

Prüfungsamt-Projekt

1.0.0

Erzeugt von Doxygen 1.7.6.1

Sam Aug 25 2012 00:03:41

Inhaltsverzeichnis

1	Klassen-Verzeichnis	1
1.1	Klassenhierarchie	1
2	Klassen-Verzeichnis	3
2.1	Auflistung der Klassen	3
3	Datei-Verzeichnis	5
3.1	Auflistung der Dateien	5
4	Klassen-Dokumentation	7
4.1	AttributeEditor Klassenreferenz	7
4.1.1	Beschreibung der Konstruktoren und Destruktoren	8
4.1.1.1	AttributeEditor	8
4.1.1.2	~AttributeEditor	8
4.1.2	Dokumentation der Elementfunktionen	9
4.1.2.1	activated	9
4.1.2.2	clicked	9
4.1.2.3	fetchTuple	9
4.1.2.4	fetchValueList	9
4.1.2.5	getValue	9
4.1.2.6	mark	9
4.1.2.7	returnPressed	9
4.1.2.8	setValue	9
4.1.2.9	userUpdate	9
4.1.2.10	valueSelected	10
4.1.3	Dokumentation der Datenelemente	10

4.1.3.1	Attribute	10
4.1.3.2	Button	10
4.1.3.3	ComboBox	10
4.1.3.4	Connection	10
4.1.3.5	Label	10
4.1.3.6	LineEdit	10
4.1.3.7	SelectAttribute	10
4.1.3.8	SelectInfoAttributes	10
4.1.3.9	SelectList	10
4.1.3.10	Selector	10
4.1.3.11	SelectValue	10
4.1.3.12	SelectView	10
4.1.3.13	View	10
4.2	DescriptionEntry Strukturreferenz	10
4.2.1	Ausführliche Beschreibung	11
4.2.2	Dokumentation der Datenelemente	11
4.2.2.1	Field	11
4.2.2.2	Help	11
4.2.2.3	Title	11
4.3	JoinEntry Strukturreferenz	11
4.3.1	Ausführliche Beschreibung	12
4.3.2	Dokumentation der Datenelemente	12
4.3.2.1	JoinAttributeR1	12
4.3.2.2	JoinAttributeR2	12
4.3.2.3	JoinTitle	12
4.3.2.4	View	12
4.4	list_t Strukturreferenz	12
4.4.1	Dokumentation der Datenelemente	12
4.4.1.1	no_of_elements	12
4.4.1.2	value	12
4.5	LoginDialog Klassenreferenz	13
4.5.1	Ausführliche Beschreibung	13
4.5.2	Beschreibung der Konstruktoren und Destruktoren	13
4.5.2.1	LoginDialog	13

4.5.3	Dokumentation der Elementfunktionen	14
4.5.3.1	login	14
4.5.4	Dokumentation der Datenelemente	14
4.5.4.1	LineEdit	14
4.6	OptimizeTable Strukturreferenz	14
4.6.1	Ausführliche Beschreibung	15
4.6.2	Dokumentation der Datenelemente	15
4.6.2.1	Table	15
4.6.2.2	Tables	15
4.7	PAClient Klassenreferenz	15
4.7.1	Ausführliche Beschreibung	16
4.7.2	Beschreibung der Konstruktoren und Destruktoren	16
4.7.2.1	PAClient	16
4.7.2.2	~PAClient	16
4.7.3	Dokumentation der Elementfunktionen	16
4.7.3.1	closeMonitor	16
4.7.3.2	information	16
4.7.3.3	openMonitor	17
4.7.3.4	optimize	17
4.7.3.5	quit	17
4.7.4	Dokumentation der Datenelemente	17
4.7.4.1	Connection	17
4.7.4.2	Monitor	17
4.8	SelectEntry Strukturreferenz	17
4.8.1	Ausführliche Beschreibung	17
4.8.2	Dokumentation der Datenelemente	18
4.8.2.1	SelectAttributeR1	18
4.8.2.2	SelectAttributeR2	18
4.8.2.3	SelectInfoAttributes	18
4.8.2.4	UseComboBox	18
4.8.2.5	View	18
4.9	SQLConnection Klassenreferenz	18
4.9.1	Ausführliche Beschreibung	19
4.9.2	Beschreibung der Konstruktoren und Destruktoren	19

4.9.2.1	SQLConnection	19
4.9.3	Dokumentation der Elementfunktionen	20
4.9.3.1	beginTransaction	20
4.9.3.2	commitTransaction	20
4.9.3.3	createCursor	20
4.9.3.4	deleteCursor	20
4.9.3.5	execute	20
4.9.3.6	executeCommandOk	21
4.9.3.7	executeTuplesOk	21
4.9.3.8	fetchCursor	21
4.9.3.9	getField	21
4.9.3.10	getFields	21
4.9.3.11	getMonitor	21
4.9.3.12	getTuples	22
4.9.3.13	getValue	22
4.9.3.14	optimize	22
4.9.3.15	rollbackTransaction	22
4.9.3.16	setMonitor	23
4.9.4	Dokumentation der Datenelemente	23
4.9.4.1	Database	23
4.9.4.2	Monitor	23
4.9.4.3	Transaction	23
4.10	SQLConnectionInterface Klassenreferenz	23
4.10.1	Ausführliche Beschreibung	24
4.10.2	Dokumentation der Elementfunktionen	24
4.10.2.1	beginTransaction	24
4.10.2.2	commitTransaction	24
4.10.2.3	createCursor	24
4.10.2.4	deleteCursor	25
4.10.2.5	execute	25
4.10.2.6	fetchCursor	25
4.10.2.7	getField	25
4.10.2.8	getFields	26
4.10.2.9	getMonitor	26

4.10.2.10	getTuples	26
4.10.2.11	getValue	26
4.10.2.12	optimize	27
4.10.2.13	rollbackTransaction	27
4.10.2.14	setMonitor	27
4.11	SQLException Klassenreferenz	27
4.11.1	Ausführliche Beschreibung	28
4.11.2	Beschreibung der Konstruktoren und Destruktoren	28
4.11.2.1	SQLException	28
4.11.2.2	~SQLException	29
4.11.3	Dokumentation der Elementfunktionen	29
4.11.3.1	toString	29
4.11.4	Freundbeziehungen und Funktionsdokumentation	29
4.11.4.1	operator<<	29
4.11.5	Dokumentation der Datenelemente	29
4.11.5.1	MaxTitleLength	29
4.11.5.2	Title	29
4.12	SQLMonitorInterface Klassenreferenz	29
4.12.1	Ausführliche Beschreibung	30
4.12.2	Dokumentation der Elementfunktionen	30
4.12.2.1	write	30
4.13	TableView Klassenreferenz	30
4.13.1	Ausführliche Beschreibung	32
4.13.2	Dokumentation der Aufzählungstypen	32
4.13.2.1	TableViewMode	32
4.13.3	Beschreibung der Konstruktoren und Destruktoren	32
4.13.3.1	TableView	32
4.13.3.2	~TableView	33
4.13.4	Dokumentation der Elementfunktionen	33
4.13.4.1	cancelClicked	33
4.13.4.2	closeEditor	33
4.13.4.3	closeEvent	33
4.13.4.4	doneSelection	33
4.13.4.5	loadTable	33

4.13.4.6	newClicked	34
4.13.4.7	nullClicked	34
4.13.4.8	okayClicked	34
4.13.4.9	search	34
4.13.4.10	selectTuple	34
4.13.4.11	selectView	34
4.13.4.12	setSortAttribute	34
4.13.5	Dokumentation der Datenelemente	34
4.13.5.1	Attributes	34
4.13.5.2	Connection	34
4.13.5.3	FindLabel	34
4.13.5.4	FindLineEdit	34
4.13.5.5	FindSection	34
4.13.5.6	JoinAttribute	34
4.13.5.7	JoinValue	34
4.13.5.8	MinNotFoundIndex	35
4.13.5.9	Mode	35
4.13.5.10	SelectAttribute	35
4.13.5.11	SelectValue	35
4.13.5.12	SkipNextUpdate	35
4.13.5.13	StatusBar	35
4.13.5.14	TableView	35
4.13.5.15	TupleEditors	35
4.13.5.16	View	35
4.13.5.17	Views	35
4.13.5.18	WhatsThis	35
4.14	TextMonitor Klassenreferenz	35
4.14.1	Ausführliche Beschreibung	36
4.14.2	Beschreibung der Konstruktoren und Destruktoren	36
4.14.2.1	TextMonitor	36
4.14.3	Dokumentation der Elementfunktionen	36
4.14.3.1	write	36
4.14.4	Dokumentation der Datenelemente	37
4.14.4.1	Output	37

4.15 Tuple Strukturreferenz	37
4.15.1 Ausführliche Beschreibung	37
4.15.2 Dokumentation der Datenelemente	38
4.15.2.1 Attribute	38
4.15.2.2 Attributes	38
4.15.2.3 ID	38
4.15.2.4 IsKey	38
4.15.2.5 MaxAttributes	38
4.15.2.6 TableName	38
4.15.2.7 Value	38
4.15.2.8 View	38
4.16 TupleEditor Klassenreferenz	38
4.16.1 Ausführliche Beschreibung	39
4.16.2 Beschreibung der Konstruktoren und Destruktoren	39
4.16.2.1 TupleEditor	39
4.16.2.2 ~TupleEditor	40
4.16.3 Dokumentation der Elementfunktionen	40
4.16.3.1 abort	40
4.16.3.2 closeEvent	40
4.16.3.3 done	40
4.16.3.4 getTuple	40
4.16.3.5 remove	40
4.16.3.6 update	40
4.16.4 Dokumentation der Datenelemente	40
4.16.4.1 AttrEdit	40
4.16.4.2 Connection	40
4.16.4.3 EditorTuple	40
4.16.4.4 NewTuple	40
4.16.4.5 Selector	41
4.16.4.6 SelectorID	41
4.16.4.7 StatusBar	41
4.16.4.8 WhatsThis	41
4.17 AttributeEditor::ValueListEntry Strukturreferenz	41
4.17.1 Dokumentation der Datenelemente	41

4.17.1.1	Key	41
4.17.1.2	Value	41
4.18	ViewEntry Strukturreferenz	41
4.18.1	Ausführliche Beschreibung	42
4.18.2	Dokumentation der Elementfunktionen	42
4.18.2.1	toHelp	42
4.18.2.2	toTitle	42
4.18.3	Dokumentation der Datenelemente	42
4.18.3.1	Description	42
4.18.3.2	Descriptions	42
4.18.3.3	Join	42
4.18.3.4	Joins	42
4.18.3.5	OrderBy	42
4.18.3.6	Select	42
4.18.3.7	Selects	42
4.18.3.8	Title	43
4.18.3.9	ViewName	43
4.19	ViewTable Strukturreferenz	43
4.19.1	Ausführliche Beschreibung	43
4.19.2	Dokumentation der Datenelemente	43
4.19.2.1	BaseTableName	43
4.19.2.2	MaxPrimaryKeyEntries	44
4.19.2.3	PixmapName	44
4.19.2.4	PrimaryKey	44
4.19.2.5	View	44
4.19.2.6	Views	44
5	Datei-Dokumentation	45
5.1	attributeeditor.cc-Dateireferenz	45
5.2	attributeeditor.h-Dateireferenz	45
5.3	generator.cc-Dateireferenz	45
5.3.1	Dokumentation der Funktionen	48
5.3.1.1	begin_transaction	48
5.3.1.2	commit_work	48

5.3.1.3	delete_student_data	48
5.3.1.4	delete_student_data	48
5.3.1.5	dp_exams_test	48
5.3.1.6	dp_exams_thesis	48
5.3.1.7	even	48
5.3.1.8	generate_exams_result	48
5.3.1.9	generate_exams_thesis_result	48
5.3.1.10	generate_personal_data	48
5.3.1.11	generate_student	48
5.3.1.12	generate_student_semester	48
5.3.1.13	load_all_lists	48
5.3.1.14	load_list	48
5.3.1.15	main	48
5.3.1.16	odd	49
5.3.1.17	qualies_test	49
5.3.1.18	rand_element	49
5.3.1.19	rand_int	49
5.3.1.20	Randomize	49
5.3.1.21	vacuum_database	49
5.3.1.22	w_e_s	49
5.3.1.23	w_q_s	49
5.3.1.24	write_exams_closing	49
5.3.1.25	write_exams_comment	49
5.3.1.26	write_exams_counter	49
5.3.1.27	write_exams_date	49
5.3.1.28	write_exams_entry_date	49
5.3.1.29	write_exams_intro	49
5.3.1.30	write_exams_pers_id	49
5.3.1.31	write_exams_stud_id	49
5.3.1.32	write_exams_thesis_dates	49
5.3.1.33	write_exams_thesis_entry_date	49
5.3.1.34	write_exams_thesis_intro	49
5.3.1.35	write_exams_thesis_theme	49
5.3.1.36	write_exams_typ	49

5.3.1.37	write_qualies_closing	49
5.3.1.38	write_qualies_comment	49
5.3.1.39	write_qualies_date	49
5.3.1.40	write_qualies_date_late	50
5.3.1.41	write_qualies_intro	50
5.3.1.42	write_qualies_pers_id	50
5.3.1.43	write_qualies_stud_id	50
5.3.1.44	write_qualies_title	50
5.3.1.45	write_qualies_typ	50
5.3.2	Variablen-Dokumentation	50
5.3.2.1	ABI_AGE	50
5.3.2.2	BIRTH_START	50
5.3.2.3	BIRTH_VAR	50
5.3.2.4	DP_HD_ARBEIT	50
5.3.2.5	DP_HD_DIPLOM	50
5.3.2.6	DP_HD_INF_A	50
5.3.2.7	DP_HD_INF_B	50
5.3.2.8	DP_HD_INF_C	50
5.3.2.9	DP_HD_NF	50
5.3.2.10	DP_VD_DIPLOM	50
5.3.2.11	DP_VD_INF_A	50
5.3.2.12	DP_VD_INF_B	50
5.3.2.13	DP_VD_MATHE	50
5.3.2.14	DP_VD_NF	50
5.3.2.15	IMMAT_VAR	50
5.3.2.16	list_names_family	50
5.3.2.17	list_names_female	50
5.3.2.18	list_names_male	51
5.3.2.19	list_names_str_typ	51
5.3.2.20	list_names_street	51
5.3.2.21	list_names_towns	51
5.3.2.22	MAX_DIPL_TRY	51
5.3.2.23	MAX_DP_EXAMS	51
5.3.2.24	MAX_ELEMENTS	51

