031     }
00032     function addFilterIndex(index) {
00033         if(!dataView.activeFilterIndicatorIndexes) {
00034             dataView.activeFilterIndicatorIndexes = [];
00035         }
00036         var tmp = dataView.activeFilterIndicatorIndexes
00037         tmp.push(index);
00038         return tmp;
00039     }
00040     function sort(index) {
00041         repeater.model.setSortKeyColumn(index);
00042     }
00043     function filter(index, remove) {
00044         if(remove) {
00045             dataView.activeFilterIndicatorIndexes = dataView.addFilterIndex(index);
00046             repeater.model.setFilterKeyColumn(index);
00047         } else {
00048             dataView.activeFilterIndicatorIndexes = dataView.removeFilterIndex(index);
00049             var nbFilters = dataView.activeFilterIndicatorIndexes.length;
00050             repeater.model.invalidateFilter(index);
00051             for(var i = 0; i < nbFilters; i++) {
00052                 repeater.model.setFilterKeyColumn(dataView.activeFilterIndicatorIndexes.at(i));
00053             }
00054         }
00055     }
00056 }
00057
00058 Repeater {
00059     id: repeater
00060     property bool sectioning: (sectionIndex != 0)
00061     property int sectionIndex : repeater.model==0 ? 0: repeater.model.sortKeyColumn
00062     /*property int currentIndex: 0
00063     property int currentSectionSize: 0
00064     property int previousSectionSize: 0
00065     delegate:
00066         Item {
00067             Binding {
00068                 target: repeater
00069                 property: "currentIndex"
00070                 value: index
00071             }
00072             Item {
00073                 id: filler
00074                 visible: false
00075                 state: (repeater.currentIndex > 0 && repeater.model.data(repeater.currentIndex-1,
00076 repeater.sectionIndex) === repeater.model.data(repeater.currentIndex, repeater.sectionIndex)) ? "sameSection":
00077 "newSection"
00078                 states: [
00079                     State {
00080                         name:"newSection"
00081                         PropertyChanges {
00082                             target: filler
00083                             visible: true
00084                         }
00085                     },
00086                     State {
00087                         name:"sameSection"
00088                         PropertyChanges {
00089                             target: filler
00090                             visible: false
00091                         }
00092                     }
00093                 ]
00094                 onStateChanged: {
00095                     console.log("\n"+filler.state)
00096                     if(filler.state == "sameSection") {
00097                         var previousSize = repeater.currentSectionSize;
00098                         filler.cols = 0;
00099                         repeater.currentSectionSize = previousSize+1;
00100                         repeater.previousSectionSize = previousSize;
00101                     } else if(filler.state == "newSection") {
00102                         repeater.previousSectionSize = repeater.currentSectionSize;
00103                         repeater.currentSectionSize = 1;
00104                         filler.cols = ((dataView.columns-(repeater.previousSectionSize % dataView.columns))
00105
00106             % dataView.columns);
00107         }
00108     }
00109 }
```

```

00134
00135         console.log("nouvelle section '" + repeater.model.data(repeater.currentIndex,
00136         repeater.sectionIndex)+"'");
00137         console.log(filler.cols + "/" + dataView.columns + " colonnes à remplir");
00138     }
00139     console.log("previous section size: "+repeater.previousSectionSize);
00140     console.log("next section size: " +repeater.currentSectionSize);
00141   }
00142   Component.onCompleted: {
00143     console.log("index : "+repeater.currentIndex);
00144     console.log("index in section : "+repeater.currentSectionSize);
00145   }
00146   property int cols: 0
00147   Layout.fillHeight: true
00148   Layout.fillWidth: true
00149   Layout.minimumWidth: cols*(dataView.width / dataView.columns - dataView.columnSpacing)
00150   Layout.rowSpan: cols
00151   ColumnLayout {
00152     anchors.fill: parent
00153     Repeater {
00154       model: (filler.cols > 0) ? filler.cols : 0
00155       delegate:
00156         Rectangle {
00157           id: emptyContainer
00158           Layout.fillHeight: true
00159           Layout.fillWidth: true
00160           Layout.preferredHeight: (dataView.rows > 0) ? (Math.floor(dataView.height /
00161             dataView.rows) - dataView.rowSpacing) : (Math.floor((dataView.height * dataView.columns) / repeater.count) -
00162             dataView.rowSpacing)
00163           Layout.preferredWidth: (dataView.columns > 0) ? (Math.floor(dataView.width /
00164             dataView.columns) - dataView.columnSpacing) : (Math.floor((dataView.width * dataView.rows) / repeater.count) -
00165             dataView.columnSpacing)
00166           Loader {
00167             id: emptyLoader
00168             source: dataView.emptyDelegate
00169             Binding {
00170               target: emptyLoader.item
00171               property: "anchors.centerIn"
00172               value: emptyContainer
00173             }
00174             Binding {
00175               target: emptyLoader.item
00176               property: "width"
00177               value: emptyContainer.width - 1
00178             }
00179             Binding {
00180               target: emptyLoader.item
00181               property: "height"
00182               value: emptyContainer.height - 1
00183             }
00184             Binding {
00185               target: emptyLoader.item
00186               property: "enabled"
00187               value: false
00188             }
00189             Binding {
00190               target: emptyLoader.item
00191               property: "visible"
00192               value: true
00193             }
00194           Item {
00195             id: section
00196             Layout.fillHeight: true
00197             Layout.fillWidth: true
00198             Layout.minimumWidth: dataView.width
00199             Layout.columnSpan: dataView.columns
00200             RowLayout {
00201               Rectangle {
00202                 Layout.minimumHeight: dataView.rowSpacing
00203                 Layout.maximumHeight: 3*dataView.rowSpacing
00204               }
00205               Loader {
00206                 id: sectionLoader
00207                 source: dataView.sectionDelegate
00208                 Binding {
00209                   target: sectionLoader.item
00210                   property: "value"
00211                   value: repeater.model.data(repeater.currentIndex, repeater.sectionIndex)
00212                 }
00213               }
00214             }
00215           }

```

```

00216      } */
00217      Rectangle {
00218          id: contentContainer
00219          Layout.fillHeight: true
00220          Layout.fillWidth: true
00221          Layout.maximumHeight: (dataView.rows > 0) ? (Math.floor(dataView.height / dataView.rows) -
00222              dataView.rowSpacing) : (Math.floor((dataView.height * dataView.columns) / repeater.count) - dataView.rowSpacing)
00223          Layout.maximumWidth: (dataView.columns > 0) ? (Math.floor(dataView.width / dataView.columns) -
00224              dataView.columnSpacing) : (Math.floor((dataView.width * dataView.rows) / repeater.count) - dataView.
00225              columnSpacing)
00226          Layout.row: computeRow()
00227          Layout.column: computeColumn()
00228          function computeRow() {
00229              var next = 0;
00230              if(index > 0){
00231                  var previousRow = repeater.itemAt(index-1).Layout.row;
00232                  var previousColumn = repeater.itemAt(index-1).Layout.column;
00233                  console.log("\ncompute row pour l'item d'id " + (index+1) + " successeur de [" +
00234                      previousRow + ", " + previousColumn+"]");
00235                  if(repeater.sectioning && (repeater.model.data(index-1, repeater.sectionIndex) !==
00236                      repeater.model.data(index, repeater.sectionIndex))) {
00237                      console.log("nouvelle section "+repeater.model.data(index, repeater.sectionIndex)+"
00238                          remplaçant "+repeater.model.data(index-1, repeater.sectionIndex));
00239                      if(dataView.columns > 0) {
00240                          console.log("layout horizontal de "+dataView.columns+" colonnes non complétées
00241                          par la section\nnon va à la ligne");
00242                          next=previousRow+1;
00243                      } else {
00244                          console.log("layout vertical de "+dataView.rows+" lignes\nnon retourne en
00245                          première ligne");
00246                          next = 1;
00247                      }
00248                  } else if(repeater.sectioning){
00249                      console.log("même section");
00250                      if(dataView.columns > 0) {
00251                          console.log("layout horizontal de "+dataView.columns+" colonnes");
00252                          if((previousColumn % dataView.columns) == 0) { /*dernière colonne*/
00253                              console.log("on passe à la ligne suivante");
00254                              next = previousRow + 1;
00255                          } else {
00256                              console.log("on ne change pas de ligne");
00257                              next = previousRow;
00258                          }
00259                      } else{
00260                          console.log("layout vertical de "+dataView.rows+" lignes\nnon passe à la ligne
00261                          suivante (ou on boucle)");
00262                          next = previousRow % dataView.rows + 1;
00263                      }
00264                  } else {
00265                      if(dataView.rows > 0) {
00266                          next = index % dataView.rows + 1;
00267                      } else {
00268                          next = Math.floor(index / dataView.columns) + 1;
00269                      }
00270                  }
00271              }
00272          }
00273          console.log("ligne "+next)
00274          return next;
00275      }
00276      function computeColumn() {
00277          var next = 0;
00278          if(index > 0){
00279              var previousRow = repeater.itemAt(index-1).Layout.row;
00280              var previousColumn = repeater.itemAt(index-1).Layout.column;
00281              console.log("\ncompute column pour l'item d'id " + (index+1) + " successeur de [" +
00282                  previousRow + ", " + previousColumn+"]");
00283              if(repeater.sectioning && (repeater.model.data(index-1, repeater.sectionIndex) !==
00284                  repeater.model.data(index, repeater.sectionIndex))) {
00285                  console.log("nouvelle section "+repeater.model.data(index, repeater.sectionIndex)+"
00286                      remplaçant "+repeater.model.data(index-1, repeater.sectionIndex));
00287                  if(dataView.columns > 0) {
00288                      console.log("layout horizontal de "+dataView.columns+" colonnes\nnon retourne à
00289                      la première colonne");
00290                      next = 1;
00291                  } else {
00292                      console.log("layout vertical de "+dataView.rows+" lignes non complétées par la
00293                      section\nnon va à la colonne suivante");
00294                      next = previousColumn+1;
00295                  }
00296              }
00297          } else if(repeater.sectioning){
00298              console.log("même section");
00299              if(dataView.rows > 0) {
00300                  console.log("layout vertical de "+dataView.rows+" lignes");
00301                  if((previousRow % dataView.rows) == 0) { /*dernière ligne*/
00302

```

```

00305             console.log("on passe à la colonne suivante");
00306             next = previousColumn + 1;
00307         } else {
00308             console.log("on ne change pas de colonne");
00309             next = previousColumn;
00310         }
00311     } else{
00312         console.log("layout horizontal de "+dataView.columns+" colonnes\nnon passe à la
00313         colonne suivante (ou on boucle) ");
00314         next = previousColumn % dataView.columns + 1;
00315     } else {
00316         if(dataView.columns > 0) {
00317             next= index % dataView.columns + 1;
00318         } else {
00319             next= Math.floor(index / dataView.rows) + 1;
00320         }
00321     }
00322 } else {
00323     next = 1;
00324 }
00325
00326     console.log("colonne "+next)
00327     return next;
00328 }
00329
00330 Loader {
00331     id:contentLoader
00332     source: dataView.SH_DataDelegate
00333     Binding {
00334         target: contentLoader.item
00335         property: "width"
00336         value: contentContainer.width - 1
00337     }
00338     Binding {
00339         target: contentLoader.item
00340         property: "height"
00341         value: contentContainer.height - 2
00342     }
00343     Connections {
00344         target: contentLoader.item
00345         onClicked: {
00346             if(dataView.enabled) {
00347                 dataView.selected(contentLoader.item.value);
00348             }
00349         }
00350     }
00351 }
00352 }
00353 /*} */
00354 }
00355 }

```

5.169 Référence du fichier views/qml/SH_DataDelegate.qml

Classes

- class [SH_DataDelegate](#)

5.170 SH_DataDelegate.qml

```

00001 import QtQuick 2.1
00002 import QtQuick.Window 2.1
00003 import QtQuick.Controls 1.0
00004 import QtQuick.Controls.Styles 1.0
00005 import QtQuick.Layouts 1.0
00006 import PreCheck 1.0
00007
00011 Button {
00012     id: btn
00013     property string value: ""
00014     property int fontSize: 8
00015     text: ""
00016     style: ButtonStyle {
00017         label: Text {
00018             id: styleText
00019             renderType: Text.NativeRendering
00020             verticalAlignment: Text.AlignVCenter

```

```

00021         horizontalAlignment: Text.AlignHCenter
00022         wrapMode: Text.Wrap
00023         text:btn.text
00024         font.pointSize: btn.fontSize
00025     }
00026     background:
00027     Item {
00028         anchors.margins: 0
00029         BorderImage {
00030             anchors.fill: parent
00031             border.top: 3
00032             border.bottom: 3
00033             border.left: 3
00034             border.right: 3
00035             anchors.bottomMargin: -1
00036             source: btn.pressed ? "../img/buttondown.png" : "../img/buttonup.png"
00037         }
00038     }
00039 }
00040 }
```

5.171 Référence du fichier views/qml/SH_HeaderView.qml

Classes

- class [SH_HeaderView](#)

5.172 SH_HeaderView.qml

```

00001 import QtQuick 2.1
00002 import QtQuick.Window 2.1
00003 import QtQuick.Controls 1.0
00004 import QtQuick.Controls.Styles 1.0
00005 import QtQuick.Layouts 1.0
00006 import PreCheck 1.0
00007
00011 GridLayout {
00012     id: filterGrid
00013     property var delegateModel
00014     property alias model: repeater.model
00015
00016     /*maxi 2 lignes s'il y a 4 colonnes; et maxi 3 lignes s'il y a 6 colonnes; sinon 10 toutes petites
00017     colonnes et autant de lignes que nécessaire*/
00018     /*maxNbRowsFew: 2
00019     maxNbColsFew: 4
00020     maxNbRowsStd: 3
00021     maxNbColsStd: 6
00022     maxNbFlowMany: 10*/
00023     columns: 4
00024     signal up(int index)
00025     signal checked(int index, bool ch)
00026     Repeater {
00027         id:repeater
00028         delegate:
00029             Item {
00030                 visible: filterGrid.visible
00031                 enabled: filterGrid.enabled
00032                 Layout.fillWidth: true
00033                 Layout.fillHeight: true
00034                 SH_TriStateCheckImage {
00035                     id: cbx
00036                     property var dataModel
00037                     enabled: filterGrid.enabled
00038                     anchors.fill: parent
00039                     Component.onCompleted: {
00040                         dataModel = Qt.binding(function() {return filterGrid.delegateModel.field(index);});
00041                         text = Qt.binding(function() {return dataModel.text;});
00042                     }
00043                     onUpChanged: {
00044                         if(cbx.up) {
00045                             dataModel.sortOrder = Qt.AscendingOrder;
00046                         } else {
00047                             dataModel.sortOrder = Qt.DescendingOrder
00048                         }
00049                     }
00050                     onCheckedChanged: {
```

```

00052             filterGrid.checked(index, !cbx.checked);
00053         }
00054         onPressed: {
00055             filterGrid.currentIndex = index
00056         }
00057     }
00058 }
00059
00060 }
00061 }
```

5.173 Référence du fichier views/qml/SH_Keyboard.qml

Classes

- class [SH_Keyboard](#)

5.174 SH_Keyboard.qml

```

00001 import QtQuick 2.1
00002 import QtQuick.Window 2.1
00003 import QtQuick.Controls 1.0
00004 import QtQuick.Controls.Styles 1.0
00005 import QtQuick.Layouts 1.0
00006 import PreCheck 1.0
00007
00011 GridLayout {
00012     id: layout
00013     rowSpacing:0
00014     columnSpacing:0
00015     property var actionsList: []
00016     Repeater {
00017         id: repeater
00018         model: (layout.actionsList==null) ? 0 : layout.actionsList.length
00019         delegate:
00020             SH_DataDelegate {
00021                 id: btn
00022                 enabled: layout.enabled
00023                 Layout.minimumHeight: (layout.rows > 0) ? (Math.floor(layout.height / layout.rows) - layout.
rowSpacing) : (Math.floor((layout.height * layout.columns) / repeater.count) - layout.rowSpacing)
00024                 Layout.minimumWidth: (layout.columns > 0) ? Math.floor(layout.width / layout.columns) - layout.
columnSpacing : Math.floor((layout.width * layout.rows) / repeater.count) - layout.columnSpacing
00025                 Layout.fillHeight: true
00026                 Layout.fillWidth: true
00027                 text: actionsList[index].text
00028                 tooltip: actionsList[index].tooltip
00029                 iconSource: actionsList[index].iconSource
00030                 checkable: false
00031                 action: Action {
00032                     id: act
00033                     text: actionsList[index].text
00034                     shortcut: actionsList[index].shortcut
00035                     checkable: false
00036                     enabled: actionsList[index].enabled
00037                     iconName: actionsList[index].iconName
00038                     iconSource: actionsList[index].iconSource
00039                     tooltip: actionsList[index].tooltip
00040                     Component.onCompleted: {
00041                         act.triggered.connect(actionsList[index].triggered);
00042                     }
00043                 }
00044             }
00045         }
00046 }
```

5.175 Référence du fichier views/qml/SH_OutputZone.qml

Classes

- class [SH_OutputZone](#)

5.176 SH_OutputZone.qml