5.3.2.25	MAX_HD_TRY	51
5.3.2.26	MAX_PERS_ID	51
5.3.2.27	MAX_QUALIES	51
5.3.2.28	MAX_STUD_ID	51
5.3.2.29	MAX_VD_TRY	51
5.3.2.30	MIN_PERS_ID	51
5.3.2.31	MIN_STUD_ID	51
5.3.2.32	NO	51
5.3.2.33	NOT_PASSED	51
5.3.2.34	Q_DIPLOM	51
5.3.2.35	Q_HZB	51
5.3.2.36	Q_INF_1	51
5.3.2.37	Q_INF_2	51
5.3.2.38	Q_INF_3	51
5.3.2.39	Q_INF_4	51
5.3.2.40	Q_INF_PC	51
5.3.2.41	Q_INF_PP	51
5.3.2.42	Q_INF_PS	52
5.3.2.43	Q_INF_SA	52
5.3.2.44	Q_INF_SB	52
5.3.2.45	Q_M_IR_1	52
5.3.2.46	Q_M_IR_2	52
5.3.2.47	Q_M_LA_1	52
5.3.2.48	Q_M_LA_2	52
5.3.2.49	Q_M_PMWR	52
5.3.2.50	Q_V_DIPL	52
5.3.2.51	SEM_LIMIT	52
5.3.2.52	SEX_QUOTA	52
5.3.2.53	SOM_SEM	52
5.3.2.54	WIN_SEM	52
5.3.2.55	YEAR_NOW	52
5.3.2.56	YES	52
5.3.2.57	YES_PASSED	52
5.4	logindialog.cc-Dateireferenz	52

5.5	logindialog.h-Dateireferenz	52
5.6	paclient.cc-Dateireferenz	53
5.7	paclient.h-Dateireferenz	53
5.8	padb.cc-Dateireferenz	53
5.8.1	Dokumentation der Funktionen	53
5.8.1.1	main	53
5.9	paviews.cc-Dateireferenz	54
5.9.1	Variablen-Dokumentation	55
5.9.1.1	Adressen_DescriptionEntries	55
5.9.1.2	Adressen_SelectEntries	55
5.9.1.3	Adressen_ViewEntries	55
5.9.1.4	Adressen_ViewTable	56
5.9.1.5	Adresstypen_DescriptionEntries	56
5.9.1.6	Adresstypen_ViewEntries	56
5.9.1.7	Adresstypen_ViewTable	56
5.9.1.8	Diplomarbeiten_DescriptionEntries	57
5.9.1.9	Diplomarbeiten_SelectEntries	57
5.9.1.10	Diplomarbeiten_ViewEntries	57
5.9.1.11	Diplomarbeiten_ViewTable	57
5.9.1.12	Leistungsnachweise_DescriptionEntries	58
5.9.1.13	Leistungsnachweise_SelectEntries	58
5.9.1.14	Leistungsnachweise_ViewEntries	58
5.9.1.15	Leistungsnachweise_ViewTable	59
5.9.1.16	PADB_OptimizeTable	59
5.9.1.17	PADB_OptimizeTableEntries	59
5.9.1.18	Pruefer_DescriptionEntries	59
5.9.1.19	Pruefer_JoinEntries	60
5.9.1.20	Pruefer_ViewEntries	60
5.9.1.21	Pruefer_ViewTable	60
5.9.1.22	Pruefungen_DescriptionEntries	61
5.9.1.23	Pruefungen_SelectEntries	61
5.9.1.24	Pruefungen_ViewEntries	61
5.9.1.25	Pruefungen_ViewTable	61
5.9.1.26	Pruefungstatus_DescriptionEntries	62

5.9.1.27	Pruefungsstatus_ViewEntries	62
5.9.1.28	Pruefungsstatus_ViewTable	62
5.9.1.29	Pruefungstypen_DescriptionEntries	62
5.9.1.30	Pruefungstypen_ViewEntries	63
5.9.1.31	Pruefungstypen_ViewTable	63
5.9.1.32	Qualifikattypen_DescriptionEntries	63
5.9.1.33	Qualifikattypen_ViewEntries	63
5.9.1.34	Qualifikattypen_ViewTable	64
5.9.1.35	Semestertext_DescriptionEntries	64
5.9.1.36	Semestertext_ViewEntries	64
5.9.1.37	Semestertext_ViewTable	64
5.9.1.38	Statistiken_DescriptionEntries	65
5.9.1.39	Statistiken_ViewEntries	65
5.9.1.40	Statistiken_ViewTable	65
5.9.1.41	Studenten_DescriptionEntries	65
5.9.1.42	Studenten_JoinEntries	65
5.9.1.43	Studenten_ViewEntries	66
5.9.1.44	Studenten_ViewTable	66
5.9.1.45	Studientypen_DescriptionEntries	66
5.9.1.46	Studientypen_ViewEntries	66
5.9.1.47	Studientypen_ViewTable	66
5.9.1.48	Studium_DescriptionEntries	67
5.9.1.49	Studium_SelectEntries	67
5.9.1.50	Studium_ViewEntries	67
5.9.1.51	Studium_ViewTable	67
5.10	paviews.h-Dateireferenz	68
5.10.1	Variablen-Dokumentation	68
5.10.1.1	PADB_OptimizeTable	68
5.10.1.2	Pruefer_ViewTable	68
5.10.1.3	Pruefungstypen_ViewTable	68
5.10.1.4	Qualifikattypen_ViewTable	69
5.10.1.5	Statistiken_ViewTable	69
5.10.1.6	Studenten_ViewTable	69
5.11	sqlconnection.cc-Dateireferenz	69

5.12	sqlconnection.h-Dateireferenz	69
5.13	sqlconnectioninterface.h-Dateireferenz	69
5.14	sqlexception.cc-Dateireferenz	69
5.14.1	Dokumentation der Funktionen	70
5.14.1.1	operator<<	70
5.15	sqlexception.h-Dateireferenz	70
5.16	sqlmessages.cc-Dateireferenz	70
5.16.1	Dokumentation der Funktionen	70
5.16.1.1	sqlError	70
5.16.1.2	sqlWarning	71
5.17	sqlmessages.h-Dateireferenz	71
5.17.1	Dokumentation der Funktionen	71
5.17.1.1	sqlError	71
5.17.1.2	sqlWarning	71
5.18	sqlmonitorinterface.h-Dateireferenz	71
5.19	system.h-Dateireferenz	72
5.19.1	Makro-Dokumentation	72
5.19.1.1	_GNU_SOURCE	72
5.19.1.2	_THREAD_SAFE	72
5.19.1.3	CPU_BYTEORDER	72
5.19.1.4	USE_PTHREADS	72
5.19.2	Dokumentation der benutzerdefinierten Typen	73
5.19.2.1	card16	73
5.19.2.2	card32	73
5.19.2.3	card64	73
5.19.2.4	card8	73
5.19.2.5	cardinal	73
5.19.2.6	int16	73
5.19.2.7	int32	73
5.19.2.8	int64	73
5.19.2.9	int8	73
5.19.2.10	integer	73
5.19.2.11	sbyte	74
5.19.2.12	ubyte	74

5.20	tableviewer.cc-Dateireferenz	74
5.20.1	Dokumentation der Funktionen	74
5.20.1.1	trim	74
5.21	tableviewer.h-Dateireferenz	74
5.22	textmonitor.cc-Dateireferenz	74
5.23	textmonitor.h-Dateireferenz	75
5.24	tupleeditor.cc-Dateireferenz	75
5.25	tupleeditor.h-Dateireferenz	75

Kapitel 1

Klassen-Verzeichnis

1.1 Klassenhierarchie

Die Liste der Ableitungen ist -mit Einschränkungen- alphabetisch sortiert:

AttributeEditor	7
DescriptionEntry	10
JoinEntry	11
list_t	12
LoginDialog	13
OptimizeTable	14
PAClient	15
SelectEntry	17
SQLConnectionInterface	23
SQLConnection	18
SQLException	27
SQLMonitorInterface	29
TextMonitor	35
TableView	30
Tuple	37
TupleEditor	38
AttributeEditor::ValueListEntry	41
ViewEntry	41
ViewTable	43

Kapitel 2

Klassen-Verzeichnis

2.1 Auflistung der Klassen

Hier folgt die Aufzählung aller Klassen, Strukturen, Varianten und Schnittstellen mit einer Kurzbeschreibung:

AttributeEditor	7
DescriptionEntry	
Description Entry	10
JoinEntry	
Join Entry	11
list_t	12
LoginDialog	
Login Dialog	13
OptimizeTable	
Optimize Table	14
PAClient	
PAClient	15
SelectEntry	
Select Entry	17
SQLConnection	
SQL Connection	18
SQLConnectionInterface	
SQL Connection Interface	23
SQLException	
SQL Exception	27
SQLMonitorInterface	
SQL Monitor Interface	29
TableViewer	
Table Viewer	30
TextMonitor	
Text Monitor	35
Tuple	
Tuple	37

TupleEditor		
Tuple Editor	38
AttributeEditor::ValueListEntry	41
ViewEntry		
View Entry	41
ViewTable		
View Table	43

Kapitel 3

Datei-Verzeichnis

3.1 Auflistung der Dateien

Hier folgt die Aufzählung aller Dateien mit einer Kurzbeschreibung:

attributeeditor.cc	45
attributeeditor.h	45
generator.cc	45
logindialog.cc	52
logindialog.h	52
paclient.cc	53
paclient.h	53
padb.cc	53
paviews.cc	54
paviews.h	68
sqlconnection.cc	69
sqlconnection.h	69
sqlconnectioninterface.h	69
sqlexception.cc	69
sqlexception.h	70
sqlmessages.cc	70
sqlmessages.h	71
sqlmonitorinterface.h	71
system.h	72
tableviewer.cc	74
tableviewer.h	74
textmonitor.cc	74
textmonitor.h	75
tupleeditor.cc	75
tupleeditor.h	75

Kapitel 4

Klassen-Dokumentation

4.1 AttributeEditor Klassenreferenz

```
#include <attributeeditor.h>
```

Klassen

- struct [ValueListEntry](#)

Öffentliche Slots

- void [returnPressed](#) ()
- void [clicked](#) ()
- void [activated](#) (int number)
- void [valueSelected](#) ([TableView](#) *viewer, const bool selected, const QString &selection)

Signale

- void [userUpdate](#) ()

Öffentliche Methoden

- [AttributeEditor](#) ([SQLConnectionInterface](#) *connection, const [ViewEntry](#) *view, const QString &attribute, const bool editable=true, const QString &value=QString::null, QWidget *parent=NULL, const char *name=NULL)
- [~AttributeEditor](#) ()
- QString [getValue](#) () const
- void [setValue](#) (const QString &value)
- void [mark](#) ()

Private Methoden

- QString [fetchTuple](#) (const QString &tableName, const QString &attributes, const QString &key, const QString &keyValue)
- QList< [ValueListEntry](#) > * [fetchValueList](#) (const QString &tableName, const QString &attributes, const QString &key)

Private Attribute

- [SQLConnectionInterface](#) * [Connection](#)
- const [ViewEntry](#) * [View](#)
- QString [Attribute](#)
- QLineEdit * [LineEdit](#)
- QLabel * [Label](#)
- QComboBox * [ComboBox](#)
- QPushButton * [Button](#)
- [TableView](#) * [Selector](#)
- const [ViewTable](#) * [SelectView](#)
- QString [SelectValue](#)
- QString [SelectAttribute](#)
- QString [SelectInfoAttributes](#)
- QList< [ValueListEntry](#) > * [SelectList](#)

4.1.1 Beschreibung der Konstruktoren und Destruktoren

- 4.1.1.1 **AttributeEditor::AttributeEditor** ([SQLConnectionInterface](#) * *connection*, const [ViewEntry](#) * *view*, const QString & *attribute*, const bool *editable* = true, const QString & *value* = QString::null, QWidget * *parent* = NULL, const char * *name* = NULL)

Constructor.

Parameter

<i>connection</i>	Datenbank-Verbindung.
<i>view</i>	ViewEntry des Attributes.
<i>editable</i>	Ist Wert editierbar?
<i>value</i>	Startwert.
<i>parent</i>	Parent QWidget; default: NULL.
<i>name</i>	Widget-Name; default: NULL.

- 4.1.1.2 **AttributeEditor::~AttributeEditor** ()

Destructor.

4.1.2 Dokumentation der Elementfunktionen

4.1.2.1 void AttributeEditor::activated (int *number*) [slot]

Qt-Slot: Wert ComboBox gewählt.

4.1.2.2 void AttributeEditor::clicked () [slot]

Qt-Slot: "Auswahl..."-Button angeklickt.

4.1.2.3 QString AttributeEditor::fetchTuple (const QString & *tableName*, const QString & *attributes*, const QString & *key*, const QString & *keyValue*) [private]

4.1.2.4 QList< AttributeEditor::ValueListEntry > * AttributeEditor::fetchValueList (const QString & *tableName*, const QString & *attributes*, const QString & *key*) [private]

4.1.2.5 QString AttributeEditor::getValue () const

Wert des Attributes zurückgeben.

Rückgabe

Attributwert.

4.1.2.6 void AttributeEditor::mark ()

Eingabe markieren (z.B. nach Update-Fehler).