```
00001 import QtQuick 2.1
00002 import QtQuick.Window 2.1
00003 import QtQuick.Controls 1.0
00004 import QtQuick.Controls.Styles 1.0
00005 import QtQuick.Layouts 1.0
00006 import PreCheck 1.0
00007
00011 Rectangle {
00012     id: output
00013     visible: true
00014     color: "lightgrey"
00015     property int lastVisibleRow: -1
00016
00017     signal display(string text)
00018     signal displayNewFixed(string text)
00019     signal displayNew(string text, bool editable)
00020     signal replace(string text)
00021     signal selected(string selectedItem)
00022
00023     onReplace: {
00024         if(output.lastVisibleRow >= 0) {
00025             if(rep.itemAt(output.lastVisibleRow).model === "") {
00026                 output.lastVisibleRow = Math.max(0, output.lastVisibleRow-1);
00027             }
00028
00029             layout.changeTextDisplay(output.lastVisibleRow, text, rep.itemAt(output.lastVisibleRow).editable
00030         );
00030         } else {
00031             display(text);
00032         }
00033     }
00034     onDisplay: layout.continueTextDisplay(output.lastVisibleRow, text);
00035     onDisplayNew: layout.changeTextDisplay(output.lastVisibleRow+1, text, editable);
00036     onDisplayNewFixed: layout.changeTextDisplay(output.lastVisibleRow+1, text, false);
00037
00046     function clear(row) {
00047         if(row <= output.lastVisibleRow) {
00048             rep.itemAt(row).model = "";
00049         }
00050     }
00059     function clearAll() {
00060         for(var currentRow = 0; currentRow<rep.count; currentRow++) {
00061             output.clear(currentRow);
00062         }
00063         output.lastVisibleRow = -1;
00064     }
00073     function displayText(text) {
00074         layout.changeTextDisplay(output.lastVisibleRow+1, text, false);
00075     }
00084     function displayCalendar() {
00085         /*TODO*/
00086     }
00095     function displaySqlDatas(sqlData, sqlDelegate) {
00096         rep.itemAt(output.lastVisibleRow).model = sqlData;
00097         rep.itemAt(output.lastVisibleRow).state="choices";
00098     }
00107     function displaySqlDetail(sqlData) {
00108         rep.itemAt(output.lastVisibleRow).model = sqlData;
00109         rep.itemAt(output.lastVisibleRow).state="detail";
00110         /*console.log(table+" "+row);*/
00111     }
00113     ColumnLayout {
00114         id:layout
00115         anchors.margins: 5
00116         anchors.fill: output
00117         height: output.height
00118         width: output.width
00119         function changeTextDisplay(row, text, editable) {
00120             if(row < rep.count) {
00121                 if(row > output.lastVisibleRow) {
00122                     output.lastVisibleRow=row;
00123                 }
00124                 rep.itemAt(row).editable = editable;
00125                 rep.itemAt(row).model = text;
00126             }
00127         }
00128         function continueTextDisplay(row, text) {
00129             console.log("continue display");
00130             console.log(row)
00131             console.log(text);
00132             if(output.lastVisibleRow<0) {
00133                 output.lastVisibleRow=0;
00134             }
00135         }
00136     }
00137 }
```

```

00151
00152     if(row < 0) {
00153         console.log("row > 0");
00154         layout.changeTextDisplay(row+1, text, false);
00155     } else if(row > output.lastVisibleRow) {
00156         console.log("row > "+output.lastVisibleRow);
00157         layout.changeTextDisplay(output.lastVisibleRow, text, false);
00158     } else {
00159         console.log("row <= 0");
00160         if(row >= rep.count) {
00161             row = rep.count-1;
00162         }
00163         rep.itemAt(row).model = rep.itemAt(row).model+text;
00164     }
00165 }
00166
00167 Repeater {
00168     id: rep
00169     model: 25
00170     delegate:
00171         RowLayout {
00172             id: row
00173             property var model: ""
00174             property bool editable: true
00175             Layout.minimumHeight: layout.height / rep.model - layout.spacing - layout.anchors.margins
00176             Layout.minimumWidth: layout.width - layout.anchors.margins
00177             Layout.fillWidth: true
00178             Layout.fillHeight: true
00179             visible: true
00180             state: "text"
00181             states: [
00182                 State{
00183                     name: "text"
00184                     PropertyChanges {
00185                         target: defaultContent
00186                         text: row.model
00187                         readOnly: !row.editable
00188                         visible: true
00189                     }
00190                     PropertyChanges {
00191                         target: choiceContent
00192                         model: 0
00193                         visible: false
00194                     }
00195                     /*PropertyChanges {
00196                         target: calendarContent
00197                         visible: false
00198                     }*/
00199                     PropertyChanges {
00200                         target: detailedContent
00201                         model: 0
00202                         visible: false
00203                     }
00204             },
00205             State{
00206                 name: "choices"
00207                 PropertyChanges {
00208                     target: defaultContent
00209                     visible: false
00210                 }
00211                 PropertyChanges {
00212                     target: choiceContent
00213                     model: row.model==""? 0 : row.model
00214                     visible: true
00215                 }
00216                 /*PropertyChanges {
00217                     target: calendarContent
00218                     visible: false
00219                 }*/
00220                 PropertyChanges {
00221                     target: detailedContent
00222                     model: 0
00223                     visible: false
00224             },
00225             State{
00226                 name: "choices"
00227                 PropertyChanges {
00228                     target: defaultContent
00229                     visible: false
00230                 }
00231                 PropertyChanges {
00232                     target: detailedContent
00233                     model: row.model==""? 0 : row.model
00234                     visible: true
00235                 }
00236             }
00237             /*PropertyChanges {

```

```

00238             target: calendarContent
00239             visible: false
00240         } */
00241         PropertyChanges {
00242             target: choiceContent
00243             model: 0
00244             visible: false
00245         }
00246     }
00247 ]
00248     Column {
00249         TextEdit {
00250             id: defaultContent
00251             readOnly: !row.editable
00252             wrapMode: TextEdit.WrapAtWordBoundaryOrAnywhere;
00253             Layout.fillHeight: true
00254             Layout.fillWidth: true
00255             font.pointSize: 12
00256             color: defaultContent.readOnly ? "grey" : "darkgrey"
00257         }
00258         SH_ContentView {
00259             id: choiceContent
00260             model: 0
00261             Layout.fillHeight: true
00262             Layout.fillWidth: true
00263         }
00264         SH_SqlTableView {
00265             id: detailedContent
00266             model: 0
00267             Layout.fillHeight: true
00268             Layout.fillWidth: true
00269         }
00270
00271         /*SH_CalendarDialog {
00272             id: calendarContent
00273             Layout.fillHeight: true
00274             Layout.fillWidth: true
00275         }*/
00276     }
00277 }
00278 }
00279 }
00280 }
```

5.177 Référence du fichier views/qml/SH_RoomsDelegate.qml

Classes

- class [SH_RoomsDelegate](#)

5.178 SH_RoomsDelegate.qml

```

00001 import QtQuick 2.1
00002 import QtQuick.Window 2.1
00003 import QtQuick.Controls 1.0
00004 import QtQuick.Controls.Styles 1.0
00005 import QtQuick.Layouts 1.0
00006 import PreCheck 1.0
00007
00011 SH_DataDelegate {
00012     id: btn
00013     text: qsTr("%L1").arg(NUMBER);
00014     value: NUMBER
00015     style: ButtonStyle {
00016         background:
00017             Item {
00018                 BorderImage {
00019                     anchors.fill: parent
00020                     border.top: 1
00021                     border.bottom: 1
00022                     border.left: 1
00023                     border.right: 1
00024                     anchors.bottomAnchor: -1
00025                     /*property color color:(AVAILABILITY == '') ? 'blue' : ((AVAILABILITY === '1' ||
00026                         AVAILABILITY === 1) ? "green" : "orange")*/
00026                     property string color:(AVAILABILITY == '') ? 'blue' : ((AVAILABILITY === '1' ||
00027                         AVAILABILITY === 1) ? "green" : "orange")
00027                     /*source: btn.pressed ? ".../img/button"+color.toString().toUpperCase()+"down.png" : ".../img
```

```

00028     source: btn.pressed ? "../img/button"+color.toUpperCase()+"down.png" : "../img/button"+
00029         color.toUpperCase()+"up.png"
00030     }
00031 }
00032 }

```

5.179 Référence du fichier views/qml/SH_RoomsSectionsDelegate.qml

Classes

- class [SH_RoomsSectionsDelegate](#)

5.180 SH_RoomsSectionsDelegate.qml

```

00001 import QtQuick 2.1
00002 import QtQuick.Window 2.1
00003 import QtQuick.Controls 1.0
00004 import QtQuick.Controls.Styles 1.0
00005 import QtQuick.Layouts 1.0
00006 import PreCheck 1.0
00007
00011 Rectangle {
00012     id: btn
00013     property string value
00014     color: "lightblue"
00015     Text {
00016         text: qsTr("Étage %L1").arg(btn.value);
00017         color: "darkblue"
00018         renderType: Text.NativeRendering
00019         verticalAlignment: Text.AlignVCenter
00020         horizontalAlignment: Text.AlignHCenter
00021         wrapMode: Text.Wrap
00022     }
00023 }

```

5.181 Référence du fichier views/qml/SH_ServicesDelegate.qml

Classes

- class [SH_ServicesDelegate](#)

5.182 SH_ServicesDelegate.qml

```

00001 import QtQuick 2.1
00002 import QtQuick.Window 2.1
00003 import QtQuick.Controls 1.0
00004 import QtQuick.Controls.Styles 1.0
00005 import QtQuick.Layouts 1.0
00006 import PreCheck 1.0
00007
00011 SH_DataDelegate {
00012     value: ID
00013     text: SERVICENAME
00014     /*visible: (ISAVAILABLE == 1) */
00015     /*enabled: (ROOMNEEDED == 1) */
00016 }

```

5.183 Référence du fichier views/qml/SH_SqlDataView.qml

Classes

- class [SH_SqlDataView](#)

5.184 SH_SqlDataView.qml

```

00001 import QtQuick 2.1
00002 import QtQuick.Window 2.1
00003 import QtQuick.Controls 1.0
00004 import QtQuick.Controls.Styles 1.0
00005 import QtQuick.Layouts 1.0
00006 import PreCheck 1.0
00007
00008 Rectangle {
00009     id:display
00010     /* le modèle*/
00011     property var sqlModel: []
00012     property string itemDelegate
00013     property string emptyDelegate : "SH_DataDelegate.qml"
00014     property string sectionDelegate : "SH_DataDelegate.qml"
00015     property bool filterIndicatorsVisibles: true
00016     property string filtersTitle
00017     signal selected(string selectedItem)
00018     signal newItem
00019     property bool isEmpty: true
00020     enabled: true
00021     height: childrenRect.height
00022     width: childrenRect.width
00023     anchors.margins: 3
00024     Text{
00025         text: qsTr("Aucun résultat")
00026         visible: display.isEmpty
00027         color: "#dd1f1f"
00028     }
00029     Component.onCompleted: {
00030         isEmpty = (display.sqlModel == []) || (!display.sqlModel.fetch());
00031     }
00032     ColumnLayout {
00033         id:colView
00034         spacing: 3
00035         visible: !display.isEmpty
00036         anchors.fill:parent
00037         /*Rectangle {
00038             id:upArea
00039             Layout.fillWidth: true
00040             Layout.minimumWidth: childrenRect.width
00041             Layout.minimumHeight: childrenRect.height
00042             visible: filterIndicatorsVisibles
00043             border.width: 1
00044             border.color: "black"
00045             ColumnLayout {
00046                 id: upCol
00047                 anchors.fill: upArea
00048                 Text {
00049                     id: upTitle
00050                     text: filtersTitle
00051                     Layout.fillWidth: true
00052                     Layout.preferredWidth: upTitle.paintedWidth
00053                     Layout.preferredHeight: upTitle.paintedHeight
00054                     Layout.alignment: Qt.AlignLeft
00055                 }
00056                 SH_HeaderView {
00057                     id: filterGrid
00058                     visible: upArea.visible
00059                     enabled: display.enabled
00060                     Layout.minimumWidth: childrenRect.width
00061                     Layout.minimumHeight: childrenRect.height
00062                     Layout.fillHeight: true
00063                     Layout.fillWidth: true
00064                     model: display.isEmpty? 0 : display.sqlModel.fieldsCount();
00065                     delegateModel: display.sqlModel;
00066                     Component.onCompleted: {
00067                         model = Qt.binding(function() {return display.isEmpty ? 0 :
00068                             display.sqlModel.fieldsCount();});
00069                         delegateModel = Qt.binding(function() {return display.sqlModel;});
00070                     }
00071                     onUp: dataView.sort(index);
00072                     onChecked: dataView.filter(index, ch);
00073                 }
00074             }
00075         }
00076         onUp: dataView.sort(index);
00077         onChecked: dataView.filter(index, ch);
00078     }
00079 }
00080 */
00081 SH_ContentView {
00082     id:dataView
00083     model: (display.isEmpty) ? 0 : display.sqlModel
00084     Layout.fillHeight: true
00085     Layout.fillWidth: true
00086     SH_DataDelegate: display.itemDelegate
00087     emptyDelegate: display.emptyDelegate
00088     sectionDelegate: display.sectionDelegate
00089     enabled: display.enabled

```

```

00090     Layout.minimumWidth: childrenRect.width
00091     Layout.minimumHeight: childrenRect.height
00092     Layout.maximumHeight: parent.height/*-upArea.height*/
00093     Layout.maximumWidth: parent.width
00094     onSelected: {
00095         display.selected(selectedItem);
00096     }
00097 }
00098 }
00099
00100 }

```

5.185 Référence du fichier views/qml/SH_SqlTableView.qml

Classes

- class [SH_SqlTableView](#)

5.186 SH_SqlTableView.qml

```

00001 import QtQuick 2.1
00002 import QtQuick.Window 2.1
00003 import QtQuick.Controls 1.0
00004 import QtQuick.Controls.Styles 1.0
00005 import QtQuick.Layouts 1.0
00006 import PreCheck 1.0
00007
00011 TableView {
00012     id: table
00013     model: 0
00014     signal selected(string selectedItem)
00015     Component.onCompleted:{
00016         /*var tmp = [];*/
00017         if(table.model !== 0) {
00018             table.model.fetch();
00019             var count = table.model.fieldsCount();
00020             for(var index = 0; index < count; index++) {
00021                 if(!table.model.containsFilterKeyColumn(index)) {
00022                     var col = Qt.createQmlObject('import QtQuick 2.1; import QtQuick.Controls 1.0;
00023             TableViewColumn {title: table.model.field('+index+').text; role: table.model.field('+index+').role}',table,'column'+index);
00024                     /*tmp.push(col);*/
00025                     table.addColumn(col);
00026                 }
00027             }
00028             /*return tmp;*/
00029         }
00030
00032     itemDelegate:
00033         Item {
00034
00035             Text {
00036                 width: parent.width
00037                 anchors.margins: 4
00038                 anchors.left: parent.left
00039                 anchors.verticalCenter: parent.verticalCenter
00040                 elide: styleData.elideMode
00041                 text: styleData.value !== undefined ? styleData.value : ""
00042                 color: styleData.textColor
00043                 visible: !styleData.selected
00044             }
00045             Loader /* Initialize text input lazily to improve performance*/
00046             id: loaderEditor
00047             anchors.fill: parent
00048             anchors.margins: 4
00049             Connections {
00050                 target: loaderEditor.item
00051                 onAccepted: {
00052                     /*if (typeof styleData.value === 'number')*/
00053                     table.selected(styleData.row);
00054                     model.setData(styleData.row, styleData.role, styleData.value);
00055                 }
00056             }
00057             sourceComponent: styleData.selected ? editor : null
00058             Component {
00059                 id: editor

```

```

00060         TextInput {
00061             id: textinput
00062             color: styleData.textColor
00063             text: styleData.value
00064             MouseArea {
00065                 id: mouseArea
00066                 anchors.fill: parent
00067                 hoverEnabled: true
00068                 onClicked: {
00069                     textinput.forceActiveFocus()
00070                 }
00071             }
00072             onFocusChanged: {
00073                 if(styleData.value !== textinput.text) {
00074                     textinput.accepted();
00075                 }
00076             }
00077         }
00078     }
00079 }
00080 }
00081 }
00082 }
```

5.187 Référence du fichier views/qml/SH_TabZone.qml

Classes

- class [SH_TabZone](#)

5.188 SH_TabZone.qml

```

00001 import QtQuick 2.1
00002 import QtQuick.Window 2.1
00003 import QtQuick.Controls 1.0
00004 import QtQuick.Controls.Styles 1.0
00005 import QtQuick.Layouts 1.0
00006 import PreCheck 1.0
00007
00011 TabView {
00012     id: tabView
00013     currentIndex: 0
00014     signal selected(string selectedItem)
00015     signal selectedForDetail(var data)
00016     property var stdKeyboard: []
00017     signal reload()
00018     signal newBilling()
00019     signal newSelling()
00020     signal newBooking()
00029     function openTab(tabIndex) {
00030         tabView.currentIndex = tabIndex;
00031     }
00032     onNewBilling: {
00033         console.log("launch new billings thread ?");
00034         SH_App.launchBillingsThread();
00035     }
00036     onNewBooking: {
00037         SH_App.launchBookingsThread();
00038     }
00039     onNewSelling: {
00040         SH_App.launchBillThread();
00041     }
00042
00043     style: TabViewStyle {
00045         id:style
00046         frame: Rectangle {
00047             anchors.fill: parent
00048             height: tabView.height-style.frameOverlap
00049             width: tabView.width
00050             color: "#dcdcdc"
00051             border.color: "#aaa"
00052
00053             Rectangle {
00054                 anchors.fill: parent
00055                 color: "transparent"
00056                 border.color: "#66ffff"
00057                 anchors.margins: 2
00058             }
00059         }
00060     }
00061 }
```