4.1.2.7 void AttributeEditor::returnPressed () [slot]

Qt-Slot: Return in LineEdit gedrückt.

4.1.2.8 void AttributeEditor::setValue (const QString & *value*)

Wert des Attributes setzen

Parameter

<i>value</i>	Attributwert.
--------------	---------------

4.1.2.9 void AttributeEditor::userUpdate () [signal]

Qt-Signal: Wert-Eingabe in LineEdit mit Return beendet.

4.1.2.10 `void AttributeEditor::valueSelected (TableView * viewer, const bool selected, const QString & selection) [slot]`

Qt-Slot: Selektion durchgeführt.

4.1.3 Dokumentation der Datenelemente

4.1.3.1 `QString AttributeEditor::Attribute [private]`

4.1.3.2 `QPushButton* AttributeEditor::Button [private]`

4.1.3.3 `QComboBox* AttributeEditor::ComboBox [private]`

4.1.3.4 `SQLConnectionInterface* AttributeEditor::Connection [private]`

4.1.3.5 `QLabel* AttributeEditor::Label [private]`

4.1.3.6 `QLineEdit* AttributeEditor::LineEdit [private]`

4.1.3.7 `QString AttributeEditor::SelectAttribute [private]`

4.1.3.8 `QString AttributeEditor::SelectInfoAttributes [private]`

4.1.3.9 `QList<ValueListEntry>* AttributeEditor::SelectList [private]`

4.1.3.10 `TableView* AttributeEditor::Selector [private]`

4.1.3.11 `QString AttributeEditor::SelectValue [private]`

4.1.3.12 `const ViewTable* AttributeEditor::SelectView [private]`

4.1.3.13 `const ViewEntry* AttributeEditor::View [private]`

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

- [attributeeditor.h](#)
- [attributeeditor.cc](#)

4.2 DescriptionEntry Strukturreferenz

Description Entry.

```
#include <paviews.h>
```

Öffentliche Attribute

- const char * [Field](#)
- const char * [Title](#)
- const char * [Help](#)

4.2.1 Ausführliche Beschreibung

Description Entry.

Diese Struktur speichert Informationen über Felder einer Tabelle: Textumsetzung - Datenbank-Schema <-> Bildschirm und Hilfetexte. gegeben durch Zeiger auf [View-Table](#).

Autor

Thomas Dreibholz dreibh@iem.uni-due.de

Version

1.0

4.2.2 Dokumentation der Datenelemente

4.2.2.1 const char* [DescriptionEntry::Field](#)

4.2.2.2 const char* [DescriptionEntry::Help](#)

4.2.2.3 const char* [DescriptionEntry::Title](#)

Die Dokumentation für diese Struktur wurde erzeugt aufgrund der Datei:

- [paviews.h](#)

4.3 JoinEntry Strukturreferenz

Join Entry.

```
#include <paviews.h>
```

Öffentliche Attribute

- const char * [JoinAttributeR1](#)
- const char * [JoinAttributeR2](#)
- const char * [JoinTitle](#)
- const [ViewTable](#) * [View](#)

4.3.1 Ausführliche Beschreibung

Join Entry.

Diese Struktur speichert Informationen über einen Join der Tabelle mit einer anderen Tabelle, gegeben durch Zeiger auf [ViewTable](#).

Autor

Thomas Dreibholz dreibh@iem.uni-due.de

Version

1.0

4.3.2 Dokumentation der Datenelemente

4.3.2.1 `const char* JoinEntry::JoinAttributeR1`

4.3.2.2 `const char* JoinEntry::JoinAttributeR2`

4.3.2.3 `const char* JoinEntry::JoinTitle`

4.3.2.4 `const ViewTable* JoinEntry::View`

Die Dokumentation für diese Struktur wurde erzeugt aufgrund der Datei:

- [paviews.h](#)

4.4 list_t Strukturreferenz

Öffentliche Attribute

- `char** value`
- `int no_of_elements`

4.4.1 Dokumentation der Datenelemente

4.4.1.1 `int list_t::no_of_elements`

4.4.1.2 `char** list_t::value`

Die Dokumentation für diese Struktur wurde erzeugt aufgrund der Datei:

- [generator.cc](#)

4.5 LoginDialog Klassenreferenz

Login Dialog.

```
#include <logindialog.h>
```

Öffentliche Methoden

- [LoginDialog](#) (const QString &defaultServer, const QString &defaultDatabase, const QString &defaultUser, QWidget *parent=NULL, const char *name=NULL)

Öffentliche, statische Methoden

- static PgDatabase * [login](#) (const QString &defaultServer, const QString &defaultDatabase, const QString &defaultUser)

Private Attribute

- QLineEdit * [LineEdit](#) [4]

4.5.1 Ausführliche Beschreibung

Login Dialog.

[LoginDialog](#) ist ein Qt-Widget für den Login am Datenbank-Server.

Autor

Thomas Dreibholz dreibh@iem.uni-due.de

Version

1.0

4.5.2 Beschreibung der Konstruktoren und Destruktoren

4.5.2.1 [LoginDialog::LoginDialog](#) (const QString & *defaultServer*, const QString & *defaultDatabase*, const QString & *defaultUser*, QWidget * *parent* = NULL, const char * *name* = NULL)

Constructor.

Parameter

<i>default-Server</i>	Vorgegebener Server.
<i>default-Database</i>	Vorgegebener Datenbank-Name.
<i>defaultUser</i>	Vorgegebener Benutzer.
<i>parent</i>	Parent QWidget; default: NULL.
<i>name</i>	Widget-Name; default: NULL.

4.5.3 Dokumentation der Elementfunktionen

4.5.3.1 PgDatabase * LoginDialog::login (const QString & *defaultServer*, const QString & *defaultDatabase*, const QString & *defaultUser*) [static]

Login durchführen und Verbindungsobjekt für die Datenbank zurückgeben.

Parameter

<i>default-Server</i>	Vorgegebener Server.
<i>default-Database</i>	Vorgegebener Datenbank-Name.
<i>defaultUser</i>	Vorgegebener Benutzer.

Rückgabe

PgDatabase-Objekt.

4.5.4 Dokumentation der Datenelemente

4.5.4.1 QLineEdit* LoginDialog::LineEdit[4] [private]

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

- [logindialog.h](#)
- [logindialog.cc](#)

4.6 OptimizeTable Strukturreferenz

Optimize Table.

```
#include <paviews.h>
```

Öffentliche Attribute

- const [cardinal Tables](#)

- `const char ** Table`

4.6.1 Ausführliche Beschreibung

Optimize Table.

Diese Struktur enthält alle Tabellennamen für die Optimierungs-Funktion.

Autor

Thomas Dreibholz dreibh@iem.uni-due.de

Version

1.0

4.6.2 Dokumentation der Datenelemente

4.6.2.1 `const char** OptimizeTable::Table`

4.6.2.2 `const cardinal OptimizeTable::Tables`

Die Dokumentation für diese Struktur wurde erzeugt aufgrund der Datei:

- [paviews.h](#)

4.7 PAClient Klassenreferenz

[PAClient](#).

```
#include <paclient.h>
```

Öffentliche Slots

- void [information](#) ()
- void [quit](#) ()
- void [optimize](#) ()
- void [openMonitor](#) ()
- void [closeMonitor](#) ()

Öffentliche Methoden

- [PAClient](#) ([SQLConnectionInterface](#) *connection, [TextMonitor](#) *monitor=NULL, [QWidget](#) *parent=NULL, `const char *name=NULL`)
- [~PAClient](#) ()

Private Attribute

- [SQLConnectionInterface](#) * [Connection](#)
- [TextMonitor](#) * [Monitor](#)

4.7.1 Ausführliche Beschreibung

[PAClient](#).

[PAClient](#) ist ein Qt-Widget für das Hauptfenster des Prüfungsamt-Clients.

Autor

Thomas Dreiboldz dreibh@iem.uni-due.de

Version

1.0

4.7.2 Beschreibung der Konstruktoren und Destruktoren

4.7.2.1 **PAClient::PAClient** ([SQLConnectionInterface](#) * *connection*, [TextMonitor](#) * *monitor* = NULL, [QWidget](#) * *parent* = NULL, const char * *name* = NULL)

Constructor.

Parameter

<i>connection</i>	Datenbank-Verbindung.
<i>monitor</i>	SQL-Monitor.
<i>parent</i>	Parent QWidget; default: NULL.
<i>name</i>	Widget-Name; default: NULL.

4.7.2.2 **PAClient::~~PAClient** ()

Destructor.

4.7.3 Dokumentation der Elementfunktionen

4.7.3.1 void **PAClient::closeMonitor** () [slot]

Qt-Slot: SQL-Monitor schließen.

4.7.3.2 void **PAClient::information** () [slot]

Qt-Slot: Informationsfenster anzeigen.

4.7.3.3 void **PAClient::openMonitor**() [slot]

Qt-Slot: SQL-Monitor öffnen.

4.7.3.4 void **PAClient::optimize**() [slot]

Qt-Slot: Optimieren.

4.7.3.5 void **PAClient::quit**() [slot]

Qt-Slot: Beenden.

4.7.4 Dokumentation der Datenelemente

4.7.4.1 **SQLConnectionInterface* PAClient::Connection** [private]

4.7.4.2 **TextMonitor* PAClient::Monitor** [private]

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

- [paclient.h](#)
- [paclient.cc](#)

4.8 SelectEntry Strukturreferenz

Select Entry.

```
#include <paviews.h>
```

Öffentliche Attribute

- const char * [SelectAttributeR1](#)
- const char * [SelectAttributeR2](#)
- const char * [SelectInfoAttributes](#)
- const bool [UseComboBox](#)
- const [ViewTable](#) * [View](#)

4.8.1 Ausführliche Beschreibung

Select Entry.

Diese Struktur speichert Informationen über eine Selektion eines Attributes aus einer anderen Tabelle, gegeben durch Zeiger auf [ViewTable](#).

Autor

Thomas Dreibholz dreibh@iem.uni-due.de

Version

1.0

4.8.2 Dokumentation der Datenelemente

4.8.2.1 `const char* SelectEntry::SelectAttributeR1`

4.8.2.2 `const char* SelectEntry::SelectAttributeR2`

4.8.2.3 `const char* SelectEntry::SelectInfoAttributes`

4.8.2.4 `const bool SelectEntry::UseComboBox`

4.8.2.5 `const ViewTable* SelectEntry::View`

Die Dokumentation für diese Struktur wurde erzeugt aufgrund der Datei:

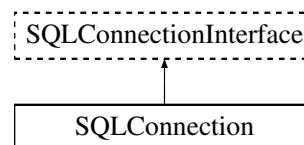
- [paviews.h](#)

4.9 SQLConnection Klassenreferenz

SQL Connection.

```
#include <sqlconnection.h>
```

Klassendiagramm für SQLConnection:

**Öffentliche Methoden**

- `SQLConnection` (`PgDatabase *database`, `SQLMonitorInterface *monitor=NULL`)
- void `beginTransaction` ()
- void `commitTransaction` ()
- void `rollbackTransaction` ()
- void `execute` (`const char *command`)
- void `createCursor` (`const char *cursorName`, `const char *command`)

- void `fetchCursor` (const char *cursorName, const char *what)
- void `deleteCursor` (const char *cursorName)
- cardinal `getTuples` ()
- cardinal `getFields` ()
- const char * `getField` (cardinal col)
- const char * `getValue` (cardinal row, cardinal col)
- void `optimize` (const char *tableName)
- SQLMonitorInterface * `getMonitor` () const
- void `setMonitor` (SQLMonitorInterface *monitor)

Private Methoden

- bool `executeCommandOk` (const char *command)
- bool `executeTuplesOk` (const char *command)

Private Attribute

- PgDatabase * `Database`
- SQLMonitorInterface * `Monitor`
- bool `Transaction`

4.9.1 Ausführliche Beschreibung

SQL Connection.

`SQLConnection` ist die Verbindung zur Datenbank.

Autor

Thomas Dreibholz dreibh@iem.uni-due.de

Version

1.0

4.9.2 Beschreibung der Konstruktoren und Destruktoren

4.9.2.1 SQLConnection::SQLConnection (PgDatabase * *database*, SQLMonitorInterface * *monitor* = NULL)

Constructor.

Parameter

<i>database</i>	PgDatabase-Objekt.
<i>monitor</i>	SQLMonitorInterface-Objekt.

4.9.3 Dokumentation der Elementfunktionen

4.9.3.1 `void SQLConnection::beginTransaction () [virtual]`

Implementation von [SQLConnectionInterface](#).

Siehe auch

[SQLConnectionInterface::beginTransaction](#)

Implementiert [SQLConnectionInterface](#).

4.9.3.2 `void SQLConnection::commitTransaction () [virtual]`

Implementation von [SQLConnectionInterface](#).

Siehe auch

[SQLConnectionInterface::commitTransaction](#)

Implementiert [SQLConnectionInterface](#).

4.9.3.3 `void SQLConnection::createCursor (const char * cursorName, const char * command) [virtual]`

Implementation von [SQLConnectionInterface](#).

Siehe auch

[SQLConnectionInterface::createCursor](#)

Implementiert [SQLConnectionInterface](#).

4.9.3.4 `void SQLConnection::deleteCursor (const char * cursorName) [virtual]`

Implementation von [SQLConnectionInterface](#).

Siehe auch

[SQLConnectionInterface::deleteCursor](#)

Implementiert [SQLConnectionInterface](#).

4.9.3.5 `void SQLConnection::execute (const char * command) [virtual]`

Implementation von [SQLConnectionInterface](#).

Siehe auch

[SQLConnectionInterface::execute](#)

Implementiert [SQLConnectionInterface](#).

4.9.3.6 `bool SQLConnection::executeCommandOk (const char * command)`
[private]

4.9.3.7 `bool SQLConnection::executeTuplesOk (const char * command)`
[private]

4.9.3.8 `void SQLConnection::fetchCursor (const char * cursorName, const char * what)`
[virtual]

Implementation von [SQLConnectionInterface](#).