```

00058         }
00059     }
00060   }
00061
00062   Component.onCompleted: {
00063     tabView.addTab(qsTr("Clavier complet"), fullKeyboardTab);
00064     tabView.reload();
00065   }
00066   onReload: {
00067     var nbTabs = tabView.count;
00068     for(var i = 1; i < nbTabs; i++) {
00069       tabView.removeTab(i)
00070     }
00071     switch(SH_App.currentMode) {
00072       case SH_AppMode.RECEPTION:
00073         tabView.addTab(qsTr("Prestations"), servicesTab);
00074         tabView.addTab(qsTr("Chambres"), roomsTab);
00075         tabView.addTab(qsTr("Facturations"), billingsTab);
00076         tabView.addTab(qsTr("Réservations"), bookingsTab);
00077         /*tabView.addTab(qsTr("Offres"), offersTab);*/
00078         break;
00079       case SH_AppMode.MANAGEMENT_Z:
00080         /*tabView.addTab(qsTr("Clôtures"), accountsTab);*/
00081       case SH_AppMode.MANAGEMENT_X:
00082         /*tabView.addTab(qsTr("Prestations"), servicesEditTab);*/
00083         tabView.addTab(qsTr("Chambres"), roomsEditTab);
00084         /*tabView.addTab(qsTr("Clients privés"), clientsEditTab);*/
00085         /*tabView.addTab(qsTr("Groupes"), groupsEditTab);*/
00086         /*tabView.addTab(qsTr("Rapports"), reportsTab);*/
00087         break;
00088       case SH_AppMode.ADMINISTRATION :
00089         /*tabView.addTab(qsTr("Paramètres"), settingsTab);*/
00090         /*tabView.addTab(qsTr("Utilisateurs"), usersTab);*/
00091         /*tabView.addTab(qsTr("Design"), skinTab);*/
00092         break;
00093     }
00094   }
00095
00096   resources: [
00097     Component {
00098       id:fullKeyboardTab
00099       SH_Keyboard {
00100         id:fullKeyboard
00101         actionsList: stdKeyboard
00102         columns: 11
00103         enabled: tabView.enabled
00104         height: tabView.height-style.frameOverlap
00105         width: tabView.width
00106       }
00107     },
00108     Component {
00109       id: servicesTab
00110       ColumnLayout {
00111         Rectangle {
00112           color:"transparent"
00113           Layout.fillWidth: true
00114           Layout.minimumHeight: 20
00115           RowLayout {
00116             SH_DataDelegate {
00117               Layout.minimumHeight: childrenRect.height
00118               Layout.minimumWidth: childrenRect.width
00119               text: qsTr("Autre");
00120               value: "0"
00121               onClicked: {
00122                 tabView.selected(value);
00123               }
00124             }
00125             SH_DataDelegate {
00126               Layout.minimumHeight: childrenRect.height
00127               Layout.minimumWidth: childrenRect.width
00128               text: qsTr("Autre..."); 
00129               value: "-1"
00130               onClicked: {
00131                 tabView.selected(value);
00132               }
00133             }
00134           }
00135         }
00136       SH_SqlDataView {
00137         Layout.fillWidth: true
00138         Layout.fillHeight: true
00139         enabled: tabView.enabled
00140         filtersTitle: qsTr("Tri des prestations")
00141         sqlModel: SH_ServicesModel { }
00142         itemDelegate: "SH_ServicesDelegate.qml"
00143         emptyDelegate: "SH_DataDelegate.qml"
00144         sectionDelegate: "SH_DataDelegate.qml"

```

```

00145             onSelected: {
00146                 tabView.selected(selectedItem);
00147             }
00148         }
00149     },
00150     Component {
00151         id: roomsTab
00152         SH_SqlDataView {
00153             filtersTitle: qsTr("Tri des chambres")
00154             sqlModel: SH_RoomsModel { }
00155             itemDelegate: "SH_RoomsDelegate.qml"
00156             emptyDelegate: "SH_DataDelegate.qml"
00157             sectionDelegate: "RoomsSectionDelegate.qml"
00158             onSelected: {
00159                 tabView.selected(selectedItem);
00160             }
00161         }
00162     },
00163 },
00164 Component {
00165     id: roomsEditTab
00166     SH_SqlTableView {
00167         id: roomsEditView
00168         model: SH_RoomsModel { }
00169         onSelected: {
00170             tabView.selectedForDetail(roomsEditView.model.table, selectedItem);
00171         }
00172     }
00173 },
00174 Component {
00175     id: billingsTab
00176     SH_SqlDataView {
00177         sqlModel: SH_BillingsModel {}
00178         filtersTitle: qsTr("Tri des facturations")
00179         itemDelegate: "SH_BillingsDelegate.qml"
00180         emptyDelegate: "SH_DataDelegate.qml"
00181         sectionDelegate: "SH_DataDelegate.qml"
00182         onSelected: {
00183             tabView.selected(selectedItem);
00184         }
00185         onNewItem: {
00186             tabView.newBilling();
00187         }
00188     }
00189 },
00190 Component {
00191     id: bookingsTab
00192     SH_SqlDataView {
00193         sqlModel: SH_BookingsModel
00194         filtersTitle: qsTr("Tri des réservations")
00195         itemDelegate: "SH_BookingsDelegate.qml"
00196         emptyDelegate: "SH_DataDelegate.qml"
00197         sectionDelegate: "SH_DataDelegate.qml"
00198         onSelected: {
00199             tabView.selected(selectedItem);
00200         }
00201         onNewItem: {
00202             tabView.newBooking();
00203         }
00204     }
00205 }
00206 ]
00207 }
00208 }
```

5.189 Référence du fichier views/qml/SH_TriStateCheckImage.qml

Classes

- class [SH_TriStateCheckImage](#)

5.190 SH_TriStateCheckImage.qml

```

00001 import QtQuick 2.1
00002 import QtQuick.Window 2.1
00003 import QtQuick.Controls 1.0
00004 import QtQuick.Controls.Styles 1.0
00005 import QtQuick.Layouts 1.0
```

```

00006 import PreCheck 1.0
00007
00011 FocusScope {
00012     id:root
00013
00014     signal pressed
00015
00016     property bool checked
00017     property bool up
00018     property real labelSpacing: 0
00019
00020     property alias text: txtPart.text
00021
00022     readonly property string imgUp: "../img/up.png"
00023     readonly property string imgDown: "../img/down.png"
00024     readonly property real length: txtPart.paintedWidth+imgPart.paintedWidth
00025     readonly property real innerHeight: root.height-root.anchors.topMargin-root.anchors.bottomMargin
00026     readonly property real innerWidth: root.width-root.anchors.rightMargin-root.anchors.leftMargin
00027
00028     Image {
00029         id: imgPart
00030         fillMode: Image.PreserveAspectFit
00031         width: root.innerHeight
00032         height: root.innerHeight
00033         source: root.up ? root.imgUp : root.imgDown
00034         visible: root.checked
00035         anchors.left: root.left
00036         anchors.top: root.top
00037         anchors.bottom: root.bottom
00038
00039         /*TouchArea */
00040         MouseArea {
00041             onClicked: {
00042                 root.pressed();
00043                 root.up = !root.up;
00044             }
00045             focus: imgPart.visible
00046             anchors.fill: imgPart
00047             hoverEnabled: true
00048         }
00049     }
00050
00051     Text {
00052         id:txtPart
00053         visible: true
00054         height: root.innerHeight
00055         width: imgPart.visible ? root.innerWidth-imgPart.paintedWidth-root.labelSpacing : root.innerWidth
00056         anchors.left: imgPart.visible ? imgPart.right : root.left
00057         anchors.right: root.right
00058         anchors.top: root.top
00059         anchors.bottom: root.bottom
00060         font.strikeout: !root.checked
00061         wrapMode: Text.Wrap
00062         /*TouchArea */
00063         MouseArea {
00064             onClicked: {
00065                 root.pressed();
00066                 root.checked = !root.checked;
00067             }
00068             focus: !imgPart.visible
00069             anchors.fill: txtPart
00070             hoverEnabled: true
00071         }
00072     }
00073 }

```

5.191 Référence du fichier views/qml/SH_WelcomePage.qml

Classes

- class [SH_WelcomePage](#)

5.192 SH_WelcomePage.qml

```

00001 import QtQuick 2.1
00002 import QtQuick.Window 2.1
00003 import QtQuick.Controls 1.0
00004 import QtQuick.ControlsStyles 1.0

```

```
00005 import QtQuick.Layouts 1.0
00006 import PreCheck 1.0
00007
00011 Item {
00012     id: welcomePage
00013     signal logOut()
00014     signal loggedOut()
00015     signal quit()
00016     signal clicked()
00017     signal reload()
00018     onLogOut: {
00019         if(SH_App.userLogOut()) {
00020             welcomePage.loggedOut();
00021         }
00022     }
00023     onReload: {
00024         buttons.anchors.centerIn = welcomePage;
00025         receptionButton.visible = SH_App.currentUser.receptionist;
00026         administrationButton.visible = SH_App.currentUser.administrator;
00027         managementXButton.visible = SH_App.currentUser.managerX;
00028         managementZButton.visible = SH_App.currentUser.managerZ;
00029     }
00030
00031     ColumnLayout {
00032         id: buttons
00033         anchors.centerIn: welcomePage
00034         width: parent.width / 2
00035         height: parent.height / 2
00036         spacing: 5
00037         RowLayout {
00038             Button {
00039                 id: logoutButton
00040                 height: (buttons.height%2)-buttons.spacing
00041                 width: (buttons.width/2)-buttons.spacing
00042                 text: qsTr("Déconnecter")
00043                 onClicked: {
00044                     welcomePage.logOut();
00045                 }
00046             }
00047             Button {
00048                 id: quitButton
00049                 height: (buttons.height%2)-buttons.spacing
00050                 width: (buttons.width/2)-buttons.spacing
00051                 text: qsTr("Quitter")
00052                 onClicked: {
00053                     welcomePage.quit();
00054                 }
00055             }
00056         }
00057         GridLayout {
00058             id: grid
00059             columns: 2
00060             property int count: SH_App.currentUser.roles
00061             Rectangle {
00062                 color: "transparent"
00063                 Layout.minimumHeight: childrenRect.height
00064                 Layout.minimumWidth: childrenRect.width
00065                 Layout.columnSpan: (grid.count%grid.columns == 0 || Layout.row <= grid.count/grid.columns)
? 1 : grid.columns-Layout.column
00066             Button {
00067                 id: receptionButton
00068                 anchors.centerIn: parent
00069                 height: (buttons.height/2)-buttons.spacing
00070                 width: (buttons.width/2)-buttons.spacing
00071                 text: qsTr("Réception")
00072                 visible: SH_App.currentUser.receptionist
00073                 onClicked: {
00074                     SH_App.currentMode = SH_AppMode.RECEPTION;
00075                     welcomePage.clicked();
00076                 }
00077             }
00078         }
00079         Rectangle {
00080             color: "transparent"
00081             Layout.minimumHeight: childrenRect.height
00082             Layout.minimumWidth: childrenRect.width
00083             Layout.columnSpan: (grid.count%grid.columns == 0 || Layout.row <= grid.count/grid.columns)
? 1 : grid.columns-Layout.column
00084             Button {
00085                 id: administrationButton
00086                 height: (buttons.height/2)-buttons.spacing
00087                 width: (buttons.width/2)-buttons.spacing
00088                 text: qsTr("Administration")
00089                 visible: SH_App.currentUser.administrator
00090                 onClicked: {
00091                     SH_App.currentMode = SH_AppMode.ADMINISTRATION;
00092                     welcomePage.clicked();
00093                 }
00094             }
00095         }
00096     }
00097 }
```

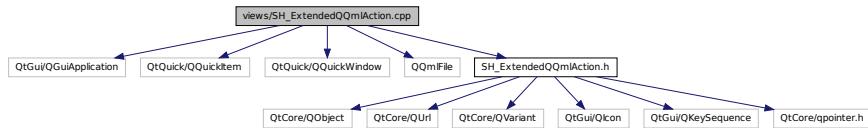
```

00093             }
00094         }
00095     }
00096     Rectangle {
00097         color: "transparent"
00098         Layout.minimumHeight: childrenRect.height
00099         Layout.minimumWidth: childrenRect.width
00100         Layout.columnSpan: (grid.count%grid.columns == 0 || Layout.row <= grid.count/grid.columns)
? 1 : grid.columns-Layout.column
00101     Button {
00102         id: managementXButton
00103         height: (buttons.height/2)-buttons.spacing
00104         width: (buttons.width/2)-buttons.spacing
00105         text: qsTr("Gestion lecture")
00106         visible: SH_App.currentUser.managerX
00107         onClicked: {
00108             SH_App.currentMode = SH_AppMode.MANAGEMENT_X;
00109             welcomePage.clicked();
00110         }
00111     }
00112 }
00113 Rectangle {
00114     color: "transparent"
00115     Layout.minimumHeight: childrenRect.height
00116     Layout.minimumWidth: childrenRect.width
00117     Layout.columnSpan: (grid.count%grid.columns == 0 || Layout.row <= grid.count/grid.columns)
? 1 : grid.columns-Layout.column
00118     Button {
00119         id: managementZButton
00120         height: (buttons.height/2)-buttons.spacing
00121         width: (buttons.width/2)-buttons.spacing
00122         text: qsTr("Gestion lecture écriture")
00123         visible: SH_App.currentUser.managerZ
00124         onClicked: {
00125             SH_App.currentMode = SH_AppMode.MANAGEMENT_Z;
00126             welcomePage.clicked();
00127         }
00128     }
00129 }
00130 }
00131 }
00132 }

```

5.193 Référence du fichier views/SH_ExtendedQQmlAction.cpp

```
#include <QtGui/QGuiApplication>
#include <QtQuick/QQuickItem>
#include <QtQuick/QQuickWindow>
#include <QQmlFile>
#include "SH_ExtendedQQmlAction.h"
Graphe des dépendances par inclusion de SH_ExtendedQQmlAction.cpp :
```



Fonctions

- bool `qShortcutContextMatcher (QObject *o, Qt::ShortcutContext context)`

5.193.1 Documentation des fonctions

5.193.1.1 bool `qShortcutContextMatcher (QObject * o, Qt::ShortcutContext context)`

Définition à la ligne 50 du fichier `SH_ExtendedQQmlAction.cpp`.

```

00051 {
00052     switch (context) {
00053     case Qt::ApplicationShortcut:
00054         return true;
00055     case Qt::WindowShortcut: {
00056         QObject *w = o;
00057         while (w && !w->isWindowType()) {
00058             w = w->parent();
00059             if (QQuickItem * item = qobject_cast<QQuickItem*>(w))
00060                 w = item->window();
00061         }
00062         if (w && w == QGuiApplication::focusWindow())
00063             return true;
00064     }
00065     case Qt::WidgetShortcut:
00066     case Qt::WidgetWithChildrenShortcut:
00067         break;
00068     }
00069
00070     return false;
00071 }
```

5.194 SH_ExtendedQQmlAction.cpp

```

00001 #include <QtGui/QGuiApplication>
00002 #include <QtQuick/QQuickItem>
00003 #include <QtQuick/QQuickWindow>
00004 #include <QQmlFile>
00005 #include "SH_ExtendedQQmlAction.h"
00006 /*#include "../src/gui/kernel/qguiapplication_p.h"*/
00007
00014 SH_ExtendedQQmlAction::SH_ExtendedQQmlAction(
    QObject *parent)
00015     : QObject(parent)
00016     , m_enabled(true)
00017 {
00018 }
00019
00025 SH_ExtendedQQmlAction::~SH_ExtendedQQmlAction()
00026 {
00027     setKeyShortcut(Qt::Key_unknown);
00028     setMnemonicFromText(QString());
00029 }
00030
00036 void SH_ExtendedQQmlAction::setText(const QString &text)
00037 {
00038     if (text == m_text)
00039         return;
00040     m_text = text;
00041     setMnemonicFromText(m_text);
00042     emit textChanged();
00043 }
00044
00050 bool qShortcutContextMatcher(QObject *o, Qt::ShortcutContext context)
00051 {
00052     switch (context) {
00053     case Qt::ApplicationShortcut:
00054         return true;
00055     case Qt::WindowShortcut: {
00056         QObject *w = o;
00057         while (w && !w->isWindowType()) {
00058             w = w->parent();
00059             if (QQuickItem * item = qobject_cast<QQuickItem*>(w))
00060                 w = item->window();
00061         }
00062         if (w && w == QGuiApplication::focusWindow())
00063             return true;
00064     }
00065     case Qt::WidgetShortcut:
00066     case Qt::WidgetWithChildrenShortcut:
00067         break;
00068     }
00069
00070     return false;
00071 }
00072
00078 void SH_ExtendedQQmlAction::setKeySequence(const QKeySequence &
sequence) {
00079     if (sequence == m_shortcut)
00080         return;
00081
00082     /*if (!m_shortcut.isEmpty())
00083         QGuiApplicationPrivate::instance()->shortcutMap.removeShortcut(0, this, m_shortcut);
```