Siehe auch

[SQLConnectionInterface::fetchCursor](#)

Implementiert [SQLConnectionInterface](#).

4.9.3.9 `const char * SQLConnection::getField (cardinal col)` [virtual]

Implementation von [SQLConnectionInterface](#).

Siehe auch

[SQLConnectionInterface::getField](#)

Implementiert [SQLConnectionInterface](#).

4.9.3.10 `cardinal SQLConnection::getFields ()` [virtual]

Implementation von [SQLConnectionInterface](#).

Siehe auch

[SQLConnectionInterface::getFields](#)

Implementiert [SQLConnectionInterface](#).

4.9.3.11 `SQLMonitorInterface * SQLConnection::getMonitor ()` const
[virtual]

Implementation von [SQLConnectionInterface](#).

Siehe auch

[SQLConnectionInterface::getMonitor](#)

Implementiert [SQLConnectionInterface](#).

4.9.3.12 cardinal SQLConnection::getTuples () [virtual]

Implementation von [SQLConnectionInterface](#).

Siehe auch

[SQLConnectionInterface::getTuples](#)

Implementiert [SQLConnectionInterface](#).

4.9.3.13 const char * SQLConnection::getValue (cardinal row, cardinal col)
[virtual]

Implementation von [SQLConnectionInterface](#).

Siehe auch

[SQLConnectionInterface::getValue](#)

Implementiert [SQLConnectionInterface](#).

4.9.3.14 void SQLConnection::optimize (const char * tableName) [virtual]

Implementation von [SQLConnectionInterface](#).

Siehe auch

[SQLConnectionInterface::optimize](#)

Implementiert [SQLConnectionInterface](#).

4.9.3.15 void SQLConnection::rollbackTransaction () [virtual]

Implementation von [SQLConnectionInterface](#).

Siehe auch

[SQLConnectionInterface::rollbackTransaction](#)

Implementiert [SQLConnectionInterface](#).

4.9.3.16 `void SQLConnection::setMonitor (SQLMonitorInterface * monitor)`
`[virtual]`

Implementation von [SQLConnectionInterface](#).

Siehe auch

[SQLConnectionInterface::setMonitor](#)

Implementiert [SQLConnectionInterface](#).

4.9.4 Dokumentation der Datenelemente

4.9.4.1 `PgDatabase* SQLConnection::Database` `[private]`

4.9.4.2 `SQLMonitorInterface* SQLConnection::Monitor` `[private]`

4.9.4.3 `bool SQLConnection::Transaction` `[private]`

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

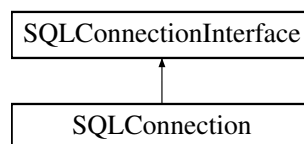
- [sqlconnection.h](#)
- [sqlconnection.cc](#)

4.10 SQLConnectionInterface Klassenreferenz

SQL Connection Interface.

```
#include <sqlconnectioninterface.h>
```

Klassendiagramm für SQLConnectionInterface:



Öffentliche Methoden

- virtual void [beginTransaction](#) ()=0
- virtual void [commitTransaction](#) ()=0
- virtual void [rollbackTransaction](#) ()=0
- virtual void [execute](#) (const char *command)=0
- virtual void [createCursor](#) (const char *cursorName, const char *command)=0
- virtual void [fetchCursor](#) (const char *cursorName, const char *what="ALL")=0

- virtual void [deleteCursor](#) (const char *cursorName)=0
- virtual [cardinal getTuples](#) ()=0
- virtual [cardinal getFields](#) ()=0
- virtual const char * [getField](#) ([cardinal col](#))=0
- virtual const char * [getValue](#) ([cardinal row](#), [cardinal col](#))=0
- virtual void [optimize](#) (const char *tableName)=0
- virtual [SQLMonitorInterface](#) * [getMonitor](#) () const =0
- virtual void [setMonitor](#) ([SQLMonitorInterface](#) *monitor)=0

4.10.1 Ausführliche Beschreibung

SQL Connection Interface.

[SQLConnectionInterface](#) ist ein Interface für eine Verbindung zu einer SQL-Datenbank.

Autor

Thomas Dreibholz dreibh@iem.uni-due.de

Version

1.0

4.10.2 Dokumentation der Elementfunktionen

4.10.2.1 `virtual void SQLConnectionInterface::beginTransaction () [pure virtual]`

Beginn einer Transaktion. Bei Fehler wird eine SQL-Exception aufgeworfen.

Implementiert in [SQLConnection](#).

4.10.2.2 `virtual void SQLConnectionInterface::commitTransaction () [pure virtual]`

Ende einer Transaktion mit Commit. Bei Fehler wird eine SQL-Exception aufgeworfen und ein Rollback gesendet.

Implementiert in [SQLConnection](#).

4.10.2.3 `virtual void SQLConnectionInterface::createCursor (const char * cursorName, const char * command) [pure virtual]`

Erstellen eines neuen SQL-Cursors für einen gegebenen Befehl. Bei Fehler wird eine SQL-Exception aufgeworfen.

Parameter

<i>cursorName</i>	Name des Cursors.
<i>command</i>	SQL-Befehl.

Implementiert in [SQLConnection](#).

4.10.2.4 `virtual void SQLConnectionInterface::deleteCursor (const char * cursorName) [pure virtual]`

Entfernen eines SQL-Cursors. Bei Fehler wird eine SQL-Exception aufgeworfen.

Parameter

<i>cursorName</i>	Name des Cursors.
-------------------	-------------------

Implementiert in [SQLConnection](#).

4.10.2.5 `virtual void SQLConnectionInterface::execute (const char * command) [pure virtual]`

Ausführung einer SQL-Anweisung. Bei Fehler wird eine SQL-Exception aufgeworfen.

Parameter

<i>command</i>	SQL-Befehl.
----------------	-------------

Implementiert in [SQLConnection](#).

4.10.2.6 `virtual void SQLConnectionInterface::fetchCursor (const char * cursorName, const char * what = "ALL") [pure virtual]`

Fetch-Ausführung für einen gegebenen SQL-Cursor. Bei Fehler wird eine SQL-Exception aufgeworfen.

Parameter

<i>cursorName</i>	Name des Cursors.
<i>what</i>	Beschreibung des Fetch-Bereichs, z.B. "ALL".

Implementiert in [SQLConnection](#).

4.10.2.7 `virtual const char* SQLConnectionInterface::getField (cardinal col) [pure virtual]`

Attributnamen für gegebene Spalte zurückgeben.

Parameter

<i>col</i>	Spalten-Nummer.
------------	-----------------

Rückgabe

Attributname.

Implementiert in [SQLConnection](#).

4.10.2.8 `virtual cardinal SQLConnectionInterface::getFields () [pure virtual]`

Anzahl der Attribute im Cursor zurückgeben.

Rückgabe

Anzahl der Attribute.

Implementiert in [SQLConnection](#).

4.10.2.9 `virtual SQLMonitorInterface* SQLConnectionInterface::getMonitor () const [pure virtual]`

Zeige auf [SQLMonitorInterface](#) zurückgeben.

Rückgabe

[SQLMonitorInterface](#).

Implementiert in [SQLConnection](#).

4.10.2.10 `virtual cardinal SQLConnectionInterface::getTuples () [pure virtual]`

Anzahl der Tupel im Cursor zurückgeben.

Rückgabe

Anzahl der Tupel.

Implementiert in [SQLConnection](#).

4.10.2.11 `virtual const char* SQLConnectionInterface::getValue (cardinal row, cardinal col) [pure virtual]`

Wert in gegebener Zeile und Spalte zurückgeben.

Parameter

<i>row</i>	Zeilen-Nummer.
<i>col</i>	Spalten-Nummer.

Rückgabe

Wert

Implementiert in [SQLConnection](#).

4.10.2.12 `virtual void SQLConnectionInterface::optimize (const char * tableName)`
[pure virtual]

Tabelle optimieren. Bei Fehler wird eine SQL-Exception aufgeworfen.

Parameter

<i>tableName</i>	Tabellen-Name.
------------------	----------------

Implementiert in [SQLConnection](#).

4.10.2.13 `virtual void SQLConnectionInterface::rollbackTransaction ()` [pure
virtual]

Ende einer Transaktion mit Rollback. Bei Fehler wird eine SQL-Exception aufgeworfen.

Implementiert in [SQLConnection](#).

4.10.2.14 `virtual void SQLConnectionInterface::setMonitor (SQLMonitorInterface *
monitor)` [pure virtual]

[SQLMonitorInterface](#) setzen

Parameter

<i>monitor</i>	SQLMonitorInterface .
----------------	---------------------------------------

Implementiert in [SQLConnection](#).

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Datei:

- [sqlconnectioninterface.h](#)

4.11 SQLException Klassenreferenz

SQL Exception.

```
#include <sqlexception.h>
```

Öffentliche Methoden

- [SQLException](#) (const char *title=NULL)
- virtual [~SQLException](#) ()
- virtual const char * [toString](#) () const

Private Attribute

- char [Title](#) [[MaxTitleLength](#)]

Statische private Attribute

- static const [card32 MaxTitleLength](#) = 256

Freundbeziehungen

- ostream & [operator<<](#) (ostream &os, const [SQLException](#) e)

4.11.1 Ausführliche Beschreibung

SQL Exception.

[SQLException](#) ist ein Exception, die bei einem SQL-Fehler aufgeworfen wird.

Autor

Thomas Dreibholz dreibh@iem.uni-due.de

Version

1.0

4.11.2 Beschreibung der Konstruktoren und Destruktoren

4.11.2.1 [SQLException::SQLException](#) (const char * *title* = NULL)

Constructor.

Parameter

<i>title</i>	Exception-Titel.
--------------	------------------

4.11.2.2 `SQLException::~~SQLException ()` [virtual]

Destructor.

4.11.3 Dokumentation der Elementfunktionen

4.11.3.1 `const char * SQLException::toString () const` [virtual]

Exception-Titel zurückgeben.

Rückgabe

Titel

4.11.4 Freundbeziehungen und Funktionsdokumentation

4.11.4.1 `ostream& operator<<(ostream & os, const SQLException e)` [friend]

Ausgabe-Operator.

4.11.5 Dokumentation der Datenelemente

4.11.5.1 `const card32 SQLException::MaxTitleLength = 256` [static, private]

4.11.5.2 `char SQLException::Title[MaxTitleLength]` [private]

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

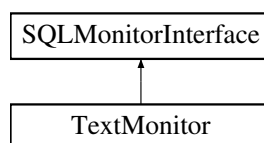
- [sqlexception.h](#)
- [sqlexception.cc](#)

4.12 SQLMonitorInterface Klassenreferenz

SQL Monitor Interface.

```
#include <sqlmonitorinterface.h>
```

Klassendiagramm für SQLMonitorInterface:



Öffentliche Methoden

- virtual void [write](#) (const char *string="")=0

4.12.1 Ausführliche Beschreibung

SQL Monitor Interface.

[SQLMonitorInterface](#) ist ein Interface für den SQL-Monitor.

Autor

Thomas Dreibholz dreibh@iem.uni-due.de

Version

1.0

4.12.2 Dokumentation der Elementfunktionen

4.12.2.1 virtual void [SQLMonitorInterface::write](#) (const char * *string* = " ") [pure virtual]

Ausgabe eines SQL-Befehls.

Parameter

<i>string</i>	Befehls-String.
---------------	-----------------

Implementiert in [TextMonitor](#).

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Datei:

- [sqlmonitorinterface.h](#)

4.13 TableViewer Klassenreferenz

Table Viewer.

```
#include <tableviewer.h>
```

Öffentliche Typen

- enum [TableViewMode](#) { [TVM_EditMode](#) = (1 << 0), [TVM_SelectMode](#) = (1 << 1), [TVM_ReadOnly](#) = (1 << 10) }

Öffentliche Slots

- void [search](#) (const QString &newText)
- void [setSortAttribute](#) (int section)
- void [selectView](#) (int selection)
- void [selectTuple](#) (QListViewItem *item)
- void [closeEditor](#) (TupleEditor *editor, const bool committed)
- void [newClicked](#) ()
- void [okayClicked](#) ()
- void [nullClicked](#) ()
- void [cancelClicked](#) ()

Signale

- void [doneSelection](#) (TableViewer *viewer, const bool selected, const QString &selected)

Öffentliche Methoden

- [TableViewer](#) (SQLConnectionInterface *connection, const [ViewTable](#) *viewTable, const [cardinal](#) mode=[TVM_EditMode](#), const QString &joinAttribute=QString::null, const QString &joinValue=QString::null, const QString &selectAttribute=QString::null, const QString &selectValue=QString::null, QWidget *parent=NULL, const char *name=NULL)
- [~TableViewer](#) ()
- void [loadTable](#) (const char *tableName=NULL, const char *orderBy=NULL)

Private Methoden

- void [closeEvent](#) (QCloseEvent *event)

Private Attribute

- [SQLConnectionInterface](#) * [Connection](#)
- [QWhatsThis](#) * [WhatsThis](#)
- [QLabel](#) * [StatusBar](#)
- [QDict](#)< [TupleEditor](#) > [TupleEditors](#)
- [cardinal](#) [Mode](#)
- const [ViewTable](#) * [Views](#)
- const [ViewEntry](#) * [View](#)
- [QString](#) [JoinAttribute](#)
- [QString](#) [JoinValue](#)
- [QString](#) [SelectAttribute](#)
- [QString](#) [SelectValue](#)
- [QListView](#) * [TableView](#)

- [QList< QString > Attributes](#)
- [QLabel * FindLabel](#)
- [QLineEdit * FindLineEdit](#)
- [cardinal FindSection](#)
- [bool SkipNextUpdate](#)
- [cardinal MinNotFoundIndex](#)

4.13.1 Ausführliche Beschreibung

Table Viewer.