```

00084     */
00085     m_shortcut = sequence;
00086
00087     if (!m_shortcut.isEmpty()) {
00088         Qt::ShortcutContext context = Qt::WindowShortcut;
00089         /*QGuiApplicationPrivate::instance()->shortcutMap.addShortcut(this, m_shortcut, context,
00090             qShortcutContextMatcher);*/
00091     }
00092     emit shortcutChanged(shortcut());
00093 }
00094
00095 void SH_ExtendedQQmlAction::setKeyShortcut(const Qt::Key &shortcut)
00096 {
00097     setKeySequence(QKeySequence(shortcut));
00098 }
00099
00100 QString SH_ExtendedQQmlAction::shortcut() const
00101 {
00102     return m_shortcut.toString(QKeySequence::NativeText);
00103 }
00104
00105 void SH_ExtendedQQmlAction::setShortcut(const QString &arg)
00106 {
00107     if(shortcut() == arg)
00108         return;
00109
00110     setKeySequence(QKeySequence::fromString(arg));
00111 }
00112
00113 void SH_ExtendedQQmlAction::setMnemonicFromText(const QString &
00114     text)
00115 {
00116     QKeySequence sequence = QKeySequence::mnemonic(text);
00117     if (m_mnemonic == sequence)
00118         return;
00119
00120     /*if (!m_mnemonic.isEmpty())
00121         QGuiApplicationPrivate::instance()->shortcutMap.removeShortcut(0, this, m_mnemonic);
00122     */
00123     m_mnemonic = sequence;
00124
00125     if (!m_mnemonic.isEmpty()) {
00126         Qt::ShortcutContext context = Qt::WindowShortcut;
00127         /*QGuiApplicationPrivate::instance()->shortcutMap.addShortcut(this, m_mnemonic, context,
00128             qShortcutContextMatcher);*/
00129     }
00130 }
00131
00132 void SH_ExtendedQQmlAction::setIconSource(const QUrl &iconSource)
00133 {
00134     if (iconSource == m_iconSource)
00135         return;
00136
00137     m_iconSource = iconSource;
00138     if (m iconName.isEmpty() || m_icon.isNull()) {
00139         QString urlString = QQmlFile::urlToLocalFileOrQrc(iconSource);
00140         m_icon = QIcon(urlString);
00141
00142         emit iconChanged();
00143     }
00144     emit iconSourceChanged();
00145 }
00146
00147 }
00148
00149 QString SH_ExtendedQQmlAction::iconName() const
00150 {
00151     return m_iconName;
00152 }
00153
00154 void SH_ExtendedQQmlAction::setIconName(const QString &iconName)
00155 {
00156     if (iconName == m_iconName)
00157         return;
00158     m_iconName = iconName;
00159     m_icon = QIcon::fromTheme(m_iconName, QIcon(QQmlFile::urlToLocalFileOrQrc(
00160         m_iconSource)));
00161     emit iconNameChanged();
00162     emit iconChanged();
00163 }
00164
00165 }
00166
00167 }
00168
00169 void SH_ExtendedQQmlAction::setTooltip(const QString &arg)
00170 {
00171     if (m_tooltip != arg) {
00172         m_tooltip = arg;
00173         emit tooltipChanged(arg);
00174     }
00175 }
00176
00177 }
00178
00179
00180
00181
00182
00183
00184
00185
00186
00187
00188
00189
00190
00191
00192
00193
00194
00195
00196
00197
00198
00199
00200
00201
00202
00203
00204
00205
00206

```

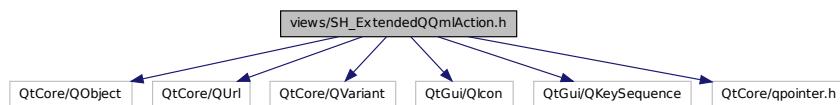
```

00212 void SH_ExtendedQQmlAction::setEnabled(bool e)
00213 {
00214     if (e == m_enabled)
00215         return;
00216     m_enabled = e;
00217     emit enabledChanged();
00218 }
00219
00220
00221
00222 bool SH_ExtendedQQmlAction::event(QEvent *e)
00223 {
00224     if (!m_enabled)
00225         return false;
00226
00227     if (e->type() != QEvent::Shortcut)
00228         return false;
00229
00230     QShortcutEvent *se = static_cast<QShortcutEvent *>(e);
00231
00232     Q_ASSERT_X(se->key() == m_shortcut || se->key() == m_mnemonic,
00233                 "QQQuickAction::event",
00234                 "Received shortcut event from incorrect shortcut");
00235
00236     if (se->isAmbiguous())
00237         qWarning("QQQuickAction::event: Ambiguous shortcut overload: %s", se->key().toString(
00238             QKeySequence::NativeText).toLatin1().constData());
00239
00240     return false;
00241 }
00242
00243     trigger();
00244
00245
00246     return true;
00247 }
00248
00249 }
00250
00251 void SH_ExtendedQQmlAction::trigger()
00252 {
00253     if (!m_enabled)
00254         return;
00255     emit triggered();
00256 }
00257 }
```

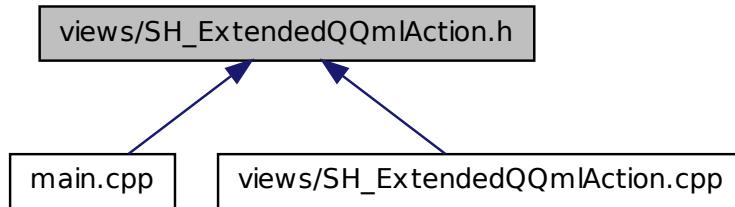
5.195 Référence du fichier views/S_H_ExtendedQQmlAction.h

```
#include <QtCore/QObject>
#include <QtCore/QUrl>
#include <QtCore/QVariant>
#include <QtGui/QIcon>
#include <QtGui/QKeySequence>
#include <QtCore/qpointer.h>
```

Graphe des dépendances par inclusion de SH_ExtendedQQmlAction.h :



Ce graphe montre quels fichiers incluent directement ou indirectement ce fichier :



Classes

- class [SH_ExtendedQQmlAction](#)

5.196 SH_ExtendedQQmlAction.h

```

00001 #ifndef QQQQuickAction_H
00002 #define QQQQuickAction_H
00003
00004 #include <QtCore/QObject>
00005 #include <QtCore/QUrl>
00006 #include <QtCore/QVariant>
00007 #include <QtGui/QIcon>
00008 #include <QtGui/QKeySequence>
00009 #include <QtCore/qpointer.h>
00010
00011 class SH_ExtendedQQmlAction : public QObject
00012 {
00013     Q_OBJECT
00014
00015     Q_PROPERTY(QString text READ text WRITE setText NOTIFY
00016                 textChanged)
00017     Q_PROPERTY(QUrl iconSource READ iconSource WRITE
00018                 setIconSource NOTIFY iconSourceChanged)
00019     Q_PROPERTY(QString iconName READ iconName WRITE setIconName NOTIFY
00020                 iconNameChanged)
00021     Q_PROPERTY(QVariant __icon READ iconVariant NOTIFY
00022                 iconChanged)
00023     Q_PROPERTY(QString tooltip READ tooltip WRITE setTooltip NOTIFY
00024                 tooltipChanged)
00025     Q_PROPERTY(bool enabled READ isEnabled WRITE setEnabled NOTIFY
00026                 enabledChanged)
00027 #ifndef QT_NO_SHORTCUT
00028     Q_PROPERTY(QString shortcut READ shortcut WRITE setShortcut NOTIFY
00029                 shortcutChanged)
00030     Q_PROPERTY(Qt::Key keyShortcut WRITE setKeyShortcut NOTIFY
00031                 keyShortcutChanged)
00032 #endif
00033
00034 public:
00035     explicit SH_ExtendedQQmlAction(QObject *parent = 0);
00036     ~SH_ExtendedQQmlAction();
00037
00038     QString text() const { return m_text; }
00039     void setText(const QString &text);
00040
00041     QString shortcut() const;
00042     void setShortcut(const QString &shortcut);
00043
00044     QKeySequence keyShortcut() const { return m_shortcut; }
00045     void setKeyShortcut(const Qt::Key &shortcut);
00046
00047     void setMnemonicFromText(const QString &mnemonic);
00048
00049     QString iconName() const;
  
```