[TableView](#) ist ein Qt-Widget zur Darstellung einer Tabelle.

Autor

Thomas Dreibholz dreibh@iem.uni-due.de

Version

1.0

4.13.2 Dokumentation der Aufzählungstypen

4.13.2.1 enum `TableView::TableViewMode`

Modus für den Viewer: Editieren der Tupel oder Selektion eines Tupels.

Aufzählungswerte:

TVM_EditMode

TVM_SelectMode

TVM_ReadOnly

4.13.3 Beschreibung der Konstruktoren und Destruktoren

4.13.3.1 `TableView::TableView (SQLConnectionInterface * connection,
const ViewTable * viewTable, const cardinal mode = TVM_EditMode, const
QString & joinAttribute = QString::null, const QString & joinValue =
QString::null, const QString & selectAttribute = QString::null, const
QString & selectValue = QString::null, QWidget * parent = NULL, const char
* name = NULL)`

Constructor.

Parameter

<i>connection</i>	Datenbank-Verbindung.
<i>viewTable</i>	ViewTable-Struktur für den Viewer.
<i>mode</i>	Modus des Viewers: Editier- oder Selektiermodus.
<i>joinAttribute</i>	Attribut für Join.
<i>joinValue</i>	Wert für Join.
<i>select-Attribute</i>	Attribut für Selektion.
<i>selectValue</i>	Start-Wert für Selektion.
<i>parent</i>	Parent QWidget; default: NULL.
<i>name</i>	Widget-Name; default: NULL.

4.13.3.2 TableView::~~TableView ()

Destructor.

4.13.4 Dokumentation der Elementfunktionen

4.13.4.1 void TableView::cancelClicked () [slot]

Qt-Slot: Tupel-Auswahl: Abbruch-Button geklickt.

4.13.4.2 void TableView::closeEditor (TupleEditor * editor, const bool committed) [slot]

Qt-Slot: Einen geöffneten Tupel-Editor wieder schließen.

4.13.4.3 void TableView::closeEvent (QCloseEvent * event) [private]

4.13.4.4 void TableView::doneSelection (TableView * viewer, const bool selected, const QString & selected) [signal]

Qt-Signal: Tupel-Auswahl: Auswahl durchgeführt.

4.13.4.5 void TableView::loadTable (const char * tableName = NULL, const char * orderBy = NULL)

Laden einer Tabelle aus der Datenbank.

Parameter

<i>tableName</i>	Tabellen-Name.
<i>orderBy</i>	Sortierung (z.B. 'Name, Vorname')

4.13.4.6 void **TableView::newClicked** () [slot]

Qt-Slot: Neues Tupel einfügen.

4.13.4.7 void **TableView::nullClicked** () [slot]

Qt-Slot: Tupel-Auswahl: Null-Button geklickt.

4.13.4.8 void **TableView::okayClicked** () [slot]

Qt-Slot: Tupel-Auswahl: Okay-Button geklickt.

4.13.4.9 void **TableView::search** (const QString & *newText*) [slot]

Qt-Slot: Datensuche.

4.13.4.10 void **TableView::selectTuple** (QListViewItem * *item*) [slot]

Qt-Slot: Tupel ausgewählt.

4.13.4.11 void **TableView::selectView** (int *selection*) [slot]

Qt-Slot: View ausgewählt.

4.13.4.12 void **TableView::setSortAttribute** (int *section*) [slot]

Qt-Slot: Header des ListView-Widget angeklickt.

4.13.5 Dokumentation der Datenelemente

4.13.5.1 QList<QString> **TableView::Attributes** [private]

4.13.5.2 SQLConnectionInterface* **TableView::Connection** [private]

4.13.5.3 QLabel* **TableView::FindLabel** [private]

4.13.5.4 QLineEdit* **TableView::FindLineEdit** [private]

4.13.5.5 cardinal **TableView::FindSection** [private]

4.13.5.6 QString **TableView::JoinAttribute** [private]

4.13.5.7 QString **TableView::JoinValue** [private]

- 4.13.5.8 `cardinal TableView::MinNotFoundIndex` [private]
- 4.13.5.9 `cardinal TableView::Mode` [private]
- 4.13.5.10 `QString TableView::SelectAttribute` [private]
- 4.13.5.11 `QString TableView::SelectValue` [private]
- 4.13.5.12 `bool TableView::SkipNextUpdate` [private]
- 4.13.5.13 `QLabel* TableView::StatusBar` [private]
- 4.13.5.14 `QListView* TableView::TableView` [private]
- 4.13.5.15 `QDict<TupleEditor> TableView::TupleEditors` [private]
- 4.13.5.16 `const ViewEntry* TableView::View` [private]
- 4.13.5.17 `const ViewTable* TableView::Views` [private]
- 4.13.5.18 `QWhatsThis* TableView::WhatsThis` [private]

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

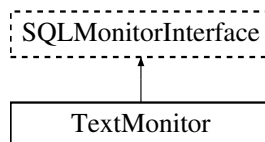
- [tableviewer.h](#)
- [tableviewer.cc](#)

4.14 TextMonitor Klassenreferenz

Text Monitor.

```
#include <textmonitor.h>
```

Klassendiagramm für TextMonitor:



Öffentliche Slots

- void `write` (const char *string="")

Öffentliche Methoden

- [TextMonitor](#) (QWidget *parent=NULL, const char *name=NULL)

Private Attribute

- QMultiLineEdit * [Output](#)

4.14.1 Ausführliche Beschreibung

Text Monitor.

[TextMonitor](#) ist ein Qt-Widget für die Ausgabe von Textzeilen wie z.B. beim SQL Monitor.

Autor

Thomas Dreibholz dreibh@iem.uni-due.de

Version

1.0

4.14.2 Beschreibung der Konstruktoren und Destruktoren

4.14.2.1 `TextMonitor::TextMonitor (QWidget * parent = NULL, const char * name = NULL)`

Constructor.

Parameter

<i>parent</i>	Parent QWidget; default: NULL.
<i>name</i>	Widget-Name; default: NULL.

4.14.3 Dokumentation der Elementfunktionen

4.14.3.1 `void TextMonitor::write (const char * string = " ") [virtual, slot]`

Ausgabe eines SQL-Befehls.

Parameter

<i>string</i>	Befehls-String.
---------------	-----------------

Implementiert [SQLMonitorInterface](#).

4.14.4 Dokumentation der Datenelemente

4.14.4.1 QMultiLineEdit* TextMonitor::Output [private]

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

- [textmonitor.h](#)
- [textmonitor.cc](#)

4.15 Tuple Strukturreferenz

[Tuple](#).

```
#include <tupleeditor.h>
```

Öffentliche Attribute

- QString [ID](#)
- QString [TableName](#)
- const [ViewEntry](#) * [View](#)
- [cardinal](#) [Attributes](#)
- QString [Attribute](#) [[MaxAttributes](#)]
- bool [IsKey](#) [[MaxAttributes](#)]
- QString [Value](#) [[MaxAttributes](#)]

Statische öffentliche Attribute

- static const [cardinal](#) [MaxAttributes](#) = 32

4.15.1 Ausführliche Beschreibung

[Tuple](#).

Die Tuple-Struktur speichert Informationen für den Tupel-Editor.

Autor

Thomas Dreibholz dreibh@iem.uni-due.de

Version

1.0

4.15.2 Dokumentation der Datenelemente

4.15.2.1 QString Tuple::Attribute[MaxAttributes]

4.15.2.2 cardinal Tuple::Attributes

4.15.2.3 QString Tuple::ID

4.15.2.4 bool Tuple::IsKey[MaxAttributes]

4.15.2.5 const cardinal Tuple::MaxAttributes = 32 [static]

4.15.2.6 QString Tuple::TableName

4.15.2.7 QString Tuple::Value[MaxAttributes]

4.15.2.8 const ViewEntry* Tuple::View

Die Dokumentation für diese Struktur wurde erzeugt aufgrund der Datei:

- [tupleeditor.h](#)

4.16 TupleEditor Klassenreferenz

[Tuple](#) Editor.

```
#include <tupleeditor.h>
```

Öffentliche Slots

- void [update](#) ()
- void [remove](#) ()
- void [abort](#) ()

Signale

- void [done](#) ([TupleEditor](#) *editor, const bool committed)

Öffentliche Methoden

- [TupleEditor](#) ([SQLConnectionInterface](#) *connection, [Tuple](#) *tuple, const bool new-Tuple, const QString &joinAttribute, QWidget *parent=NULL, const char *name=NULL)
- [~TupleEditor](#) ()
- const [Tuple](#) * [getTuple](#) () const

Private Methoden

- void `closeEvent` (QCloseEvent *event)

Private Attribute

- `SQLConnectionInterface` * `Connection`
- `QWhatsThis` * `WhatsThis`
- `QLabel` * `StatusBar`
- `AttributeEditor` ** `AttrEdit`
- `Tuple` * `EditorTuple`
- `TableView` * `Selector`
- int `SelectorID`
- bool `NewTuple`

4.16.1 Ausführliche Beschreibung

`Tuple` Editor.

`TupleEditor` ist ein Qt-Widget für den Tupel-Editor.

Autor

Thomas Dreibholz dreibh@iem.uni-due.de

Version

1.0

4.16.2 Beschreibung der Konstruktoren und Destruktoren

4.16.2.1 `TupleEditor::TupleEditor` (`SQLConnectionInterface` * `connection`, `Tuple` * `tuple`, const bool `newTuple`, const `QString` & `joinAttribute`, `QWidget` * `parent` = NULL, const char * `name` = NULL)

Constructor.

Parameter

<i>connection</i>	Datenbank-Verbindung.
<i>tuple</i>	Tuple-Struktur, die zu editierendes Tupel beschreibt.
<i>newTuple</i>	true, um neues Tupel zu erzeugen; false sonst.
<i>joinAttribute</i>	Join-Attribut.
<i>parent</i>	Parent QWidget; default: NULL.
<i>name</i>	Widget-Name; default: NULL.

4.16.2.2 TupleEditor::~~TupleEditor ()

Destructor.

4.16.3 Dokumentation der Elementfunktionen

4.16.3.1 void TupleEditor::abort () [slot]

Qt-Slot: Abbrechen.

4.16.3.2 void TupleEditor::closeEvent (QCloseEvent * event) [private]

4.16.3.3 void TupleEditor::done (TupleEditor * editor, const bool committed) [signal]

Qt-Signal: Editieren beendet.

4.16.3.4 const Tuple * TupleEditor::getTuple () const

Rückgabe der Tuple-Struktur des Editors.

Rückgabe

[Tuple](#)

4.16.3.5 void TupleEditor::remove () [slot]

Qt-Slot: Delete durchführen.

4.16.3.6 void TupleEditor::update () [slot]

Qt-Slot: Update bzw. Insert durchführen.

4.16.4 Dokumentation der Datenelemente

4.16.4.1 AttributeEditor** TupleEditor::AttrEdit [private]

4.16.4.2 SQLConnectionInterface* TupleEditor::Connection [private]

4.16.4.3 Tuple* TupleEditor::EditorTuple [private]

4.16.4.4 bool TupleEditor::NewTuple [private]

4.16.4.5 `TableViewer* TupleEditor::Selector` [private]

4.16.4.6 `int TupleEditor::SelectorID` [private]

4.16.4.7 `QLabel* TupleEditor::StatusBar` [private]

4.16.4.8 `QWhatsThis* TupleEditor::WhatsThis` [private]

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

- [tupleeditor.h](#)
- [tupleeditor.cc](#)

4.17 AttributeEditor::ValueListEntry Strukturreferenz

Öffentliche Attribute

- `QString` [Key](#)
- `QString` [Value](#)

4.17.1 Dokumentation der Datenelemente

4.17.1.1 `QString AttributeEditor::ValueListEntry::Key`

4.17.1.2 `QString AttributeEditor::ValueListEntry::Value`

Die Dokumentation für diese Struktur wurde erzeugt aufgrund der Datei:

- [attributeeditor.h](#)

4.18 ViewEntry Strukturreferenz

View Entry.

```
#include <paviews.h>
```

Öffentliche Methoden

- `const QString` [toTitle](#) (const char *name) const
- `const QString` [toHelp](#) (const char *name) const

Öffentliche Attribute

- const char * [Title](#)
- const char * [ViewName](#)
- const char * [OrderBy](#)
- const [cardinal Joins](#)
- const [JoinEntry](#) * [Join](#)
- const [cardinal Selects](#)
- const [SelectEntry](#) * [Select](#)
- const [cardinal Descriptions](#)
- const [DescriptionEntry](#) * [Description](#)

4.18.1 Ausführliche Beschreibung

View Entry.

Diese Struktur speichert Informationen eine Sicht einer Tabelle.

Autor

Thomas Dreibholz dreibh@iem.uni-due.de

Version

1.0

4.18.2 Dokumentation der Elementfunktionen

4.18.2.1 const QString ViewEntry::toHelp (const char * *name*) const

4.18.2.2 const QString ViewEntry::toTitle (const char * *name*) const

4.18.3 Dokumentation der Datenelemente

4.18.3.1 const [DescriptionEntry](#)* ViewEntry::Description

4.18.3.2 const [cardinal](#) ViewEntry::Descriptions

4.18.3.3 const [JoinEntry](#)* ViewEntry::Join

4.18.3.4 const [cardinal](#) ViewEntry::Joins

4.18.3.5 const char* ViewEntry::OrderBy

4.18.3.6 const [SelectEntry](#)* ViewEntry::Select

4.18.3.7 const [cardinal](#) ViewEntry::Selects

4.18.3.8 `const char* ViewEntry::Title`

4.18.3.9 `const char* ViewEntry::ViewName`

Die Dokumentation für diese Struktur wurde erzeugt aufgrund der Dateien:

- [pviews.h](#)
- [pviews.cc](#)

4.19 ViewTable Strukturreferenz

View Table.

```
#include <pviews.h>
```

Öffentliche Attribute

- `const char * PrimaryKey [MaxPrimaryKeyEntries]`
- `const char * BaseTableName`
- `const char * PixmapName`
- `const cardinal Views`
- `const ViewEntry * View`

Statische öffentliche Attribute

- `static const cardinal MaxPrimaryKeyEntries = 5`

4.19.1 Ausführliche Beschreibung

View Table.

Diese Struktur enthält alle Sichten einer Tabelle.

Autor

Thomas Dreibholz dreibh@iem.uni-due.de

Version

1.0

4.19.2 Dokumentation der Datenelemente

4.19.2.1 `const char* ViewTable::BaseTableName`

4.19.2.2 `const cardinal ViewTable::MaxPrimaryKeyEntries = 5` [static]

4.19.2.3 `const char* ViewTable::PixmapName`

4.19.2.4 `const char* ViewTable::PrimaryKey[MaxPrimaryKeyEntries]`

4.19.2.5 `const ViewEntry* ViewTable::View`

4.19.2.6 `const cardinal ViewTable::Views`

Die Dokumentation für diese Struktur wurde erzeugt aufgrund der Datei:

- [paviews.h](#)

Kapitel 5

Datei-Dokumentation

5.1 attributeeditor.cc-Dateireferenz

```
#include "system.h" #include "attributeeditor.h" #include
"sqlmessages.h" #include <qlayout.h> #include <qpushbutton.-
h> #include <qcombobox.h> #include "attributeeditor.moc"
```

5.2 attributeeditor.h-Dateireferenz

```
#include "system.h" #include "paclient.h" #include "paviews.-
h" #include "tableviewer.h" #include <qapp.h> #include
<qstring.h> #include <qlabel.h> #include <qlineedit.h>
#include <qlist.h>
```

Klassen

- class [AttributeEditor](#)
- struct [AttributeEditor::ValueListEntry](#)

5.3 generator.cc-Dateireferenz

```
#include <stdio.h> #include <stdlib.h> #include <string.-
h> #include <fstream.h> #include <iostream.h> #include
<time.h>
```

Klassen

- struct [list_t](#)

Funktionen

- int `odd` (int num)
- int `even` (int num)
- void `Randomize` ()
- `list_t` * `load_list` (const char *name, const int number)
- char * `rand_element` (const `list_t` *list)
- int `rand_int` (int number)
- void `load_all_lists` (void)
- void `begin_transaction` ()
- void `commit_work` ()
- void `write_qualies_intro` ()
- void `w_q_s` ()
- void `write_qualies_stud_id` (int stud_id)
- void `write_qualies_typ` (int q_typ)
- void `write_qualies_title` (int q_typ)
- void `write_qualies_pers_id` (int faculty)
- void `write_qualies_date` (int sem_year, int sem_typ)
- void `write_qualies_date_late` (int sem_year, int sem_typ)
- void `write_qualies_comment` ()
- void `write_qualies_closing` ()
- int `qualies_test` (int q_typ, int stud_level, int stud_id, int sem_year, int sem_typ)
- void `w_e_s` ()
- void `write_exams_date` (int sem_year, int sem_typ)
- int `generate_exams_result` (int stud_level, int sem_year, int sem_typ)
- void `write_exams_intro` ()
- void `write_exams_stud_id` (int stud_id)
- void `write_exams_typ` (int exam_typ)
- void `write_exams_counter` (int counter, int subcounter)
- void `write_exams_entry_date` (int sem_year, int sem_typ)
- void `write_exams_pers_id` (int stud_id)
- void `write_exams_comment` (int stud_level)
- void `write_exams_closing` ()
- int `dp_exams_test` (int exam_typ, int stud_level, int stud_id, int sem_year, int sem_typ, int counter)
- void `write_exams_thesis_intro` ()
- void `write_exams_thesis_entry_date` (int sem_year, int sem_typ)
- void `write_exams_thesis_dates` (int sem_year, int sem_typ)
- void `write_exams_thesis_theme` ()
- int `generate_exams_thesis_result` (int stud_level)
- int `dp_exams_thesis` (int exam_typ, int stud_level, int stud_id, int sem_year, int sem_typ, int counter)
- void `delete_student_data` ()
- void `delete_student_data` (int min_id, int max_id)
- int `generate_personal_data` (int stud_id)
- void `generate_student_semester` (int stud_id, int immat_year)
- void `generate_student` (int stud_id)
- void `vacuum_database` ()
- int `main` (int argc, char *argv[])

Variablen

- `list_t * list_names_male`
- `list_t * list_names_female`
- `list_t * list_names_family`
- `list_t * list_names_towns`
- `list_t * list_names_street`
- `list_t * list_names_str_typ`
- `const int MAX_ELEMENTS = 500`
- `const int MIN_STUD_ID = 20000000`
- `const int MAX_STUD_ID = 50000000`
- `const int MIN_PERS_ID = 98765431`
- `const int MAX_PERS_ID = 98765460`
- `const int SEX_QUOTA = 20`
- `const int BIRTH_START = 1970`
- `const int BIRTH_VAR = 11`
- `const int ABI_AGE = 18`
- `const int IMMAT_VAR = 6`
- `const int YEAR_NOW = 2000`
- `const int SEM_LIMIT = 6`
- `const int MAX_QUALIES = 17`
- `const int Q_HZB = 0`
- `const int Q_INF_1 = 1`
- `const int Q_INF_2 = 2`
- `const int Q_INF_3 = 3`
- `const int Q_INF_4 = 4`
- `const int Q_INF_PS = 5`
- `const int Q_INF_PP = 6`
- `const int Q_M_IR_1 = 7`
- `const int Q_M_IR_2 = 8`
- `const int Q_M_LA_1 = 9`
- `const int Q_M_LA_2 = 10`
- `const int Q_V_DIPL = 11`
- `const int Q_INF_SA = 12`
- `const int Q_INF_SB = 13`
- `const int Q_INF_PC = 14`
- `const int Q_M_PMWR = 15`
- `const int Q_DIPLOM = 16`
- `const int MAX_DP_EXAMS = 11`
- `const int DP_VD_DIPLOM = 0`
- `const int DP_VD_INF_A = 1`
- `const int DP_VD_INF_B = 2`
- `const int DP_VD_MATHE = 3`
- `const int DP_VD_NF = 4`
- `const int DP_HD_INF_A = 5`
- `const int DP_HD_INF_B = 6`
- `const int DP_HD_INF_C = 7`

- const int DP_HD_NF = 8
- const int DP_HD_ARBEIT = 9
- const int DP_HD_DIPLOM = 10
- const int NO = 1
- const int YES = 0
- const int NOT_PASSED = 1
- const int YES_PASSED = 0
- const int MAX_VD_TRY = 3
- const int MAX_HD_TRY = 3
- const int MAX_DIPL_TRY = 2
- const int WIN_SEM = 0
- const int SOM_SEM = 1

5.3.1 Dokumentation der Funktionen

5.3.1.1 void begin_transaction ()

5.3.1.2 void commit_work ()

5.3.1.3 void delete_student_data ()

5.3.1.4 void delete_student_data (int min_id, int max_id)

5.3.1.5 int dp_exams_test (int exam_typ, int stud_level, int stud_id, int sem_year, int sem_typ, int counter)

5.3.1.6 int dp_exams_thesis (int exam_typ, int stud_level, int stud_id, int sem_year, int sem_typ, int counter)

5.3.1.7 int even (int num)

5.3.1.8 int generate_exams_result (int stud_level, int sem_year, int sem_typ)

5.3.1.9 int generate_exams_thesis_result (int stud_level)

5.3.1.10 int generate_personal_data (int stud_id)

5.3.1.11 void generate_student (int stud_id)

5.3.1.12 void generate_student_semester (int stud_id, int immat_year)

5.3.1.13 void load_all_lists (void)

5.3.1.14 list_t* load_list (const char * name, const int number)

5.3.1.15 int main (int argc, char * argv[])

- 5.3.1.16 `int odd (int num)`
- 5.3.1.17 `int qualies_test (int q_typ, int stud_level, int stud_id, int sem_year, int sem_typ)`
- 5.3.1.18 `char* rand_element (const list_t* list)`
- 5.3.1.19 `int rand_int (int number)`
- 5.3.1.20 `void Randomize ()`
- 5.3.1.21 `void vacuum_database ()`
- 5.3.1.22 `void w_e_s ()`
- 5.3.1.23 `void w_q_s ()`
- 5.3.1.24 `void write_exams_closing ()`
- 5.3.1.25 `void write_exams_comment (int stud_level)`
- 5.3.1.26 `void write_exams_counter (int counter, int subcounter)`
- 5.3.1.27 `void write_exams_date (int sem_year, int sem_typ)`
- 5.3.1.28 `void write_exams_entry_date (int sem_year, int sem_typ)`
- 5.3.1.29 `void write_exams_intro ()`
- 5.3.1.30 `void write_exams_pers_id (int stud_id)`
- 5.3.1.31 `void write_exams_stud_id (int stud_id)`
- 5.3.1.32 `void write_exams_thesis_dates (int sem_year, int sem_typ)`
- 5.3.1.33 `void write_exams_thesis_entry_date (int sem_year, int sem_typ)`
- 5.3.1.34 `void write_exams_thesis_intro ()`
- 5.3.1.35 `void write_exams_thesis_theme ()`
- 5.3.1.36 `void write_exams_typ (int exam_typ)`
- 5.3.1.37 `void write_qualies_closing ()`
- 5.3.1.38 `void write_qualies_comment ()`
- 5.3.1.39 `void write_qualies_date (int sem_year, int sem_typ)`

5.3.1.40 void write_qualies_date_late (int sem_year, int sem_typ)

5.3.1.41 void write_qualies_intro ()

5.3.1.42 void write_qualies_pers_id (int faculty)

5.3.1.43 void write_qualies_stud_id (int stud.id)

5.3.1.44 void write_qualies_title (int q_typ)

5.3.1.45 void write_qualies_typ (int q_typ)

5.3.2 Variablen-Dokumentation

5.3.2.1 const int ABI_AGE = 18

5.3.2.2 const int BIRTH_START = 1970

5.3.2.3 const int BIRTH_VAR = 11

5.3.2.4 const int DP_HD_ARBEIT = 9

5.3.2.5 const int DP_HD_DIPLOM = 10

5.3.2.6 const int DP_HD_INF_A = 5

5.3.2.7 const int DP_HD_INF_B = 6

5.3.2.8 const int DP_HD_INF_C = 7

5.3.2.9 const int DP_HD_NF = 8

5.3.2.10 const int DP_VD_DIPLOM = 0

5.3.2.11 const int DP_VD_INF_A = 1

5.3.2.12 const int DP_VD_INF_B = 2

5.3.2.13 const int DP_VD_MATHE = 3

5.3.2.14 const int DP_VD_NF = 4

5.3.2.15 const int IMMAT_VAR = 6

5.3.2.16 list_t* list_names_family

5.3.2.17 list_t* list_names_female

- 5.3.2.18 list_t* list_names_male
- 5.3.2.19 list_t* list_names_str_typ
- 5.3.2.20 list_t* list_names_street
- 5.3.2.21 list_t* list_names_towns
- 5.3.2.22 const int MAX_DIPL_TRY = 2
- 5.3.2.23 const int MAX_DP_EXAMS = 11
- 5.3.2.24 const int MAX_ELEMENTS = 500
- 5.3.2.25 const int MAX_HD_TRY = 3
- 5.3.2.26 const int MAX_PERS_ID = 98765460
- 5.3.2.27 const int MAX_QUALIES = 17
- 5.3.2.28 const int MAX_STUD_ID = 50000000
- 5.3.2.29 const int MAX_VD_TRY = 3
- 5.3.2.30 const int MIN_PERS_ID = 98765431
- 5.3.2.31 const int MIN_STUD_ID = 20000000
- 5.3.2.32 const int NO = 1
- 5.3.2.33 const int NOT_PASSED = 1
- 5.3.2.34 const int Q_DIPLOM = 16
- 5.3.2.35 const int Q_HZB = 0
- 5.3.2.36 const int Q_INF_1 = 1
- 5.3.2.37 const int Q_INF_2 = 2
- 5.3.2.38 const int Q_INF_3 = 3
- 5.3.2.39 const int Q_INF_4 = 4
- 5.3.2.40 const int Q_INF_PC = 14
- 5.3.2.41 const int Q_INF_PP = 6

5.3.2.42 `const int Q_INF_PS = 5`

5.3.2.43 `const int Q_INF_SA = 12`

5.3.2.44 `const int Q_INF_SB = 13`

5.3.2.45 `const int Q_M_IR_1 = 7`

5.3.2.46 `const int Q_M_IR_2 = 8`

5.3.2.47 `const int Q_M_LA_1 = 9`

5.3.2.48 `const int Q_M_LA_2 = 10`

5.3.2.49 `const int Q_M_PMWR = 15`

5.3.2.50 `const int Q_V_DIPL = 11`

5.3.2.51 `const int SEM_LIMIT = 6`

5.3.2.52 `const int SEX_QUOTA = 20`

5.3.2.53 `const int SOM_SEM = 1`

5.3.2.54 `const int WIN_SEM = 0`

5.3.2.55 `const int YEAR_NOW = 2000`

5.3.2.56 `const int YES = 0`

5.3.2.57 `const int YES_PASSED = 0`

5.4 logindialog.cc-Dateireferenz

```
#include "system.h" #include "textmonitor.h" #include
"sqlmessages.h" #include <qapp.h> #include <qlayout.h> ×
#include <qlabel.h> #include <qpushbutton.h> #include
<qlineedit.h> #include "logindialog.moc"
```

5.5 logindialog.h-Dateireferenz

```
#include "system.h" #include "sqlconnection.h" #include
<qapp.h> #include <qdialog.h> #include <qstring.h> ×
#include <qlineedit.h>
```

Klassen

- class [LoginDialog](#)
Login Dialog.