```
00113     void setIconName(const QString &iconName);
00114
00121     QUrl iconSource() const { return m_iconSource; }
00128     void setIconSource(const QUrl &iconSource);
00129
00136     QString tooltip() const { return m_tooltip; }
00143     void setTooltip(const QString &tooltip);
00144
00151     bool isEnabled() const { return m_enabled; }
00158     void setEnabled(bool e);
00159
00166     QIcon icon() const { return m_icon; }
00173     QVariant iconVariant() const { return QVariant(m_icon); }
00180     void setIcon(QIcon icon) { m_icon = icon; emit
00181         iconChanged(); }
00181
00189     bool event(QEvent *e);
00190
00191
00192     public Q_SLOTS:
00198     void trigger();
00199
00200     Q_SIGNALS:
00206     void triggered();
00213     void toggled(bool checked);
00214
00220     void textChanged();
00227     void shortcutChanged(QString shortcut);
00234     void keyShortcutChanged(Qt::Key keyShortcut);
00235
00241     void iconChanged();
00247     void iconNameChanged();
00253     void iconSourceChanged();
00260     void tooltipChanged(QString arg);
00266     void enabledChanged();
00267
00268     protected:
00275     void setKeySequence(const QKeySequence &sequence);
00276
00277     private:
00281     QString m_text;
00285     QUrl m_iconSource;
00289     QString m_iconName;
00293     QIcon m_icon;
00297     bool m_enabled;
00301     QKeySequence m_shortcut;
00305     QKeySequence m_mnemonic;
00309     QString m_tooltip;
00310 };
00311
00312 #endif /* QQQuickAction_H */
```

Index

~SH_DatabaseManager
 SH_DatabaseManager, 200

~SH_ExtendedQQmlAction
 SH_ExtendedQQmlAction, 258

__icon
 SH_ExtendedQQmlAction, 270

_instance
 SH_DatabaseManager, 207

ACCUEIL
 SH_ApplicationCore, 49

ADMINISTRATION
 SH_ApplicationCore, 49

actionsList
 SH_Keyboard, 336

activeFilterIndicatorIndexes
 SH_ContentView, 182

addChildrenNextTransition
 SH_AddressCreationStateMachine, 30
 SH_BillingCreationStateMachine, 65
 SH_ClientCreationStateMachine, 136
 SH_InOutStateMachine, 319
 Sh_LoopingInOutStateMachine, 339
 SH_ServiceCharging, 441

addFilterIndex
 SH_ContentView, 182

addFilterKeyColumn
 SH_BillingsTableModel, 85
 SH_BillsTableModel, 100
 SH_BookingsTableModel, 117
 SH_ClientsTableModel, 154
 SH_ExtendedProxyModel, 241
 SH_GroupsTableModel, 292
 SH_RoomsTableModel, 425
 SH_ServicesTableModel, 465

addIOState
 SH_AddressCreationStateMachine, 31
 SH_BillingCreationStateMachine, 66
 SH_ClientCreationStateMachine, 137
 SH_InOutStateMachine, 320
 Sh_LoopingInOutStateMachine, 341
 SH_ServiceCharging, 443

addIOStateMachine
 SH_AddressCreationStateMachine, 33
 SH_BillingCreationStateMachine, 68
 SH_ClientCreationStateMachine, 139
 SH_InOutStateMachine, 322
 Sh_LoopingInOutStateMachine, 343
 SH_ServiceCharging, 445

administrator

SH_Trainee, 542

SH_User, 559

answerInvalid
 SH_DatabaseContentQuestionState, 186
 SH_DateQuestionState, 212
 SH.DecimalQuestionState, 226
 SH_NumericQuestionState, 369
 SH_QuestionState, 391
 SH_RegExpQuestionState, 404
 SH_StringQuestionState, 520

answerValid
 SH_DatabaseContentQuestionState, 186
 SH_DateQuestionState, 212
 SH.DecimalQuestionState, 226
 SH_NumericQuestionState, 369
 SH_QuestionState, 392
 SH_RegExpQuestionState, 405
 SH_StringQuestionState, 520

AppMode
 SH_ApplicationCore, 49

ApplicationWindow, 9

applyRoles
 SH_SqlDataModel, 486

balanceLogRoutine
 SH_ApplicationCore, 50

booleanSet
 SH_ExtendedProxyModel, 252

Button, 10

CONNEXION
 SH_ApplicationCore, 49

cancelProcess
 SH_CommonPage, 167

cancelReplace
 SH_CommonPage, 167

cancelReplacement
 SH_AddressCreationStateMachine, 34
 SH_ApplicationCore, 50
 SH_BillingCreationStateMachine, 69
 SH_ClientCreationStateMachine, 140
 SH_InOutStateMachine, 322
 Sh_LoopingInOutStateMachine, 344
 SH_ServiceCharging, 446

cancelRunningThread
 SH_ApplicationCore, 50

checkUsername
 SH_ConexionPage, 180

checkValidity
 SH_DatabaseContentQuestionState, 187

SH_DateQuestionState, 212
SH.DecimalQuestionState, 226
SH.NumericQuestionState, 369
SH.QuestionState, 392
SH.RegExpQuestionState, 405
SH.StringQuestionState, 520
checked
 SH_HeaderView, 305
 SH_TriStateCheckImage, 546
choiceList
 SH_DatabaseContentQuestionState, 187
clear
 SH_OutputZone, 382
clearAll
 SH_AddressCreationStateMachine, 34
 SH_ApplicationCore, 51
 SH_BillingCreationStateMachine, 69
 SH_ClientCreationStateMachine, 140
 SH_InOutStateMachine, 323
 Sh_LoopingInOutStateMachine, 344
 SH_OutputZone, 383
 SH_ServiceCharging, 446
clicked
 SH_CalendarDialog, 131
 SH_WelcomePage, 571
closed
 SH_CalendarDialog, 131
confirm
 SH_CommonPage, 167
confirmInput
 SH_AddressCreationStateMachine, 35
 SH_BillingCreationStateMachine, 70
 SH_ClientCreationStateMachine, 141
 SH_ConfirmationState, 171
 SH_InOutStateMachine, 323
 Sh_LoopingInOutStateMachine, 345
 SH_ServiceCharging, 446
 SH_ValidationState, 562
connectRunningThread
 SH_ApplicationCore, 51
containsFilterKeyColumn
 SH_BillingsTableModel, 85
 SH_BillsTableModel, 100
 SH_BookingsTableModel, 117
 SH_ClientsTableModel, 154
 SH_ExtendedProxyModel, 241
 SH_GroupsTableModel, 292
 SH_RoomsTableModel, 425
 SH_ServicesTableModel, 465
current
 Sh_LoopingInOutStateMachine, 345
 SH_ServiceCharging, 447
currentDate
 SH_CalendarDialog, 131
currentDay
 SH_CalendarDialog, 131
currentFSMchanged
 SH_ApplicationCore, 52
currentMode
 SH_ApplicationCore, 59
currentMonth
 SH_CalendarDialog, 131
currentMonthLength
 SH_CalendarDialog, 131
currentSortKeyColumn
 SH_BillingsTableModel, 85
 SH_BillsTableModel, 100
 SH_BookingsTableModel, 117
 SH_ClientsTableModel, 154
 SH_ExtendedProxyModel, 241
 SH_GroupsTableModel, 292
 SH_RoomsTableModel, 425
 SH_ServicesTableModel, 465
currentUser
 SH_ApplicationCore, 59
currentWeekday
 SH_CalendarDialog, 131
currentYear
 SH_CalendarDialog, 131
DEBUG
 SH_MessageManager, 360
DEBUG_VERBOSE
 SH_MessageManager, 360
data
 SH_BillingsTableModel, 85, 86
 SH_BillsTableModel, 100, 101
 SH_BookingsTableModel, 117, 118
 SH_ClientsTableModel, 154, 155
 SH_ExtendedProxyModel, 241, 242
 SH_GroupsTableModel, 292, 293
 SH_RoomsTableModel, 425, 426
 SH_ServicesTableModel, 465, 466
 SH_SqlDataModel, 487
dataCount
 SH_DatabaseManager, 200
datas
 SH_SqlDataModel, 488
dbAliasNameStr
 SH_DatabaseManager.h, 692
dbCannotOpenStr
 SH_DatabaseManager.h, 692
dbConnect
 SH_DatabaseManager, 201
dbConnection
 SH_DatabaseManager, 207
dbDisconnect
 SH_DatabaseManager, 202
dbDriverNotExistStr
 SH_DatabaseManager.h, 692
dbDriverStr
 SH_DatabaseManager.h, 692
dbFileNameStr
 SH_DatabaseManager.h, 692
dbFilePathStr
 SH_DatabaseManager.h, 692
dbFolderPathStr

SH_DatabaseManager.h, 692
 dbPasswordStr
 SH_DatabaseManager.h, 692
 dbUsernameStr
 SH_DatabaseManager.h, 693
 delegateModel
 SH_HeaderView, 305
 display
 SH_ConfirmationState, 172
 SH_DatabaseContentQuestionState, 188
 SH_DateQuestionState, 213
 SH.DecimalQuestionState, 227
 SH_FileSelectionState, 273
 SH_InOutState, 308
 SH_NumericQuestionState, 370
 SH_OutputZone, 383
 SH_QuestionState, 393
 SH_RegExpQuestionState, 406
 SH_StatementState, 509
 SH_StringQuestionState, 521
 SH_ValidationState, 563
 displayCalendar
 SH_AddressCreationStateMachine, 35
 SH_ApplicationCore, 52
 SH_BillingCreationStateMachine, 70
 SH_ClientCreationStateMachine, 141
 SH_InOutStateMachine, 324
 Sh_LoopingInOutStateMachine, 346
 SH_OutputZone, 383
 SH_ServiceCharging, 447
 displayChoiceList
 SH_DatabaseContentQuestionState, 188
 displayFileDialog
 SH_AddressCreationStateMachine, 36
 SH_BillingCreationStateMachine, 71
 SH_ClientCreationStateMachine, 142
 SH_InOutStateMachine, 324
 Sh_LoopingInOutStateMachine, 346
 SH_ServiceCharging, 448
 displayNew
 SH_OutputZone, 383
 displayNewFixed
 SH_OutputZone, 383
 displaySqlDatas
 SH_OutputZone, 383
 displaySqlDetail
 SH_OutputZone, 383
 displayText
 SH_OutputZone, 383
 divideQVariantMap
 SH_DatabaseManager, 202
 ERROR
 SH_MessageManager, 360
 empty
 SH_BillingsTableModel, 96
 SH_BillsTableModel, 111
 SH_BookingsTableModel, 128
 SH_ClientsTableModel, 165
 SH_ExtendedProxyModel, 254
 SH_GroupsTableModel, 303
 SH_RoomsTableModel, 436
 SH_ServicesTableModel, 476
 emptyDelegate
 SH_ContentView, 182
 SH_SqlDataView, 504
 enableLogging
 main.cpp, 644
 enabled
 SH_ExtendedQQmlAction, 270
 enabledChanged
 SH_ExtendedQQmlAction, 258
 errorMessage
 SH_MessageManager, 360
 ErrorMode
 SH_MessageManager, 360
 event
 SH_ExtendedQQmlAction, 258
 execInsertReturningQuery
 SH_DatabaseManager, 203
 execReplaceQuery
 SH_DatabaseManager, 204
 execSelectQuery
 SH_DatabaseManager, 205
 exists
 SH_Trainee, 536
 SH_User, 550
 exportlog
 main.cpp, 644
 fetch
 SH_BillingsTableModel, 87
 SH_BillsTableModel, 102
 SH_BookingsTableModel, 119
 SH_ClientsTableModel, 156
 SH_ExtendedProxyModel, 243
 SH_GroupsTableModel, 294
 SH_RoomsTableModel, 427
 SH_ServicesTableModel, 467
 SH_SqlDataModel, 489
 field
 SH_BillingsTableModel, 87
 SH_BillsTableModel, 102
 SH_BookingsTableModel, 119
 SH_ClientsTableModel, 156
 SH_ExtendedProxyModel, 243
 SH_GroupsTableModel, 294
 SH_RoomsTableModel, 427
 SH_ServicesTableModel, 467
 SH_SqlDataModel, 490
 fieldFromRole
 SH_SqlDataModel, 491
 fields
 SH_BillingsTableModel, 88
 SH_BillsTableModel, 103
 SH_BookingsTableModel, 120
 SH_ClientsTableModel, 157
 SH_ExtendedProxyModel, 244

SH_GroupsTableModel, 295
SH_RoomsTableModel, 428
SH_ServicesTableModel, 468
fieldsChanged
 SH_SQLDataModel, 491
fieldsCount
 SH_BillingsTableModel, 88
 SH_BillsTableModel, 103
 SH_BookingsTableModel, 120
 SH_ClientsTableModel, 157
 SH_ExtendedProxyModel, 244
 SH_GroupsTableModel, 295
 SH_RoomsTableModel, 428
 SH_ServicesTableModel, 468
 SH_SQLDataModel, 492
fieldsList
 SH_BillingsTableModel, 96
 SH_BillsTableModel, 111
 SH_BookingsTableModel, 128
 SH_ClientsTableModel, 165
 SH_ExtendedProxyModel, 254
 SH_GroupsTableModel, 303
 SH_RoomsTableModel, 436
 SH_ServicesTableModel, 476
 SH_SQLDataModel, 492
fillModel
 SH_BillingsTableModel, 88
 SH_BillsTableModel, 103
 SH_BookingsTableModel, 120
 SH_ClientsTableModel, 157
 SH_ExtendedProxyModel, 244
 SH_GroupsTableModel, 295
 SH_RoomsTableModel, 428
 SH_ServicesTableModel, 468
filter
 SH_ContentView, 182
 SH_SQLDataModel, 493, 501
filterAcceptsRow
 SH_BillingsTableModel, 89
 SH_BillsTableModel, 104
 SH_BookingsTableModel, 121
 SH_ClientsTableModel, 158
 SH_ExtendedProxyModel, 245
 SH_GroupsTableModel, 296
 SH_RoomsTableModel, 429
 SH_ServicesTableModel, 469
filterChanged
 SH_SQLDataModel, 493
filterIndicatorsVisibles
 SH_SQLTableView, 504
filters
 SH_ExtendedProxyModel, 253
filtersTitle
 SH_SQLTableView, 504
firstWeekdayIndex
 SH_CalendarDialog, 131
flags
 SH_BillingsTableModel, 89
SH_BillsTableModel, 104
SH_BookingsTableModel, 121
SH_ClientsTableModel, 158
SH_ExtendedProxyModel, 245
SH_GroupsTableModel, 296
SH_RoomsTableModel, 430
SH_ServicesTableModel, 469
FocusScope, 11
fontSize
 SH_BillingsDelegate, 81
 SH_BookingsDelegate, 113
 SH_DataDelegate, 208
 SH_RoomsDelegate, 419
 SH_ServicesDelegate, 461
getContentPane
 SH_AddressCreationStateMachine, 36
 SH_BillingCreationStateMachine, 71
 SH_ClientCreationStateMachine, 142
 SH_InOutStateMachine, 325
 Sh_LoopingInOutStateMachine, 347
 SH_ServiceCharging, 448
getDbConnection
 SH_DatabaseManager, 205
getFuture
 SH_DateQuestionState, 213
getInstance
 SH_DatabaseManager, 205
getPast
 SH_DateQuestionState, 214
givenAnswer
 SH_DatabaseContentQuestionState, 188
 SH_DateQuestionState, 214
 SH.DecimalQuestionState, 227
 SH_NumericQuestionState, 370
 SH_QuestionState, 393
 SH_RegExpQuestionState, 406
 SH_StringQuestionState, 521
GridLayout, 12
historyValue
 SH_AddressCreationStateMachine, 37
 SH_BillingCreationStateMachine, 72
 SH_ClientCreationStateMachine, 143
 SH_InOutStateMachine, 325
 Sh_LoopingInOutStateMachine, 347
 SH_ServiceCharging, 449
icon
 SH_ExtendedQQmlAction, 259
iconChanged
 SH_ExtendedQQmlAction, 259
iconName
 SH_ExtendedQQmlAction, 260, 270
iconNameChanged
 SH_ExtendedQQmlAction, 260
iconSource
 SH_ExtendedQQmlAction, 260, 270
iconSourceChanged

SH_ExtendedQQmlAction, 261
 iconVariant
 SH_ExtendedQQmlAction, 261
 id
 SH_Trainee, 537, 542
 SH_User, 551, 559
 imgDown
 SH_TriStateCheckImage, 546
 imgUp
 SH_TriStateCheckImage, 546
 infoMessage
 SH_MessageManager, 361
 init
 SH_ApplicationCore, 52
 innerHeight
 SH_TriStateCheckImage, 546
 innerWidth
 SH_TriStateCheckImage, 546
 input
 SH_ConfirmationState, 172
 SH_DatabaseContentQuestionState, 189
 SH_DateQuestionState, 214
 SH.DecimalQuestionState, 228
 SH_FileSelectionState, 274
 SH_InOutState, 309
 SH_NumericQuestionState, 371
 SH_QuestionState, 393
 SH_RegExpQuestionState, 407
 SH_StatementState, 510
 SH_StringQuestionState, 522
 SH_ValidationState, 563
 insertUpdate
 SH_AdaptDatabaseState, 24
 invalidateFilter
 SH_BillingsTableModel, 90
 SH_BillsTableModel, 104
 SH_BookingsTableModel, 122
 SH_ClientsTableModel, 158
 SH_ExtendedProxyModel, 246
 SH_GroupsTableModel, 296
 SH_RoomsTableModel, 430
 SH_ServicesTableModel, 470
 ioContent
 SH_AddressCreationStateMachine, 37
 SH_BillingCreationStateMachine, 72
 SH_ClientCreationStateMachine, 143
 SH_InOutStateMachine, 326
 Sh_LoopingInOutStateMachine, 348
 SH_ServiceCharging, 449
 ioStatesHistory
 SH_AddressCreationStateMachine, 38
 SH_BillingCreationStateMachine, 73
 SH_ClientCreationStateMachine, 144
 SH_InOutStateMachine, 326
 Sh_LoopingInOutStateMachine, 348
 SH_ServiceCharging, 450
 isAdministrator
 SH_Trainee, 537
 SH_User, 551
 isAnswerValid
 SH_DatabaseContentQuestionState, 189
 SH_DateQuestionState, 214
 SH.DecimalQuestionState, 228
 SH_NumericQuestionState, 371
 SH_QuestionState, 394
 SH_RegExpQuestionState, 407
 SH_StringQuestionState, 522
 isConnected
 SH_DatabaseManager, 206
 isEmpty
 SH_BillingsTableModel, 90
 SH_BillsTableModel, 105
 SH_BookingsTableModel, 122
 SH_ClientsTableModel, 159
 SH_ExtendedProxyModel, 246
 SH_GroupsTableModel, 297
 SH_RoomsTableModel, 430
 SH_ServicesTableModel, 470
 SH_SqlDataModel, 493
 SH_SqlTableView, 504
 isEnabled
 SH_ExtendedQQmlAction, 261
 isManagerX
 SH_Trainee, 538
 SH_User, 552
 isManagerZ
 SH_Trainee, 538
 SH_User, 552
 isReceptionist
 SH_Trainee, 538
 SH_User, 553
 isValid
 SH_Trainee, 539
 SH_User, 553
 item, 13
 itemDelegate
 SH_SqlTableView, 504
 iterations
 main.cpp, 647
 keySelected
 SH_CommonPage, 167
 keyShortcut
 SH_ExtendedQQmlAction, 261, 270
 keyShortcutChanged
 SH_ExtendedQQmlAction, 261
 labelSpacing
 SH_TriStateCheckImage, 546
 lastError
 SH_BillingsTableModel, 90, 96
 SH_BillsTableModel, 105, 111
 SH_BookingsTableModel, 122, 128
 SH_ClientsTableModel, 159, 165
 SH_ExtendedProxyModel, 246, 254
 SH_GroupsTableModel, 297, 303
 SH_RoomsTableModel, 430, 436

SH_ServicesTableModel, 470, 476
SH_SQLDataModel, 494, 501
lastErrorChanged
 SH_SQLDataModel, 494
lastVisibleRow
 SH_OutputZone, 383
launchBillThread
 SH_ApplicationCore, 53
launchBillingsThread