5.6 paclient.cc-Dateireferenz

```
#include "system.h" #include "paclient.h" #include "paviews.-  
h" #include "tableviewer.h" #include "textmonitor.h" ×  
#include "sqlconnectioninterface.h" #include "sqlmessages.-  
h" #include <qapp.h> #include <qplatinumstyle.h> #include  
<qlayout.h> #include <qmenubar.h> #include <qpopupmenu.-  
h> #include <qtabwidget.h> #include <qprogressdialog.h>  
#include <qmessagebox.h> #include "paclient.moc"
```

5.7 paclient.h-Dateireferenz

```
#include "system.h" #include "sqlconnectioninterface.-  
h" #include "textmonitor.h" #include <qapp.h> #include  
<qmainwindow.h>
```

Klassen

- class [PAClient](#)
PAClient.

5.8 padb.cc-Dateireferenz

```
#include "system.h" #include "paclient.h" #include "logindialog.-  
h" #include "sqlconnection.h" #include <qapp.h> #include  
<qplatinumstyle.h>
```

Funktionen

- int [main](#) (int argc, char **argv)

5.8.1 Dokumentation der Funktionen

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

5.9 paviews.cc-Dateireferenz

```
#include "system.h" #include "paviews.h"
```

Variablen

- const [ViewTable Pruefer_ViewTable](#)
- const [ViewTable Studenten_ViewTable](#)
- const [DescriptionEntry Adresstypen_DescriptionEntries \[\]](#)
- const [ViewEntry Adresstypen_ViewEntries \[\]](#)
- const [ViewTable Adresstypen_ViewTable](#)
- const [DescriptionEntry Adressen_DescriptionEntries \[\]](#)
- const [SelectEntry Adressen_SelectEntries \[\]](#)
- const [ViewEntry Adressen_ViewEntries \[\]](#)
- const [ViewTable Adressen_ViewTable](#)
- const [DescriptionEntry Studientypen_DescriptionEntries \[\]](#)
- const [ViewEntry Studientypen_ViewEntries \[\]](#)
- const [ViewTable Studientypen_ViewTable](#)
- const [DescriptionEntry Semestertext_DescriptionEntries \[\]](#)
- const [ViewEntry Semestertext_ViewEntries \[\]](#)
- const [ViewTable Semestertext_ViewTable](#)
- const [DescriptionEntry Studium_DescriptionEntries \[\]](#)
- const [SelectEntry Studium_SelectEntries \[\]](#)
- const [ViewEntry Studium_ViewEntries \[\]](#)
- const [ViewTable Studium_ViewTable](#)
- const [DescriptionEntry Qualifikattypen_DescriptionEntries \[\]](#)
- const [ViewEntry Qualifikattypen_ViewEntries \[\]](#)
- const [ViewTable Qualifikattypen_ViewTable](#)
- const [DescriptionEntry Leistungsnachweise_DescriptionEntries \[\]](#)
- const [SelectEntry Leistungsnachweise_SelectEntries \[\]](#)
- const [ViewEntry Leistungsnachweise_ViewEntries \[\]](#)
- const [ViewTable Leistungsnachweise_ViewTable](#)
- const [DescriptionEntry Pruefungstypen_DescriptionEntries \[\]](#)
- const [ViewEntry Pruefungstypen_ViewEntries \[\]](#)
- const [ViewTable Pruefungstypen_ViewTable](#)
- const [DescriptionEntry Pruefungsstatus_DescriptionEntries \[\]](#)
- const [ViewEntry Pruefungsstatus_ViewEntries \[\]](#)
- const [ViewTable Pruefungsstatus_ViewTable](#)
- const [DescriptionEntry Pruefungen_DescriptionEntries \[\]](#)
- const [SelectEntry Pruefungen_SelectEntries \[\]](#)
- const [ViewEntry Pruefungen_ViewEntries \[\]](#)
- const [ViewTable Pruefungen_ViewTable](#)
- const [DescriptionEntry Diplomarbeiten_DescriptionEntries \[\]](#)
- const [SelectEntry Diplomarbeiten_SelectEntries \[\]](#)
- const [ViewEntry Diplomarbeiten_ViewEntries \[\]](#)
- const [ViewTable Diplomarbeiten_ViewTable](#)

- const `DescriptionEntry` `Pruefer_DescriptionEntries` []
- const `JoinEntry` `Pruefer_JoinEntries` []
- const `ViewEntry` `Pruefer_ViewEntries` []
- const `DescriptionEntry` `Studenten_DescriptionEntries` []
- const `JoinEntry` `Studenten_JoinEntries` []
- const `ViewEntry` `Studenten_ViewEntries` []
- const `DescriptionEntry` `Statistiken_DescriptionEntries` []
- const `ViewEntry` `Statistiken_ViewEntries` []
- const `ViewTable` `Statistiken_ViewTable`
- const char * `PADB_OptimizeTableEntries` []
- struct `OptimizeTable` `PADB_OptimizeTable`

5.9.1 Variablen-Dokumentation

5.9.1.1 const `DescriptionEntry` `Adressen_DescriptionEntries` []

Initialisierung:

```
{
  {"id",           "Nummer",      "Kennung"},
  {"zaehler",     "Zähler",      "Zähler"},
  {"strasse",     "Straße",      "Straße und Hausnummer"},
  {"ort",         "Ort",         "Ort"},
  {"plz",         "PLZ",        "Postleitzahl"},
  {"land",        "Land",       "Land"},
  {"telefon1",    "Telefon 1",    "1. Telefon-Nummer"},
  {"telefon2",    "Telefon 2",    "2. Telefon-Nummer"},
  {"fax",         "Telefax",     "Telefax-Nummer"},
  {"email",       "EMail",       "EMail-Adresse"},
  {"adr_typ",     "Typ",        "Adreß-Typ"},
  {"sonstiges",   "Sonstiges",    "Sonstiges"},
  {"raum",        "Raum",        "Raum"},
  {"adr_typ",     "Typ",        "Adreß-Typ"},
  {"pers_adressen", "Adresse",     "Tupel der Adressen-Tabelle"}
}
```

5.9.1.2 const `SelectEntry` `Adressen_SelectEntries` []

Initialisierung:

```
{
  {"adr_typ", "adr_typ", "adr_txt", true, &Adresstypen_ViewTable},
}
```

5.9.1.3 const `ViewEntry` `Adressen_ViewEntries` []

Initialisierung:

```

{
    {
        "Alle Adressen", "pers_adressen", "ort",
        0, NULL,
        sizeof(Adressen_SelectEntries) / sizeof(SelectEntry),
        (const SelectEntry*)&Adressen_SelectEntries,
        sizeof(Adressen_DescriptionEntries) / sizeof(DescriptionEntry),
        (const DescriptionEntry*)&Adressen_DescriptionEntries
    }
}

```

5.9.1.4 const ViewTable Adressen_ViewTable

Initialisierung:

```

{
    {"id", "zaehler", NULL, NULL, NULL},
    "pers_adressen",
    "Pixmaps/Adressen.xpm",
    sizeof(Adressen_ViewEntries) / sizeof(ViewEntry),
    (const ViewEntry*)&Adressen_ViewEntries
}

```

5.9.1.5 const DescriptionEntry Adresstypen_DescriptionEntries[]

Initialisierung:

```

{
    {"adr_typ", "Typ", "Adreß-Typ"},
    {"adr_txt", "Beschreibung", "Beschreibung"},
    {"typ_adressen", "Adreßtyp", "Tupel der Adreßtypen-Tabelle"}
}

```

5.9.1.6 const ViewEntry Adresstypen_ViewEntries[]

Initialisierung:

```

{
    {
        "Alle Adreßtypen", "typ_adressen", "adr_typ",
        0, NULL,
        0, NULL,
        sizeof(Adresstypen_DescriptionEntries) / sizeof(DescriptionEntry),
        (const DescriptionEntry*)&Adresstypen_DescriptionEntries
    }
}

```

5.9.1.7 const ViewTable Adresstypen_ViewTable

Initialisierung:

```

{
    {"adr_typ", NULL, NULL, NULL, NULL},
    "typ_adressen",
    "Pixmaps/Adreßtypen.xpm",
    sizeof(Adresstypen_ViewEntries) / sizeof(ViewEntry),
    (const ViewEntry*)&Adresstypen_ViewEntries
}

```

5.9.1.8 const DescriptionEntry Diplomarbeiten_DescriptionEntries[]

Initialisierung:

```

{
    {"stud_id",          "MatrNr",          "Matrikelnummer"},
    {"typ_id",           "Typ",           "Typ"},
    {"gew_pruefer_id",  "Gew.Pr.",      "Gewünschter Prüfer"},
    {"datum",           "Datum",        "Datum"},
    {"versuch",         "Versuch",      "Versuch"},
    {"zaehler",         "Zähler",       "Zähler"},
    {"thema",           "Thema",        "Thema"},
    {"anmelde_datum",  "Anmeldung",    "Datum der Anmeldung"},
    {"abgabe_datum",   "Abgabe",       "Datum der Abgabe"},
    {"abgabe_limit",   "Abgabe-Limit", "Abgabe-Limit"},
    {"ausgabe_datum",  "Ausgabe",      "Datum der Ausgabe"},
    {"pruefer1_id",    "1.Prüfer",     "Erster Prüfer"},
    {"pruefer2_id",    "2.Prüfer",     "Zweiter Prüfer"},
    {"status_id",       "Status",        "Status"},
    {"note",            "Note",          "Note"},
    {"bemerkung",       "Bemerkung",    "Bemerkung"},
    {"uni_diplarbeit",  "Diplomarbeit", "Tupel der Diplomarbeiten-Tabelle"}
}

```

5.9.1.9 const SelectEntry Diplomarbeiten_SelectEntries[]

Initialisierung:

```

{
    {"gew_pruefer_id", "pers_id",      "vorname, name, pers_id", false, &
     Pruefer_ViewTable},
    {"pruefer1_id",   "pers_id",      "vorname, name, pers_id", false, &
     Pruefer_ViewTable},
    {"pruefer2_id",   "pers_id",      "vorname, name, pers_id", false, &
     Pruefer_ViewTable},
    {"status_id",     "status_typ", "status_txt", true, &
     Pruefungsstatus_ViewTable},
    {"stud_id", "stud_id", "vorname, name, stud_id", false, &Studenten_ViewTable
    }
}

```

5.9.1.10 const ViewEntry Diplomarbeiten_ViewEntries[]

5.9.1.11 const ViewTable Diplomarbeiten_ViewTable

Initialisierung:

```
{
    {"stud_id", "versuch", "zaehler", NULL, NULL},
    "uni_diplarbeit",
    "Pixmaps/Diplomarbeiten.xpm",
    sizeof(Diplomarbeiten_ViewEntries) / sizeof(ViewEntry),
    (const ViewEntry*)&Diplomarbeiten_ViewEntries
}
```

5.9.1.12 const DescriptionEntry Leistungsnachweise_DescriptionEntries[]

Initialisierung:

```
{
    {"stud_id", "MatrNr", "Matrikelnummer"},
    {"pers_id", "PersNr", "Personalnummer"},
    {"datum", "Datum", "Datum"},
    {"titel", "Titel", "Titel"},
    {"bemerkung", "Bemerkung", "Bemerkung"},
    {"gal_typ", "Qualifikation", "Qualifikations-Art"},
    {"uni_qualifikate", "Leistungsnachweis", "Tupel der
    Leistungsnachweise-Tabelle"}
}
```

5.9.1.13 const SelectEntry Leistungsnachweise_SelectEntries[]

Initialisierung:

```
{
    {"gal_typ", "gal_typ", "gal_txt", false, &
    Qualifikattypen_ViewTable},
    {"pers_id", "pers_id", "vorname, name, pers_id", false, &Pruefer_ViewTable},
    {"stud_id", "stud_id", "vorname, name, stud_id", false, &Studenten_ViewTable
    }
}
```

5.9.1.14 const ViewEntry Leistungsnachweise_ViewEntries[]

Initialisierung:

```
{
    {
        "Alle Leistungsnachweise", "uni_qualifikate", "datum",
        0,
        NULL,
        sizeof(Leistungsnachweise_SelectEntries) / sizeof(SelectEntry),
        (const SelectEntry*)&Leistungsnachweise_SelectEntries,
        sizeof(Leistungsnachweise_DescriptionEntries) / sizeof(DescriptionEntry),
        (const DescriptionEntry*)&Leistungsnachweise_DescriptionEntries
    }
}
```

5.9.1.15 const ViewTable Leistungsnachweise_ViewTable

Initialisierung:

```
{
    {"stud_id", "qal_typ", "datum", "titel", NULL},
    "uni_qualifikate",
    "Pixmaps/Leistungsnachweise.xpm",
    sizeof(Leistungsnachweise_ViewEntries) / sizeof(ViewEntry),
    (const ViewEntry*)&Leistungsnachweise_ViewEntries
}
```

5.9.1.16 struct OptimizeTable PADB_OptimizeTable

Initialisierung:

```
{
    sizeof(PADB_OptimizeTableEntries) / sizeof(const char*),
    (const char*)&PADB_OptimizeTableEntries
}
```

Optimierungstabelle.