 SH_ApplicationCore, 53
launchBookingsThread
 SH_ApplicationCore, 54
length
 SH_TriStateCheckImage, 546
limit
 Sh_LoopingInOutStateMachine, 349, 358
 SH_ServiceCharging, 450, 459
limitChanged
 Sh_LoopingInOutStateMachine, 349
 SH_ServiceCharging, 450
logIn
 SH_ConexionPage, 180
 SH_Trainee, 539
 SH_User, 554
logOut
 SH_WelcomePage, 571
loggedIn
 SH_ConexionPage, 180
loggedOut
 SH_WelcomePage, 571
logic/SH_AdaptDatabaseState.cpp, 575
logic/SH_AdaptDatabaseState.h, 576
logic/SH_AddressCreation.cpp, 577
logic/SH_AddressCreation.h, 577, 578
logic/SH_BillingCreation.cpp, 579
logic/SH_BillingCreation.h, 581, 582
logic/SH_ClientCreation.cpp, 583
logic/SH_ClientCreation.h, 584, 585
logic/SH_ConfirmationState.cpp, 585, 586
logic/SH_ConfirmationState.h, 587, 588
logic/SH_DatabaseContentQuestionState.cpp, 588
logic/SH_DatabaseContentQuestionState.h, 589, 590
logic/SH_DateQuestionState.cpp, 591, 592
logic/SH_DateQuestionState.h, 593, 594
logic/SH.DecimalQuestionState.cpp, 595, 596
logic/SH.DecimalQuestionState.h, 597, 599
logic/SH_FileSelectionState.cpp, 599, 600
logic/SH_FileSelectionState.h, 600, 601
logic/SH_GenericDebugableState.cpp, 602
logic/SH_GenericDebugableState.h, 603, 604
logic/SH_IOSTate.cpp, 604, 605
logic/SH_IOSTate.h, 606, 607
logic/SH_IOSTateMachine.cpp, 607, 608
logic/SH_IOSTateMachine.h, 611, 612
logic/SH_LoopingIOStateMachine.cpp, 613
logic/SH_LoopingIOStateMachine.h, 615, 616
logic/SH_NamedObject.cpp, 616, 617
logic/SH_NamedObject.h, 617, 618
logic/SH_NumericQuestionState.cpp, 618, 619
logic/SH_NumericQuestionState.h, 620, 622
logic/SH_PrintingState.cpp, 622, 623
logic/SH_PrintingState.h, 623, 624
logic/SH_QuestionState.cpp, 625
logic/SH_QuestionState.h, 626, 627
logic/SH_RegExpQuestionState.cpp, 628, 629
logic/SH_RegExpQuestionState.h, 630, 631
logic/SH_ServiceCharging.cpp, 631, 632
logic/SH_ServiceCharging.h, 633, 634
logic/SH_StatementState.cpp, 635
logic/SH_StatementState.h, 636, 637
logic/SH_StringQuestionState.cpp, 637, 638
logic/SH_StringQuestionState.h, 639, 640
logic/SH_ValidationState.cpp, 640, 641
logic/SH_ValidationState.h, 642, 643
MANAGEMENT_X
 SH_ApplicationCore, 49
MANAGEMENT_Z
 SH_ApplicationCore, 49
m_administrator
 SH_User, 558
m_choices
 SH_DatabaseContentQuestionState, 196
m_choicesDisplayed
 SH_DatabaseContentQuestionState, 196
m_condition
 SH_DatabaseContentQuestionState, 196
m_contents
 Sh_LoopingInOutStateMachine, 357
m_current
 Sh_LoopingInOutStateMachine, 357
m_currentFSM
 SH_ApplicationCore, 59
m_currentUser
 SH_ApplicationCore, 59
m_display
 SH_InOutState, 315
m_enabled
 SH_ExtendedQQmlAction, 268
m_field
 SH_DatabaseContentQuestionState, 196
m_future
 SH_DateQuestionState, 222
m_givenAnswer
 SH_QuestionState, 401
m_icon
 SH_ExtendedQQmlAction, 269
m_iconName
 SH_ExtendedQQmlAction, 269
m_iconSource
 SH_ExtendedQQmlAction, 269
m_id
 SH_User, 558
m_input
 SH_InOutState, 315
m_ioContent
 SH_AddressCreationStateMachine, 44

SH_BillingCreationStateMachine, 79
 SH_ClientCreationStateMachine, 150
 SH_InOutStateMachine, 334
 Sh_LoopingInOutStateMachine, 357
 SH_ServiceCharging, 458
 m_ioStatesHistory
 SH_AddressCreationStateMachine, 44
 SH_BillingCreationStateMachine, 79
 SH_ClientCreationStateMachine, 150
 SH_InOutStateMachine, 334
 Sh_LoopingInOutStateMachine, 357
 SH_ServiceCharging, 458
 m_isVisible
 SH_InOutState, 316
 m_limit
 Sh_LoopingInOutStateMachine, 357
 m_managerX
 SH_User, 558
 m_managerZ
 SH_User, 558
 m_max
 SH.DecimalQuestionState, 237
 SH.NumericQuestionState, 380
 m_maxLen
 SH.StringQuestionState, 531
 m_min
 SH.DecimalQuestionState, 237
 SH.NumericQuestionState, 380
 m_minLen
 SH.StringQuestionState, 531
 m_mnemonic
 SH.ExtendedQQmlAction, 269
 m_mode
 SH.ApplicationCore, 59
 m_name
 SH.NamedObject, 365
 SH.SqlDataFields, 483
 SH.User, 558
 m_output
 SH.InOutState, 316
 m_past
 SH.DateQuestionState, 222
 m_persistentContent
 Sh_LoopingInOutStateMachine, 358
 m_priceMin
 SH.ServiceCharging, 459
 m_ptraddress
 SH.NamedObject, 365
 m_receptionist
 SH.User, 559
 m_regexp
 SH.RegExpQuestionState, 417
 m_shortcut
 SH.ExtendedQQmlAction, 269
 m_sortOrder
 SH.SqlDataFields, 483
 m_table
 SH.DatabaseContentQuestionState, 196

m_tableName
 SH.AddressCreationStateMachine, 44
 SH.BillingCreationStateMachine, 79
 SH.ClientCreationStateMachine, 150
 SH.InOutStateMachine, 334
 Sh.LoopingInOutStateMachine, 358
 SH.ServiceCharging, 459
 m_text
 SH.ExtendedQQmlAction, 269
 SH.SqlDataFields, 483
 m_tooltip
 SH.ExtendedQQmlAction, 269
 m_vat
 SH.ServiceCharging, 459
 mDataFields
 SH.SqlDataModel, 500
 mFilter
 SH.SqlDataModel, 500
 mRecords
 SH.SqlDataModel, 500
 mRoles
 SH.SqlDataModel, 500
 mSort
 SH.SqlDataModel, 500
 mSqlQuery
 SH.SqlDataModel, 501
 mTable
 SH.SqlDataModel, 501
 main
 main.cpp, 645
 main.cpp, 643
 enableLogging, 644
 exportlog, 644
 iterations, 647
 main, 645
 spin, 646
 statusChanged, 646
 managerX
 SH.Trainee, 542
 SH.User, 559
 managerZ
 SH.Trainee, 542
 SH.User, 559
 max
 SH.DecimalQuestionState, 228
 SH.NumericQuestionState, 371
 maxLen
 SH.RegExpQuestionState, 407
 SH.StringQuestionState, 522
 min
 SH.DecimalQuestionState, 229
 SH.NumericQuestionState, 372
 minLen
 SH.RegExpQuestionState, 408
 SH.StringQuestionState, 523
 mode
 SH.ApplicationCore, 54
 modeChanged

SH_ApplicationCore, 55
model
 SH_BillingsTableModel, 96
 SH_BillsTableModel, 110
 SH_BookingsTableModel, 128
 SH_ClientsTableModel, 164
 SH_ContentView, 182
 SH_ExtendedProxyModel, 253
 SH_GroupsTableModel, 302
 SH_HeaderView, 305
 SH_RoomsTableModel, 436
 SH_ServicesTableModel, 475
models/SH_BillingsTableModel.cpp, 649
models/SH_BillingsTableModel.h, 650, 651
models/SH_BillsTableModel.cpp, 651
models/SH_BillsTableModel.h, 652
models/SH_BookingsTableModel.cpp, 653
models/SH_BookingsTableModel.h, 654, 655
models/SH_ClientsTableModel.cpp, 655
models/SH_ClientsTableModel.h, 656
models/SH_Company.cpp, 657
models/SH_Company.h, 657, 658
models/SH_ExtendedProxyModel.cpp, 658, 659
models/SH_ExtendedSqlProxyModel.h, 661, 662
models/SH_GroupsTableModel.cpp, 663, 664
models/SH_GroupsTableModel.h, 664, 665
models/SH_RoomsTableModel.cpp, 666
models/SH_RoomsTableModel.h, 667
models/SH_ServicesTableModel.cpp, 668
models/SH_ServicesTableModel.h, 669
models/SH_SqlDataField.cpp, 670
models/SH_SqlDataField.h, 671
models/SH_SqlDataModel.cpp, 672
models/SH_SqlDataModel.h, 676
models/SH_Trainee.cpp, 677, 678
models/SH_Trainee.h, 678, 679
models/SH_User.cpp, 680
models/SH_User.h, 681, 682
monthsList
 SH_CalendarDialog, 131
name
 SH_GenericState, 283
 SH_InOutStateMachine, 327
 SH_NamedObject, 364
 SH_SqlDataFields, 479, 483
 SH_Trainee, 540, 543
 SH_User, 555, 559
nameChanged
 SH_SqlDataFields, 479
 SH_Trainee, 541
 SH_User, 555
newBilling
 SH_TabZone, 533
newBooking
 SH_TabZone, 533
newItem
 SH_SqlDataView, 503
newSelling

SH_TabZone, 533
next
 SH_AdaptDatabaseState, 25
 SH_AddressCreationStateMachine, 38
 SH_BillingCreationStateMachine, 73
 SH_ClientCreationStateMachine, 144
 SH_ConfirmationState, 173
 SH_DatabaseContentQuestionState, 189
 SH_DateQuestionState, 215
 SH.DecimalQuestionState, 229
 SH_FileSelectionState, 274
 SH_GenericState, 284
 SH_InOutState, 309
 SH_InOutStateMachine, 327
 Sh_LoopingInOutStateMachine, 349
 SH_NumericQuestionState, 372
 SH_PrintingState, 386
 SH_QuestionState, 394
 SH_RegExpQuestionState, 408
 SH_ServiceCharging, 451
 SH_StatementState, 510
 SH_StringQuestionState, 523
 SH_ValidationState, 564
notNullSet
 SH_ExtendedProxyModel, 253
nullSet
 SH_ExtendedProxyModel, 253
onEntry
 SH_AdaptDatabaseState, 25
 SH_ConfirmationState, 173
 SH_DatabaseContentQuestionState, 190
 SH_DateQuestionState, 215
 SH.DecimalQuestionState, 230
 SH_FileSelectionState, 275
 SH_GenericState, 284
 SH_InOutState, 310
 SH_NumericQuestionState, 373
 SH_PrintingState, 386
 SH_QuestionState, 394
 SH_RegExpQuestionState, 409
 SH_StatementState, 511
 SH_StringQuestionState, 524
 SH_ValidationState, 564
onExit
 SH_AdaptDatabaseState, 26
 SH_ConfirmationState, 173
 SH_DatabaseContentQuestionState, 190
 SH_DateQuestionState, 216
 SH.DecimalQuestionState, 230
 SH_FileSelectionState, 275
 SH_GenericState, 285
 SH_InOutState, 310
 SH_NumericQuestionState, 373
 SH_PrintingState, 387
 SH_QuestionState, 395
 SH_RegExpQuestionState, 409
 SH_StatementState, 511
 SH_StringQuestionState, 524

SH_ValidationState, 564
 onMachineStarted
 SH_GenericState, 285
 onTransitionTriggered
 SH_GenericState, 286
 openTab
 SH_ApplicationCore, 55
 SH_TabZone, 533
 opened
 SH_CalendarDialog, 131
 output
 SH_ConfirmationState, 174
 SH_DatabaseContentQuestionState, 191
 SH_DateQuestionState, 216
 SH.DecimalQuestionState, 231
 SH_FileSelectionState, 275
 SH_InOutState, 310
 SH_NumericQuestionState, 374
 SH_QuestionState, 395
 SH_RegExpQuestionState, 410
 SH_StatementState, 512
 SH_StringQuestionState, 525
 SH_ValidationState, 565

 pages
 SH_app, 46
 passwordSet
 SH_ExtendedProxyModel, 253
 pressed
 SH_TriStateCheckImage, 545
 printFinished
 SH_PrintingState, 388
 printStarted
 SH_PrintingState, 388
 ptraddress
 SH_GenericState, 287
 SH_InOutStateMachine, 328
 SH_NamedObject, 364

 QAbstractListModel, 14
 QObject, 15
 QQuickItem, 17
 qShortcutContextMatcher
 SH_ExtendedQQmlAction.cpp, 730
 QSortFilterProxyModel, 18
 QState, 19
 QStateMachine, 20
 query
 SH_SqlDataModel, 494
 quit
 SH_CommonPage, 167
 SH_WelcomePage, 571

 RECEPTION
 SH_ApplicationCore, 49
 RELEASE
 SH_MessageManager, 360
 rawInput
 SH_ConfirmationState, 174

 SH_DatabaseContentQuestionState, 191
 SH_DateQuestionState, 217
 SH.DecimalQuestionState, 231
 SH_FileSelectionState, 276
 SH_InOutState, 311
 SH_NumericQuestionState, 374
 SH_QuestionState, 396
 SH_RegExpQuestionState, 410
 SH_StatementState, 512
 SH_StringQuestionState, 525
 SH_ValidationState, 565
 readonlySet
 SH_ExtendedProxyModel, 253
 receiveConfirmation
 SH_ApplicationCore, 55
 receiveInput
 SH_AddressCreationStateMachine, 39
 SH_ApplicationCore, 55
 SH_BillingCreationStateMachine, 74
 SH_ClientCreationStateMachine, 145
 SH_InOutStateMachine, 328
 Sh_LoopingInOutStateMachine, 350
 SH_ServiceCharging, 451
 receiveValidation
 SH_ApplicationCore, 55
 receptionist
 SH_Trainee, 543
 SH_User, 559
 Rectangle, 21
 refresh
 SH_CalendarDialog, 131
 regexp
 SH_RegExpQuestionState, 411
 reload
 SH_app, 46
 SH_CommonPage, 167
 SH_ConexionPage, 180
 SH_TabZone, 533
 SH_WelcomePage, 571
 removeFilterIndex
 SH_ContentView, 182
 removeFilterKeyColumn
 SH_BillingsTableModel, 90
 SH_BillsTableModel, 105
 SH_BookingsTableModel, 122
 SH_ClientsTableModel, 159
 SH_ExtendedProxyModel, 246
 SH_GroupsTableModel, 297
 SH_RoomsTableModel, 431
 SH_ServicesTableModel, 470
 replace
 SH_CommonPage, 167
 SH_OutputZone, 383
 replaceInput
 SH_AddressCreationStateMachine, 39
 SH_ApplicationCore, 55
 SH_BillingCreationStateMachine, 74
 SH_ClientCreationStateMachine, 145

SH_InOutStateMachine, 328
Sh_LoopingInOutStateMachine, 350
SH_ServiceCharging, 451
replaceSet
 SH_ExtendedProxyModel, 246
resendInput
 SH_ConfirmationState, 175
 SH_DatabaseContentQuestionState, 192
 SH_DateQuestionState, 217
 SH.DecimalQuestionState, 232
 SH_FileSelectionState, 277
 SH_InOutState, 312
 SH_NumericQuestionState, 375
 SH_QuestionState, 397
 SH_RegExpQuestionState, 411
 SH_StatementState, 513
 SH_StringQuestionState, 526
 SH_ValidationState, 566
resendText
 SH_AddressCreationStateMachine, 39
 SH_ApplicationCore, 56
 SH_BillingCreationStateMachine, 74
 SH_ClientCreationStateMachine, 145
 SH_InOutStateMachine, 329
 Sh_LoopingInOutStateMachine, 351
 SH_ServiceCharging, 452
resetFieldsToAll
 SH_SqlDataModel, 494
resetFilterCondition
 SH_SqlDataModel, 495
role
 SH_SqlDataFields, 479, 483
roleChanged
 SH_SqlDataFields, 480
roleForField
 SH_SqlDataModel, 495
roleNames
 SH_BillingsTableModel, 91
 SH_BillsTableModel, 105
 SH_BookingsTableModel, 123
 SH_ClientsTableModel, 159
 SH_ExtendedProxyModel, 247
 SH_GroupsTableModel, 297
 SH_RoomsTableModel, 431
 SH_ServicesTableModel, 470
 SH_SqlDataModel, 495
roles
 SH_Trainee, 541, 543
 SH_User, 555, 559
rolesChanged
 SH_SqlDataModel, 495
 SH_Trainee, 541
 SH_User, 555
rowCount
 SH_SqlDataModel, 496
SH_ApplicationCore
 ACCUEIL, 49
 ADMINISTRATION, 49
 CONNEXION, 49
 MANAGEMENT_X, 49
 MANAGEMENT_Z, 49
 RECEPTION, 49
SH_MessageManager
 DEBUG, 360
 DEBUG_VERBOSE, 360
 ERROR, 360
 RELEASE, 360
 TEST, 360
SH_AdaptDatabaseState, 21
 insertUpdate, 24
 next, 25
 onEntry, 25
 onExit, 26
 SH_AdaptDatabaseState, 24
 SH_AdaptDatabaseState, 24
 toString, 26
SH_AddressCreationStateMachine, 27
 addChildrenNextTransition, 30
 addIOState, 31
 addIOStateMachine, 33
 cancelReplacement, 34
 clearAll, 34
 confirmInput, 35
 displayCalendar, 35
 displayFileDialog, 36
 getContentValue, 36
 historyValue, 37
 ioContent, 37
 ioStatesHistory, 38
 m_ioContent, 44
 m_ioStatesHistory, 44
 m_tableName, 44
 next, 38
 receiveInput, 39
 replaceInput, 39
 resendText, 39
 SH_AddressCreationStateMachine, 30
 sendText, 40
 setContentValue, 40
 setIOStateHistory, 41
 setIOcontent, 41
 setTableName, 42
 SH_AddressCreationStateMachine, 30
 tableName, 42
 toString, 43
 validateInput, 43
SH_ApplicationCore, 46
 AppMode, 49
 balanceLogRoutine, 50
 cancelReplacement, 50
 cancelRunningThread, 50
 clearAll, 51
 connectRunningThread, 51
 currentFSMchanged, 52
 currentMode, 59
 currentUser, 59

displayCalendar, 52
 init, 52
 launchBillThread, 53
 launchBillingsThread, 53
 launchBookingsThread, 54
 m_currentFSM, 59
 m_currentUser, 59
 m_mode, 59
 mode, 54
 modeChanged, 55
 openTab, 55
 receiveConfirmation, 55
 receiveInput, 55
 receiveValidation, 55
 replaceInput, 55
 resendText, 56
 SH_ApplicationCore, 50
 sendText, 56
 setMode, 56
 setUser, 57
 SH_ApplicationCore, 50
 user, 57
 userChanged, 58
 userExists, 58
 userLogOut, 58
 SH_ApplicationCore.cpp, 683
 SH_ApplicationCore.h, 686
 SH_BillingCreationStateMachine, 60
 addChildrenNextTransition, 65
 addIOState, 66
 addIOStateMachine, 68
 cancelReplacement, 69
 clearAll, 69
 confirmInput, 70
 displayCalendar, 70
 displayFileDialog, 71
 getContentTypeValue, 71
 historyValue, 72
 ioContent, 72
 ioStatesHistory, 73
 m_ioContent, 79
 m_ioStatesHistory, 79
 m_tableName, 79
 next, 73
 receiveInput, 74
 replaceInput, 74
 resendText, 74
 SH_BillingCreationStateMachine, 62
 sendText, 75
 setContentValue, 75
 setIOStateHistory, 76
 setIOcontent, 76
 setTableName, 77
 SH_BillingCreationStateMachine, 62
 tableName, 77
 toString, 78
 validateInput, 78
 SH_BillingsDelegate, 80
 SH_BillingsTableModel, 82
 addFilterKeyColumn, 85
 containsFilterKeyColumn, 85
 currentSortKeyColumn, 85
 data, 85, 86
 empty, 96
 fetch, 87
 field, 87
 fields, 88
 fieldsCount, 88
 fieldsList, 96
 fillModel, 88
 filterAcceptsRow, 89
 flags, 89
 invalidateFilter, 90
 isEmpty, 90
 lastError, 90, 96
 model, 96
 removeFilterKeyColumn, 90
 roleNames, 91
 SH_BillingsTableModel, 84
 setBooleanColumns, 91
 setData, 91
 setNotNullColumns, 92
 setNullColumns, 92
 setPasswordColumns, 93
 setReadOnlyColumns, 93
 setSortKeyColumn, 93
 SH_BillingsTableModel, 84
 sort, 94
 sortChanged, 95
 sortKeyColumn, 96
 table, 96
 tableName, 95
 SH_BillsTableModel, 97
 addFilterKeyColumn, 100
 containsFilterKeyColumn, 100
 currentSortKeyColumn, 100
 data, 100, 101
 empty, 111
 fetch, 102
 field, 102
 fields, 103
 fieldsCount, 103
 fieldsList, 111
 fillModel, 103
 filterAcceptsRow, 104
 flags, 104
 invalidateFilter, 104
 isEmpty, 105
 lastError, 105, 111
 model, 110
 removeFilterKeyColumn, 105
 roleNames, 105
 SH_BillsTableModel, 99
 setBooleanColumns, 106

setData, 106
setNotNullColumns, 107
setNullColumns, 107
setPasswordColumns, 107
setReadOnlyColumns, 108
setSortKeyColumn, 108
SH_BillsTableModel, 99
sort, 109
sortChanged, 110
sortKeyColumn, 111
table, 111
tableName, 110
SH_BookingsDelegate, 112
fontSize, 113
value, 113
SH_BookingsTableModel, 114
addFilterKeyColumn, 117
containsFilterKeyColumn, 117
currentSortKeyColumn, 117
data, 117, 118
empty, 128
fetch, 119
field, 119
fields, 120
fieldsCount, 120
fieldsList, 128
fillModel, 120
filterAcceptsRow, 121
flags, 121
invalidateFilter, 122
isEmpty, 122
lastError, 122, 128
model, 128
removeFilterKeyColumn, 122
roleNames, 123
SH_BookingsTableModel, 116
setBooleanColumns, 123
setData, 123
setNotNullColumns, 124
setNullColumns, 124
setPasswordColumns, 125
setReadOnlyColumns, 125
setSortKeyColumn, 125
SH_BookingsTableModel, 116
sort, 126
sortChanged, 127
sortKeyColumn, 128
table, 128
tableName, 127
SH_CalendarDialog, 129
clicked, 131
closed, 131
currentDate, 131
currentDay, 131
currentMonth, 131
currentMonthLength, 131
currentWeekday, 131
currentYear, 131
firstWeekdayIndex, 131
monthsList, 131
opened, 131
refresh, 131
selected, 131
text, 131
weekdaysList, 131
SH_ClientCreationStateMachine, 132
addChildrenNextTransition, 136
addIOState, 137
addIOStateMachine, 139
cancelReplacement, 140
clearAll, 140
confirmInput, 141
displayCalendar, 141
displayFileDialog, 142
getContentValue, 142
historyValue, 143
ioContent, 143
ioStatesHistory, 144
m_ioContent, 150
m_ioStatesHistory, 150
m_tableName, 150
next, 144
receiveInput, 145
replaceInput, 145
resendText, 145
SH_ClientCreationStateMachine, 135
sendText, 146
setContentValue, 146
setIOStateHistory, 147
setIOcontent, 147
setTableName, 148
SH_ClientCreationStateMachine, 135
tableName, 148
toString, 149
validateInput, 149
SH_ClientsTableModel, 151
addFilterKeyColumn, 154
containsFilterKeyColumn, 154
currentSortKeyColumn, 154
data, 154, 155
empty, 165
fetch, 156
field, 156
fields, 157
fieldsCount, 157
fieldsList, 165
fillModel, 157
filterAcceptsRow, 158
flags, 158
invalidateFilter, 158
isEmpty, 159
lastError, 159, 165
model, 164
removeFilterKeyColumn, 159
roleNames, 159
SH_ClientsTableModel, 153

setBooleanColumns, 160
 setData, 160
 setNotNullColumns, 161
 setNullColumns, 161
 setPasswordColumns, 161
 setReadOnlyColumns, 162
 setSortKeyColumn, 162
 SH_ClientsTableModel, 153
 sort, 163
 sortChanged, 164
 sortKeyColumn, 165
 table, 165
 tableName, 164
 SH_CommonPage, 166
 cancelProcess, 167
 cancelReplace, 167
 confirm, 167
 keySelected, 167
 quit, 167
 reload, 167
 replace, 167
 selected, 167
 streamBuffer, 167
 validate, 167
 SH_Company, 167
 SH_ConfirmationState, 169
 confirmInput, 171
 display, 172
 input, 172
 next, 173
 onEntry, 173
 onExit, 173
 output, 174
 rawInput, 174
 resendInput, 175
 SH_ConfirmationState, 171
 sendOutput, 175
 setInput, 176
 setOutput, 176
 setVisibility, 177
 SH_ConfirmationState, 171
 toString, 177
 visibility, 178
 SH_ConexionPage, 179
 checkUsername, 180
 login, 180
 loggedIn, 180
 reload, 180
 SH_ContentView, 180
 activeFilterIndicatorIndexes, 182
 addFilterIndex, 182
 emptyDelegate, 182
 filter, 182
 model, 182
 removeFilterIndex, 182
 SH_DataDelegate, 182
 sectionDelegate, 182
 sectionIndex, 182
 selected, 182
 sort, 182
 SH_DataDelegate, 207
 fontSize, 208
 SH_ContentView, 182
 value, 208
 SH_DatabaseContentQuestionState, 182
 answerInvalid, 186
 answerValid, 186
 checkValidity, 187
 choiceList, 187
 display, 188
 displayChoiceList, 188
 givenAnswer, 188
 input, 189
 isAnswerValid, 189
 m_choices, 196
 m_choicesDisplayed, 196
 m_condition, 196
 m_field, 196
 m_table, 196
 next, 189
 onEntry, 190
 onExit, 190
 output, 191
 rawInput, 191
 resendInput, 192
 SH_DatabaseContentQuestionState, 185
 sendOutput, 192
 setGivenAnswer, 192
 setInput, 193
 setOutput, 194
 setVisibility, 194
 SH_DatabaseContentQuestionState, 185
 toString, 195
 visibility, 195
 SH_DatabaseManager, 197
 ~SH_DatabaseManager, 200
 _instance, 207
 dataCount, 200
 dbConnect, 201
 dbConnection, 207
 dbDisconnect, 202
 divideQVariantMap, 202
 execInsertReturningQuery, 203
 execReplaceQuery, 204
 execSelectQuery, 205
 getDbConnection, 205
 getInstance, 205
 isConnected, 206
 SH_DatabaseManager, 199
 SH_DatabaseManager, 199
 tableExists, 206
 SH_DatabaseManager.cpp, 687
 SH_DatabaseManager.h, 691
 dbAliasNameStr, 692
 dbCannotOpenStr, 692
 dbDriverNotExistStr, 692

dbDriverStr, 692
dbFileNameStr, 692
dbFilePathStr, 692
dbFolderPathStr, 692
dbPasswordStr, 692
dbUsernameStr, 693
SH_DateQuestionState, 208
answerInvalid, 212
answerValid, 212
checkValidity, 212
display, 213
getFuture, 213
getPast, 214
givenAnswer, 214
input, 214
isAnswerValid, 214
m_future, 222
m_past, 222
next, 215
onEntry, 215
onExit, 216
output, 216
rawInput, 217
resendInput, 217
SH_DateQuestionState, 211
sendOutput, 217
setFuture, 218
setGivenAnswer, 218
setInput, 218
setOutput, 219
setPast, 220
setVisibility, 220
SH_DateQuestionState, 211
toString, 220
visibility, 221
SH.DecimalQuestionState, 222
answerInvalid, 226
answerValid, 226
checkValidity, 226
display, 227
givenAnswer, 227
input, 228
isAnswerValid, 228
m_max, 237
m_min, 237
max, 228
min, 229
next, 229
onEntry, 230
onExit, 230
output, 231
rawInput, 231
resendInput, 232
SH.DecimalQuestionState, 225
sendOutput, 232
setGivenAnswer, 233
setInput, 233
setMax, 234
setMin, 234
setOutput, 235
setVisibility, 235
SH.DecimalQuestionState, 225
toString, 236
visibility, 236
SH.ExtendedProxyModel, 237
addFilterKeyColumn, 241
booleanSet, 252
containsFilterKeyColumn, 241
currentSortKeyColumn, 241
data, 241, 242
empty, 254
fetch, 243
field, 243
fields, 244
fieldsCount, 244
fieldsList, 254
fillModel, 244
filterAcceptsRow, 245
filters, 253
flags, 245
invalidateFilter, 246
isEmpty, 246
lastError, 246, 254
model, 253
notNullSet, 253
nullSet, 253
passwordSet, 253
readonlySet, 253
removeFilterKeyColumn, 246
replaceSet, 246
roleNames, 247
SH.ExtendedProxyModel, 241
setBooleanColumns, 248
setData, 248
setNotNullColumns, 249
setNullColumns, 249
setPasswordColumns, 249
setReadOnlyColumns, 250
setSortKeyColumn, 250
SH.ExtendedProxyModel, 241
sort, 251
sortChanged, 252
sortIndex, 253
sortKeyColumn, 254
table, 254
tableName, 252
SH.ExtendedQQmlAction, 254
~SH.ExtendedQQmlAction, 258
__icon, 270
enabled, 270
enabledChanged, 258
event, 258
icon, 259
iconChanged, 259
iconName, 260, 270
iconNameChanged, 260

iconSource, 260, 270
 iconSourceChanged, 261
 iconVariant, 261
 isEnabled, 261
 keyShortcut, 261, 270
 keyShortcutChanged, 261
 m_enabled, 268
 m_icon, 269
 m_iconName, 269
 m_iconSource, 269
 m_mnemonic, 269
 m_shortcut, 269
 m_text, 269
 m_tooltip, 269
 SH_ExtendedQQmlAction, 257
 setEnabled, 262
 setIcon, 262
 setIconName, 262
 setIconSource, 262
 setKeySequence, 263
 setKeyShortcut, 264
 setMnemonicFromText, 264
 setShortcut, 265
 setText, 265
 setTooltip, 266
 SH_ExtendedQQmlAction, 257
 shortcut, 266, 270
 shortcutChanged, 266
 text, 266, 270
 textChanged, 267
 toggled, 267
 tooltip, 267, 270
 tooltipChanged, 267
 trigger, 268
 triggered, 268
 SH_ExtendedQQmlAction.cpp
 qShortcutContextMatcher, 730
 SH_FileSelectionState, 270
 display, 273
 input, 274
 next, 274
 onEntry, 275
 onExit, 275
 output, 275
 rawInput, 276
 resendInput, 277
 SH_FileSelectionState, 273
 sendOutput, 277
 setInput, 277
 setOutput, 278
 setVisibility, 279
 SH_FileSelectionState, 273
 toString, 279
 visibility, 280
 SH_GenericState, 281
 name, 283
 next, 284
 onEntry, 284
 onExit, 285
 onMachineStarted, 285
 onTransitionTriggered, 286
 ptraddress, 287
 SH_GenericState, 283
 setName, 287
 SH_GenericState, 283
 toString, 287
 SH_GroupsTableModel, 288
 addFilterKeyColumn, 292
 containsFilterKeyColumn, 292
 currentSortKeyColumn, 292
 data, 292, 293
 empty, 303
 fetch, 294
 field, 294
 fields, 295
 fieldsCount, 295
 fieldsList, 303
 fillModel, 295
 filterAcceptsRow, 296
 flags, 296
 invalidateFilter, 296
 isEmpty, 297
 lastError, 297, 303
 model, 302
 removeFilterKeyColumn, 297
 roleNames, 297
 SH_GroupsTableModel, 291
 setBooleanColumns, 298
 setData, 298
 setNotNullColumns, 299
 setNullColumns, 299
 setPasswordColumns, 299
 setReadOnlyColumns, 300
 setSortKeyColumn, 300
 SH_GroupsTableModel, 291
 sort, 301
 sortChanged, 302
 sortKeyColumn, 303
 table, 303
 tableName, 302
 SH_HeaderView, 304
 checked, 305
 delegateModel, 305
 model, 305
 up, 305
 SH_InOutState, 305
 display, 308
 input, 309
 m_display, 315
 m_input, 315
 m_isVisible, 316
 m_output, 316
 next, 309
 onEntry, 310
 onExit, 310
 output, 310

rawInput, 311
resendInput, 312
SH_InOutState, 308
sendOutput, 312
setInput, 312
setOutput, 313
setVisibility, 314
SH_InOutState, 308
toString, 314
visibility, 315
SH_InOutStateMachine, 316
addChildrenNextTransition, 319
addIOState, 320
addIOStateMachine, 322
cancelReplacement, 322
clearAll, 323
confirmInput, 323
displayCalendar, 324
displayFileDialog, 324
getContentValue, 325
historyValue, 325
ioContent, 326
ioStatesHistory, 326
m_ioContent, 334
m_ioStatesHistory, 334
m_tableName, 334
name, 327
next, 327
ptraddress, 328
receiveInput, 328
replaceInput, 328
resendText, 329
SH_InOutStateMachine, 318
sendText, 329
setContentValue, 330
setIOStateHistory, 331
setIOStatesHistory, 331
setIcontent, 330
setName, 332
setTableName, 332
SH_InOutStateMachine, 318
tableName, 332
toString, 333
validateInput, 334
SH_Keyboard, 335
actionsList, 336
SH_MessageManager, 358
errorMessage, 360
ErrorMode, 360
infoMessage, 361
successMessage, 361
SH_MessageManager.cpp, 694
SH_MessageManager.h, 695
SH_NamedObject, 362
m_name, 365
m_ptraddress, 365
name, 364
ptraddress, 364
SH_NamedObject, 363
setName, 364
SH_NamedObject, 363
toString, 365
SH_NumericQuestionState, 366
answerInvalid, 369
answerValid, 369
checkValidity, 369
display, 370
givenAnswer, 370
input, 371
isAnswerValid, 371
m_max, 380
m_min, 380
max, 371
min, 372
next, 372
onEntry, 373
onExit, 373
output, 374
rawInput, 374
resendInput, 375
SH_NumericQuestionState, 368
sendOutput, 375
setGivenAnswer, 376
setInput, 376
setMax, 377
setMin, 377
setOutput, 378
setVisibility, 378
SH_NumericQuestionState, 368
toString, 379
visibility, 379
SH_OutputZone, 381
clear, 382
clearAll, 383
display, 383
displayCalendar, 383
displayNew, 383
displayNewFixed, 383
displaySqlDatas, 383
displaySqlDetail, 383
displayText, 383
lastVisibleRow, 383
replace, 383
selected, 383
SH_PrintingState, 383
next, 386
onEntry, 386
onExit, 387
printFinished, 388
printStarted, 388
SH_PrintingState, 386
SH_PrintingState, 386
toString, 388
SH_QuestionState, 389
answerInvalid, 391
answerValid, 392

checkValidity, 392
 display, 393
 givenAnswer, 393
 input, 393
 isAnswerValid, 394
 m_givenAnswer, 401
 next, 394
 onEntry, 394
 onExit, 395
 output, 395
 rawInput, 396
 resendInput, 397
 SH_QuestionState, 391
 sendOutput, 397
 setGivenAnswer, 397
 setInput, 398
 setOutput, 398
 setVisibility, 399
 SH_QuestionState, 391
 toString, 400
 visibility, 400
 SH_RegExpQuestionState, 401
 answerInvalid, 404
 answerValid, 405
 checkValidity, 405
 display, 406
 givenAnswer, 406
 input, 407
 isAnswerValid, 407
 m_regex, 417
 maxLen, 407
 minLen, 408
 next, 408
 onEntry, 409
 onExit, 409
 output, 410
 rawInput, 410
 regexp, 411
 resendInput, 411
 SH_RegExpQuestionState, 404
 sendOutput, 412
 setGivenAnswer, 412
 setInput, 413
 setMaxLen, 413
 setMinLen, 414
 setOutput, 414
 setRegexp, 415
 setVisibility, 415
 SH_RegExpQuestionState, 404
 toString, 416
 visibility, 416
 SH_RoomsDelegate, 418
 fontSize, 419
 value, 419
 SH_RoomsSectionsDelegate, 420
 value, 421
 SH_RoomsTableModel, 421
 addFilterKeyColumn, 425
 containsFilterKeyColumn, 425
 currentSortKeyColumn, 425
 data, 425, 426
 empty, 436
 fetch, 427
 field, 427
 fields, 428
 fieldsCount, 428
 fieldsList, 436
 fillModel, 428
 filterAcceptsRow, 429
 flags, 430
 invalidateFilter, 430
 isEmpty, 430
 lastError, 430, 436
 model, 436
 removeFilterKeyColumn, 431
 roleNames, 431
 SH_RoomsTableModel, 424
 setBooleanColumns, 431
 setData, 431
 setNotNullColumns, 432
 setNullColumns, 432
 setPasswordColumns, 433
 setReadOnlyColumns, 433
 setSortKeyColumn, 434
 SH_RoomsTableModel, 424
 sort, 434
 sortChanged, 435
 sortKeyColumn, 436
 table, 436
 tableName, 435
 SH_ServiceCharging, 437
 addChildrenNextTransition, 441
 addIOState, 443
 addIOMachine, 445
 cancelReplacement, 446
 clearAll, 446
 confirmInput, 446
 current, 447
 displayCalendar, 447
 displayFileDialog, 448
 getContentType, 448
 historyValue, 449
 ioContent, 449
 ioStatesHistory, 450
 limit, 450, 459
 limitChanged, 450
 m_ioContent, 458
 m_ioStatesHistory, 458
 m_priceMin, 459
 m_tableName, 459
 m_vat, 459
 next, 451
 receiveInput, 451
 replaceInput, 451
 resendText, 452
 SH_ServiceCharging, 439

sendText, 452
setContentValue, 453
setCurrent, 453
setIOStateHistory, 454
setIOcontent, 454
setLimit, 455
setPersistentContentValue, 455
setTableName, 456
SH_ServiceCharging, 439
stopLooping, 456
tableName, 457
toString, 457
validateInput, 458
SH_ServicesDelegate, 460
fontSize, 461
value, 461
SH_ServicesTableModel, 462
addFilterKeyColumn, 465
containsFilterKeyColumn, 465
currentSortKeyColumn, 465
data, 465, 466
empty, 476
fetch, 467
field, 467
fields, 468
fieldsCount, 468
fieldsList, 476
fillModel, 468
filterAcceptsRow, 469
flags, 469
invalidateFilter, 470
isEmpty, 470
lastError, 470, 476
model, 475
removeFilterKeyColumn, 470
roleNames, 470
SH_ServicesTableModel, 464
setBooleanColumns, 471
setData, 471
setNotNullColumns, 472
setNullColumns, 472
setPasswordColumns, 473
setReadOnlyColumns, 473
setSortKeyColumn, 473
SH_ServicesTableModel, 464
sort, 474
sortChanged, 475
sortKeyColumn, 476
table, 476
tableName, 475
SH_SqlDataFields, 476
m_name, 483
m_sortOrder, 483
m_text, 483
name, 479, 483
nameChanged, 479
role, 479, 483
roleChanged, 480
SH_SqlDataFields, 479
setName, 480
setSortOrder, 481
setText, 481
SH_SqlDataFields, 479
sortOrder, 482, 483
sortOrderChanged, 482
text, 482, 483
textChanged, 482
SH_SqlDataModel, 483
applyRoles, 486
data, 487
datas, 488
fetch, 489
field, 490
fieldFromRole, 491
fieldsChanged, 491
fieldsCount, 492
fieldsList, 492
filter, 493, 501
filterChanged, 493
isEmpty, 493
lastError, 494, 501
lastErrorChanged, 494
mDataFields, 500
mFilter, 500
mRecords, 500
mRoles, 500
mSort, 500
mSqlQuery, 501
mTable, 501
query, 494
resetFieldsToAll, 494
resetFilterCondition, 495
roleForField, 495
roleNames, 495
rolesChanged, 495
rowCount, 496
SH_SqlDataModel, 486
setFields, 496
setFilterCondition, 497
setHeaderData, 497
setOrderBy, 498
setTable, 498
SH_SqlDataModel, 486
table, 501
tableChanged, 499
tableName, 499
SH_SqlDataView, 502
emptyDelegate, 504
filterIndicatorsVisibles, 504
filtersTitle, 504
isEmpty, 504
itemDelegate, 504
newItem, 503
sectionDelegate, 504
selected, 503
sqlModel, 504

SH_SqlTableView, 505
 selected, 506
SH_StatementState, 506
 display, 509
 input, 510
 next, 510
 onEntry, 511
 onExit, 511
 output, 512
 rawInput, 512
 resendInput, 513
SH_StatementState, 509
 sendOutput, 513
 setInput, 513
 setOutput, 513
 setVisibility, 514
SH_StatementState, 509
 toString, 515
 visibility, 515
SH_StringQuestionState, 516
 answerInvalid, 520
 answerValid, 520
 checkValidity, 520
 display, 521
 givenAnswer, 521
 input, 522
 isAnswerValid, 522
 m_maxLen, 531
 m_minLen, 531
 maxLen, 522
 minLen, 523
 next, 523
 onEntry, 524
 onExit, 524
 output, 525
 rawInput, 525
 resendInput, 526
SH_StringQuestionState, 519
 sendOutput, 526
 setGivenAnswer, 527
 setInput, 527
 setMaxLen, 528
 setMinLen, 528
 setOutput, 529
 setVisibility, 529
SH_StringQuestionState, 519
 toString, 530
 visibility, 530
SH_TabZone, 532
 newBilling, 533
 newBooking, 533
 newSelling, 533
 openTab, 533
 reload, 533
 selected, 533
 selectedForDetail, 533
 stdKeyboard, 533
SH_Trainee, 533
 administrator, 542
 exists, 536
 id, 537, 542
 isAdministrator, 537
 isManagerX, 538
 isManagerZ, 538
 isReceptionist, 538
 isValid, 539
 login, 539
 managerX, 542
 managerZ, 542
 name, 540, 543
 nameChanged, 541
 receptionist, 543
 roles, 541, 543
 rolesChanged, 541
SH_Trainee, 536
SH_Trainee, 536
 traineeExists, 541
 userExists, 541
 valid, 543
 validityChanged, 542
SH_TriStateCheckImage, 544
 checked, 546
 imgDown, 546
 imgUp, 546
 innerHeight, 546
 innerWidth, 546
 labelSpacing, 546
 length, 546
 pressed, 545
 text, 546
 up, 546
SH_User, 546
 administrator, 559
 exists, 550
 id, 551, 559
 isAdministrator, 551
 isManagerX, 552
 isManagerZ, 552
 isReceptionist, 553
 isValid, 553
 login, 554
 m_administrator, 558
 m_id, 558
 m_managerX, 558
 m_managerZ, 558
 m_name, 558
 m_receptionist, 559
 managerX, 559
 managerZ, 559
 name, 555, 559
 nameChanged, 555
 receptionist, 559
 roles, 555, 559
 rolesChanged, 555
SH_User, 549
 setID, 555

setName, 556
SH_User, 549
traineeExists, 556
userExists, 557
valid, 559
validityChanged, 558
SH_ValidationState, 560
confirmInput, 562
display, 563
input, 563
next, 564
onEntry, 564
onExit, 564
output, 565
rawInput, 565
resendInput, 566
SH_ValidationState, 562
sendOutput, 566
setInput, 567
setOutput, 567
setVisibility, 568
SH_ValidationState, 562
toString, 568
visibility, 569
SH_WelcomePage, 570
clicked, 571
logOut, 571
loggedOut, 571
quit, 571
reload, 571
SH_app, 45
pages, 46
reload, 46
sectionDelegate
 SH_ContentView, 182
 SH_SqlDataView, 504
sectionIndex
 SH_ContentView, 182
selected
 SH_CalendarDialog, 131
 SH_CommonPage, 167
 SH_ContentView, 182
 SH_OutputZone, 383
 SH_SqlDataView, 503
 SH_SqlTableView, 506
 SH_TabZone, 533
selectedForDetail
 SH_TabZone, 533
sendOutput
 SH_ConfirmationState, 175
 SH_DatabaseContentQuestionState, 192
 SH_DateQuestionState, 217
 SH.DecimalQuestionState, 232
 SH_FileSelectionState, 277
 SH_InOutState, 312
 SH_NumericQuestionState, 375
 SH_QuestionState, 397
 SH_RegExpQuestionState, 412
 SH_StatementState, 513
 SH_StringQuestionState, 526
 SH_ValidationState, 566
sendText
 SH_AddressCreationStateMachine, 40
 SH_ApplicationCore, 56
 SH_BillingCreationStateMachine, 75
 SH_ClientCreationStateMachine, 146
 SH_InOutStateMachine, 329
 Sh_LoopingInOutStateMachine, 351
 SH_ServiceCharging, 452
setBooleanColumns
 SH_BillingsTableModel, 91
 SH_BillsTableModel, 106
 SH_BookingsTableModel, 123
 SH_ClientsTableModel, 160
 SH_ExtendedProxyModel, 248
 SH_GroupsTableModel, 298
 SH_RoomsTableModel, 431
 SH_ServicesTableModel, 471
setContentValue
 SH_AddressCreationStateMachine, 40
 SH_BillingCreationStateMachine, 75
 SH_ClientCreationStateMachine, 146
 SH_InOutStateMachine, 330
 Sh_LoopingInOutStateMachine, 351
 SH_ServiceCharging, 453
setCurrent
 Sh_LoopingInOutStateMachine, 352
 SH_ServiceCharging, 453
setData
 SH_BillingsTableModel, 91
 SH_BillsTableModel, 106
 SH_BookingsTableModel, 123
 SH_ClientsTableModel, 160
 SH_ExtendedProxyModel, 248
 SH_GroupsTableModel, 298
 SH_RoomsTableModel, 431
 SH_ServicesTableModel, 471
setEnabled
 SH_ExtendedQQmlAction, 262
setFields
 SH_SqlDataModel, 496
setFilterCondition
 SH_SqlDataModel, 497
setFuture
 SH_DateQuestionState, 218
setGivenAnswer
 SH_DatabaseContentQuestionState, 192
 SH_DateQuestionState, 218
 SH.DecimalQuestionState, 233
 SH_NumericQuestionState, 376
 SH_QuestionState, 397
 SH_RegExpQuestionState, 412
 SH_StringQuestionState, 527
setHeaderData
 SH_SqlDataModel, 497
setID

SH_User, 555
setIOStateHistory
 SH_AddressCreationStateMachine, 41
 SH_BillingCreationStateMachine, 76
 SH_ClientCreationStateMachine, 147
 SH_InOutStateMachine, 331
 Sh_LoopingInOutStateMachine, 353
 SH_ServiceCharging, 454
setIStatesHistory
 SH_InOutStateMachine, 331
setIcontent
 SH_AddressCreationStateMachine, 41
 SH_BillingCreationStateMachine, 76
 SH_ClientCreationStateMachine, 147
 SH_InOutStateMachine, 330
 Sh_LoopingInOutStateMachine, 352
 SH_ServiceCharging, 454
setIcon
 SH_ExtendedQQmlAction, 262
setIconName
 SH_ExtendedQQmlAction, 262
setIconSource
 SH_ExtendedQQmlAction, 262
setInput
 SH_ConfirmationState, 176
 SH_DatabaseContentQuestionState, 193
 SH_DateQuestionState, 218
 SH.DecimalQuestionState, 233
 SH_FileSelectionState, 277
 SH_InOutState, 312
 SH_NumericQuestionState, 376
 SH_QuestionState, 398
 SH_RegExpQuestionState, 413
 SH_StatementState, 513
 SH_StringQuestionState, 527
 SH_ValidationState, 567
setKeySequence
 SH_ExtendedQQmlAction, 263
setKeyShortcut
 SH_ExtendedQQmlAction, 264
setLimit
 Sh_LoopingInOutStateMachine, 353
 SH_ServiceCharging, 455
setMax
 SH.DecimalQuestionState, 234
 SH_NumericQuestionState, 377
setMaxLen
 SH_RegExpQuestionState, 413
 SH_StringQuestionState, 528
setMin
 SH.DecimalQuestionState, 234
 SH_NumericQuestionState, 377
setMinLen
 SH_RegExpQuestionState, 414
 SH_StringQuestionState, 528
setMnemonicFromText
 SH_ExtendedQQmlAction, 264
setMode

SH_ApplicationCore, 56
setName
 SH_GenericState, 287
 SH_InOutStateMachine, 332
 SH_NamedObject, 364
 SH_SqlDataFields, 480
 SH_User, 556
setNotNullColumns
 SH_BillingsTableModel, 92
 SH_BillsTableModel, 107
 SH_BookingsTableModel, 124
 SH_ClientsTableModel, 161
 SH_ExtendedProxyModel, 249
 SH_GroupsTableModel, 299
 SH_RoomsTableModel, 432
 SH_ServicesTableModel, 472
setNullColumns
 SH_BillingsTableModel, 92
 SH_BillsTableModel, 107
 SH_BookingsTableModel, 124
 SH_ClientsTableModel, 161
 SH_ExtendedProxyModel, 249
 SH_GroupsTableModel, 299
 SH_RoomsTableModel, 432
 SH_ServicesTableModel, 472
setOrderBy
 SH_SqlDataModel, 498
setOutput
 SH_ConfirmationState, 176
 SH_DatabaseContentQuestionState, 194
 SH_DateQuestionState, 219
 SH.DecimalQuestionState, 235
 SH_FileSelectionState, 278
 SH_InOutState, 313
 SH_NumericQuestionState, 378
 SH_QuestionState, 398
 SH_RegExpQuestionState, 414
 SH_StatementState, 513
 SH_StringQuestionState, 529
 SH_ValidationState, 567
setPasswordColumns
 SH_BillingsTableModel, 93
 SH_BillsTableModel, 107
 SH_BookingsTableModel, 125
 SH_ClientsTableModel, 161
 SH_ExtendedProxyModel, 249
 SH_GroupsTableModel, 299
 SH_RoomsTableModel, 433
 SH_ServicesTableModel, 473
setPast
 SH_DateQuestionState, 220
setPersistentContentValue
 Sh_LoopingInOutStateMachine, 354
 SH_ServiceCharging, 455
setReadOnlyColumns
 SH_BillingsTableModel, 93
 SH_BillsTableModel, 108
 SH_BookingsTableModel, 125

SH_ClientsTableModel, 162
SH_ExtendedProxyModel, 250
SH_GroupsTableModel, 300
SH_RoomsTableModel, 433
SH_ServicesTableModel, 473
setRegexp
 SH_RegExpQuestionState, 415
setShortcut
 SH_ExtendedQQmlAction, 265
setSortKeyColumn
 SH_BillingsTableModel, 93
 SH_BillsTableModel, 108
 SH_BookingsTableModel, 125
 SH_ClientsTableModel, 162
 SH_ExtendedProxyModel, 250
 SH_GroupsTableModel, 300
 SH_RoomsTableModel, 434
 SH_ServicesTableModel, 473
setSortOrder
 SH_SqlDataFields, 481
setTable
 SH_SqlDataModel, 498
setTableName
 SH_AddressCreationStateMachine, 42
 SH_BillingCreationStateMachine, 77
 SH_ClientCreationStateMachine, 148
 SH_InOutStateMachine, 332
 Sh_LoopingInOutStateMachine, 354
 SH_ServiceCharging, 456
setText
 SH_ExtendedQQmlAction, 265
 SH_SqlDataFields, 481
setTooltip
 SH_ExtendedQQmlAction, 266
setUser
 SH_ApplicationCore, 57
setVisibility
 SH_ConfirmationState, 177
 SH_DatabaseContentQuestionState, 194
 SH_DateQuestionState, 220
 SH.DecimalQuestionState, 235
 SH_FileSelectionState, 279
 SH_InOutState, 314
 SH_NumericQuestionState, 378
 SH_QuestionState, 399
 SH_RegExpQuestionState, 415
 SH_StatementState, 514
 SH_StringQuestionState, 529
 SH_ValidationState, 568
Sh_LoopingInOutStateMachine, 336
 addChildrenNextTransition, 339
 addIOState, 341
 addIOStateMachine, 343
 cancelReplacement, 344
 clearAll, 344
 confirmInput, 345
 current, 345
 displayCalendar, 346
 displayFileDialog, 346
 getContentType, 347
 historyValue, 347
 ioContent, 348
 ioStatesHistory, 348
 limit, 349, 358
 limitChanged, 349
 m_contents, 357
 m_current, 357
 m_ioContent, 357
 m_ioStatesHistory, 357
 m_limit, 357
 m_persistentContent, 358
 m_tableName, 358
 next, 349
 receiveInput, 350
 replaceInput, 350
 resendText, 351
 sendText, 351
 setContentValue, 351
 setCurrent, 352
 setIOStateHistory, 353
 setIOcontent, 352
 setLimit, 353
 setPersistentContentValue, 354
 setTableName, 354
 Sh_LoopingInOutStateMachine, 339
 Sh_LoopingInOutStateMachine, 339
 stopLooping, 355
 tableName, 355
 toString, 356
 validateInput, 356
shortcut
 SH_ExtendedQQmlAction, 266, 270
shortcutChanged
 SH_ExtendedQQmlAction, 266
sort
 SH_BillingsTableModel, 94
 SH_BillsTableModel, 109
 SH_BookingsTableModel, 126
 SH_ClientsTableModel, 163
 SH_ContentView, 182
 SH_ExtendedProxyModel, 251
 SH_GroupsTableModel, 301
 SH_RoomsTableModel, 434
 SH_ServicesTableModel, 474
sortChanged
 SH_BillingsTableModel, 95
 SH_BillsTableModel, 110
 SH_BookingsTableModel, 127
 SH_ClientsTableModel, 164
 SH_ExtendedProxyModel, 252
 SH_GroupsTableModel, 302
 SH_RoomsTableModel, 435
 SH_ServicesTableModel, 475
sortIndex
 SH_ExtendedProxyModel, 253
sortKeyColumn

SH_BillingsTableModel, 96
 SH_BillsTableModel, 111
 SH_BookingsTableModel, 128
 SH_ClientsTableModel, 165
 SH_ExtendedProxyModel, 254
 SH_GroupsTableModel, 303
 SH_RoomsTableModel, 436
 SH_ServicesTableModel, 476
sortOrder
 SH_SQLDataFields, 482, 483
sortOrderChanged
 SH_SQLDataFields, 482
spin
 main.cpp, 646
sqlModel
 SH_SQLTableView, 504
statusChanged
 main.cpp, 646
stdKeyboard
 SH_TabZone, 533
stopLooping
 Sh_LoopingInOutStateMachine, 355
 SH_ServiceCharging, 456
streamBuffer
 SH_CommonPage, 167
successMessage
 SH_MessageManager, 361
TEST
 SH_MessageManager, 360
TabView, 572
table
 SH_BillingsTableModel, 96
 SH_BillsTableModel, 111
 SH_BookingsTableModel, 128
 SH_ClientsTableModel, 165
 SH_ExtendedProxyModel, 254
 SH_GroupsTableModel, 303
 SH_RoomsTableModel, 436
 SH_ServicesTableModel, 476
 SH_SQLDataModel, 501
tableChanged
 SH_SQLDataModel, 499
tableExists
 SH_DatabaseManager, 206
tableName
 SH_AddressCreationStateMachine, 42
 SH_BillingCreationStateMachine, 77
 SH_BillingsTableModel, 95
 SH_BillsTableModel, 110
 SH_BookingsTableModel, 127
 SH_ClientCreationStateMachine, 148
 SH_ClientsTableModel, 164
 SH_ExtendedProxyModel, 252
 SH_GroupsTableModel, 302
 SH_InOutStateMachine, 332
 Sh_LoopingInOutStateMachine, 355
 SH_RoomsTableModel, 435
 SH_ServiceCharging, 457
SH_ServicesTableModel, 475
SH_SQLDataModel, 499
TableView, 571
text
 SH_CalendarDialog, 131
 SH_ExtendedQQmlAction, 266, 270
 SH_SQLDataFields, 482, 483
 SH_TriStateCheckImage, 546
textChanged
 SH_ExtendedQQmlAction, 267
 SH_SQLDataFields, 482
toString
 SH_AdaptDatabaseState, 26
 SH_AddressCreationStateMachine, 43
 SH_BillingCreationStateMachine, 78
 SH_ClientCreationStateMachine, 149
 SH_ConfirmationState, 177
 SH_DatabaseContentQuestionState, 195
 SH_DateQuestionState, 220
 SH.DecimalQuestionState, 236
 SH_FileSelectionState, 279
 SH_GenericState, 287
 SH_InOutState, 314
 SH_InOutStateMachine, 333
 Sh_LoopingInOutStateMachine, 356
 SH_NamedObject, 365
 SH_NumericQuestionState, 379
 SH_PrintingState, 388
 SH_QuestionState, 400
 SH_RegExpQuestionState, 416
 SH_ServiceCharging, 457
 SH_StatementState, 515
 SH_StringQuestionState, 530
 SH_ValidationState, 568
toggled
 SH_ExtendedQQmlAction, 267
tooltip
 SH_ExtendedQQmlAction, 267, 270
tooltipChanged
 SH_ExtendedQQmlAction, 267
traineeExists
 SH_Trainee, 541
 SH_User, 556
trigger
 SH_ExtendedQQmlAction, 268
triggered
 SH_ExtendedQQmlAction, 268
up
 SH_HeaderView, 305
 SH_TriStateCheckImage, 546
user
 SH_ApplicationCore, 57
userChanged
 SH_ApplicationCore, 58
userExists
 SH_ApplicationCore, 58
 SH_Trainee, 541
 SH_User, 557

userLogOut
 SH_ApplicationCore, 58

valid
 SH_Trainee, 543
 SH_User, 559

validate
 SH_CommonPage, 167

validateInput
 SH_AddressCreationStateMachine, 43
 SH_BillingCreationStateMachine, 78
 SH_ClientCreationStateMachine, 149
 SH_InOutStateMachine, 334
 Sh_LoopingInOutStateMachine, 356
 SH_ServiceCharging, 458

validityChanged
 SH_Trainee, 542
 SH_User, 558

value
 SH_BillingsDelegate, 81
 SH_BookingsDelegate, 113
 SH_DataDelegate, 208
 SH_RoomsDelegate, 419
 SH_RoomsSectionsDelegate, 421
 SH_ServicesDelegate, 461

views/SH_ExtendedQQmlAction.cpp, 730, 731

views/SH_ExtendedQQmlAction.h, 733, 734

views/qml/SH_BillingsDelegate.qml, 697

views/qml/SH_BookingsDelegate.qml, 697, 698

views/qml/SH_CalendarDialog.qml, 698

views/qml/SH_CommonPage.qml, 701

views/qml/SH_ConexionPage.qml, 711

views/qml/SH_ContentView.qml, 712

views/qml/SH_DataDelegate.qml, 716

views/qml/SH_HeaderView.qml, 717

views/qml/SH_Keyboard.qml, 718

views/qml/SH_OutputZone.qml, 718, 719

views/qml/SH_RoomsDelegate.qml, 721

views/qml/SH_RoomsSectionsDelegate.qml, 722

views/qml/SH_ServicesDelegate.qml, 722

views/qml/SH_SQLDataView.qml, 722, 723

views/qml/SH_SQLTableView.qml, 724

views/qml/SH_TabZone.qml, 725

views/qml/SH_TriStateCheckImage.qml, 727

views/qml/SH_WelcomePage.qml, 728

views/qml/SH_app.qml, 696

visibility
 SH_ConfirmationState, 178
 SH_DatabaseContentQuestionState, 195
 SH_DateQuestionState, 221
 SH.DecimalQuestionState, 236
 SH_FileSelectionState, 280
 SH_InOutState, 315
 SH_NumericQuestionState, 379
 SH_QuestionState, 400
 SH_RegExpQuestionState, 416
 SH_StatementState, 515
 SH_StringQuestionState, 530
 SH_ValidationState, 569