5.9.1.17 const char* PADB_OptimizeTableEntries[]

Initialisierung:

```
{
    "uni_pruefungen",
    "uni_diplarbeit",
    "uni_studium",
    "uni_qualifikate",
    "pers_studenten",
    "pers_pruefer",
    "pers_adressen",
    "typ_qualifikate",
    "typ_prfstatus",
    "typ_pruefung",
    "typ_adressen",
    "typ_studium"
}
```

5.9.1.18 const DescriptionEntry Pruefer_DescriptionEntries[]

Initialisierung:

```
{
    {"pers_id", "PersNr", "Personalnummer"},
    {"name", "Name", "Nachname"},
    {"vorname", "Vorname", "Vorname"},
    {"m_w", "M/W", "Männlich/Weiblich"},
    {"titel", "Titel", "Titel"},
    {"rang", "Rang", "Rang"},
}
```

```

{"einstellung_datum",      "Einstellung",  "Datum der Einstellung"},
{"pensionierung_datum",   "Pensionierung", "Datum der Pensionierung"},
{"pers_pruefer", "Prüfer", "Tupel der Prüfer-Tabelle"}
}

```

5.9.1.19 const JoinEntry Pruefer_JoinEntries[]

Initialisierung:

```

{
  {"pers_id", "pers_id",      "Leistungsnachweise", &
   Leistungsnachweise_ViewTable},
  {"pers_id", "pruefer_id",   "Prüfer",               &Pruefungen_ViewTable},
  {"pers_id", "beisitz_id",   "Beisitzer",            &Pruefungen_ViewTable},
  {"pers_id", "pruefer1_id",  "1. Prüfer DA",         &Diplomarbeiten_ViewTable},
  {"pers_id", "pruefer2_id",  "2. Prüfer DA",         &Diplomarbeiten_ViewTable},
  {"pers_id", "id",          "Adressen",             &Adressen_ViewTable}
}

```

5.9.1.20 const ViewEntry Pruefer_ViewEntries[]

Initialisierung:

```

{
  {
    "Alle Prüfer", "pers_pruefer", "name, vorname, titel",
    sizeof(Pruefer_JoinEntries) / sizeof(JoinEntry),
    (const JoinEntry*)&Pruefer_JoinEntries,
    0, NULL,
    sizeof(Pruefer_DescriptionEntries) / sizeof(DescriptionEntry),
    (const DescriptionEntry*)&Pruefer_DescriptionEntries
  },
  {
    "Vorhandene Prüfer", "pers_pruefer_np", "name, vorname, titel",
    sizeof(Pruefer_JoinEntries) / sizeof(JoinEntry),
    (const JoinEntry*)&Pruefer_JoinEntries,
    0, NULL,
    sizeof(Pruefer_DescriptionEntries) / sizeof(DescriptionEntry),
    (const DescriptionEntry*)&Pruefer_DescriptionEntries
  },
  {
    "Pensionierte Prüfer", "pers_pruefer_pe", "name, vorname, titel",
    sizeof(Pruefer_JoinEntries) / sizeof(JoinEntry),
    (const JoinEntry*)&Pruefer_JoinEntries,
    0, NULL,
    sizeof(Pruefer_DescriptionEntries) / sizeof(DescriptionEntry),
    (const DescriptionEntry*)&Pruefer_DescriptionEntries
  }
}
}

```

5.9.1.21 const ViewTable Pruefer_ViewTable

Initialisierung:

```

{
    {"pers_id", NULL, NULL, NULL, NULL},
    "pers_pruefer",
    "Pixmaps/Prüfer.xpm",
    sizeof(Pruefer_ViewEntries) / sizeof(ViewEntry),
    (const ViewEntry*)&Pruefer_ViewEntries
}

```

5.9.1.22 const DescriptionEntry Pruefungen_DescriptionEntries[]

Initialisierung:

```

{
    {"stud_id",      "MatrNr",      "Matrikelnummer"},
    {"typ_id",       "Typ",         "Typ"},
    {"gew_pruefer_id", "Gew.Pr.",    "Gewünschter Prüfer"},
    {"pruefer_id",   "Prüfer",     "Prüfer"},
    {"beisitz_id",   "Beisitzer",  "Beisitzer"},
    {"anmelde_datum", "Anmeldung",  "Datum der Anmeldung"},
    {"thema",        "Thema",       "Thema"},
    {"datum",        "Datum",      "Datum"},
    {"versuch",      "Versuch",    "Versuch"},
    {"zaehler",      "Zähler",     "Zähler"},
    {"status_id",    "Status",     "Status"},
    {"note",         "Note",       "Note"},
    {"bemerkung",    "Bemerkung",  "Bemerkung"},
    {"uni_pruefungen", "Prüfung",   "Tupel der Prüfungen-Tabelle"}
}

```

5.9.1.23 const SelectEntry Pruefungen_SelectEntries[]

Initialisierung:

```

{
    {"typ_id",      "prf_typ",    "prf_txt",    true, &
     Pruefungstypen_ViewTable},
    {"status_id",   "status_typ", "status_txt", true, &
     Pruefungsstatus_ViewTable},
    {"pruefer_id",  "pers_id",    "vorname, name, pers_id", false, &
     Pruefer_ViewTable},
    {"beisitz_id",  "pers_id",    "vorname, name, pers_id", false, &
     Pruefer_ViewTable},
    {"gew_pruefer_id", "pers_id",    "vorname, name, pers_id", false, &
     Pruefer_ViewTable},
    {"stud_id",     "stud_id",    "vorname, name, stud_id", false, &Studenten_ViewTable
    }
}

```

5.9.1.24 const ViewEntry Pruefungen_ViewEntries[]

5.9.1.25 const ViewTable Pruefungen_ViewTable

Initialisierung:

```
{
    {"stud_id", "typ_id", "versuch", "zaehler", NULL},
    "uni_pruefungen",
    "Pixmaps/Prüfungen.xpm",
    sizeof(Pruefungen_ViewEntries) / sizeof(ViewEntry),
    (const ViewEntry*)&Pruefungen_ViewEntries
}
```

5.9.1.26 const DescriptionEntry Pruefungsstatus_DescriptionEntries[]

Initialisierung:

```
{
    {"status_typ", "Status", "Prüfungsstatus-Typ"},
    {"status_txt", "Beschreibung", "Beschreibung"},
    {"status_gewertet_bestanden", "Bestanden?", "Prüfung wird als
        bestanden gewertet"},
    {"status_gewertet_nichtbestanden", "Nicht bestanden?", "Prüfung wird als
        nicht bestanden gewertet"},
    {"typ_prfstatus", "Prüfungsstatus", "Tupel der Prüfungsstatus-Tabelle"}
}
```

5.9.1.27 const ViewEntry Pruefungsstatus_ViewEntries[]

Initialisierung:

```
{
    {
        "Alle Pruefungsstatus", "typ_prfstatus", "status_typ",
        0, NULL,
        0, NULL,
        sizeof(Pruefungsstatus_DescriptionEntries) / sizeof(DescriptionEntry),
        (const DescriptionEntry*)&Pruefungsstatus_DescriptionEntries
    }
}
```

5.9.1.28 const ViewTable Pruefungsstatus_ViewTable

Initialisierung:

```
{
    {"status_typ", NULL, NULL, NULL, NULL},
    "typ_prfstatus",
    "Pixmaps/Prüfungsstatus.xpm",
    sizeof(Pruefungsstatus_ViewEntries) / sizeof(ViewEntry),
    (const ViewEntry*)&Pruefungsstatus_ViewEntries
}
```

5.9.1.29 const DescriptionEntry Pruefungstypen_DescriptionEntries[]

Initialisierung:


```
{
  {"prf_typ", "Typ",          "Prüfungs-Typ"},
  {"prf_txt", "Beschreibung", "Beschreibung"},
  {"typ_pruefung", "Prüfungstyp", "Tupel der Prüfungstypen-Tabelle"}
}
```

5.9.1.30 const ViewEntry Pruefungstypen_ViewEntries[]

Initialisierung:

```
{
  {
    "Alle Pruefungstypen", "typ_pruefung", "prf_typ",
    0, NULL,
    0, NULL,
    sizeof(Pruefungstypen_DescriptionEntries) / sizeof(DescriptionEntry),
    (const DescriptionEntry*)&Pruefungstypen_DescriptionEntries
  }
}
```

5.9.1.31 const ViewTable Pruefungstypen_ViewTable

Initialisierung:

```
{
  {"prf_typ", NULL, NULL, NULL, NULL},
  "typ_pruefung",
  "Pixmaps/Prüfungstypen.xpm",
  sizeof(Pruefungstypen_ViewEntries) / sizeof(ViewEntry),
  (const ViewEntry*)&Pruefungstypen_ViewEntries
}
```

Sichtentabelle für Pruefungstypen.

5.9.1.32 const DescriptionEntry Qualifikattypen_DescriptionEntries[]

Initialisierung:

```
{
  {"qal_typ", "Art",          "Qualifikations-Art"},
  {"qal_grp", "Gruppe",      "Gruppe"},
  {"qal_txt", "Beschreibung", "Beschreibung"},
  {"typ_qualifikate", "Qualifikation", "Tupel der Qualifikattypen-Tabelle"}
}
```

5.9.1.33 const ViewEntry Qualifikattypen_ViewEntries[]

Initialisierung:

```
{
  {
```

```

        "Alle Qualifikattypen", "typ_qualifikate", "qal_grp, qal_txt",
        0, NULL,
        0, NULL,
        sizeof(Qualifikattypen_DescriptionEntries) / sizeof(DescriptionEntry),
        (const DescriptionEntry*)&Qualifikattypen_DescriptionEntries
    }
}

```

5.9.1.34 const ViewTable Qualifikattypen_ViewTable

Initialisierung:

```

{
    {"qal_typ", NULL, NULL, NULL, NULL},
    "typ_qualifikate",
    "Pixmaps/Qualifikattypen.xpm",
    sizeof(Qualifikattypen_ViewEntries) / sizeof(ViewEntry),
    (const ViewEntry*)&Qualifikattypen_ViewEntries
}

```

Sichtentabelle für Qualifikattypen.

5.9.1.35 const DescriptionEntry Semestertext_DescriptionEntries[]

Initialisierung:

```

{
    {"semester_txt", "Beschreibung", "Beschreibung"}
}

```

5.9.1.36 const ViewEntry Semestertext_ViewEntries[]

Initialisierung:

```

{
    {
        "Alle Semestertexte", "typ_semestertext", "semester_txt",
        0, NULL,
        0, NULL,
        sizeof(Semestertext_DescriptionEntries) / sizeof(DescriptionEntry),
        (const DescriptionEntry*)&Semestertext_DescriptionEntries
    }
}

```

5.9.1.37 const ViewTable Semestertext_ViewTable

Initialisierung:

```

{
    {"semester_txt", NULL, NULL, NULL, NULL},
    "typ_semestertext",
}

```

```

    "Pixmaps/Semestertext.xpm",
    sizeof(Semestertext_ViewEntries) / sizeof(ViewEntry),
    (const ViewEntry*)&Semestertext_ViewEntries
}

```

5.9.1.38 const DescriptionEntry Statistiken_DescriptionEntries[]

5.9.1.39 const ViewEntry Statistiken_ViewEntries[]

5.9.1.40 const ViewTable Statistiken_ViewTable

Initialisierung:

```

{
    {NULL, NULL, NULL, NULL, NULL},
    "(null)",
    "Pixmaps/Statistik.xpm",
    sizeof(Statistiken_ViewEntries) / sizeof(ViewEntry),
    (const ViewEntry*)&Statistiken_ViewEntries
}

```

Sichtentabelle für Statistiken.

5.9.1.41 const DescriptionEntry Studenten_DescriptionEntries[]

Initialisierung:

```

{
    {"stud_id",      "MatrNr",      "Matrikelnummer"},
    {"name",        "Name",        "Nachname"},
    {"vorname",     "Vorname",     "Vorname"},
    {"m_w",         "M/W",         "Männlich/Weiblich"},
    {"geb_datum",   "GebDatum",   "Geburtsdatum"},
    {"geb_ort",     "GebOrt",     "Geburtsort"},
    {"semester",   "Semester",   "Semester"},
    {"immat_datum", "Immatrikulation", "Datum der Immatrikulation"},
    {"exmat_datum", "Exmatrikulation", "Datum der Exmatrikulation"},
    {"nebenfach",   "Nebenfach",   "Nebenfach"},
    {"pers_studenten", "Student",   "Tupel der Studenten-Tabelle"}
}

```

5.9.1.42 const JoinEntry Studenten_JoinEntries[]

Initialisierung:

```

{
    {"stud_id", "stud_id", "Leistungsnachweise", &Leistungsnachweise_ViewTable},
    {"stud_id", "stud_id", "Prüfungen", &Pruefungen_ViewTable},
    {"stud_id", "stud_id", "Diplomarbeit", &Diplomarbeiten_ViewTable},
    {"stud_id", "id", "Adressen", &Adressen_ViewTable},
    {"stud_id", "stud_id", "Studium", &Studium_ViewTable}
}

```

5.9.1.43 const ViewEntry Studenten_ViewEntries[]**5.9.1.44 const ViewTable Studenten_ViewTable****Initialisierung:**

```
{
    {"stud_id", NULL, NULL, NULL, NULL},
    "pers_studenten",
    "Pixmaps/Studenten.xpm",
    sizeof(Studenten_ViewEntries) / sizeof(ViewEntry),
    (const ViewEntry*)&Studenten_ViewEntries
}
```

5.9.1.45 const DescriptionEntry Studientypen_DescriptionEntries[]**Initialisierung:**

```
{
    {"studium_typ", "Studientyp", "Studientyp"},
    {"studium_txt", "Beschreibung", "Beschreibung"},
    {"typ_studium", "Studientyp", "Tupel der Studientypen-Tabelle"}
}
```

5.9.1.46 const ViewEntry Studientypen_ViewEntries[]**Initialisierung:**

```
{
    {
        "Alle Studiumstypen", "typ_studium", "studium_typ",
        0, NULL,
        0, NULL,
        sizeof(Studientypen_DescriptionEntries) / sizeof(DescriptionEntry),
        (const DescriptionEntry*)&Studientypen_DescriptionEntries
    }
}
```

5.9.1.47 const ViewTable Studientypen_ViewTable**Initialisierung:**

```
{
    {"studium_typ", NULL, NULL, NULL, NULL},
    "typ_studium",
    "Pixmaps/Studientypen.xpm",
    sizeof(Studientypen_ViewEntries) / sizeof(ViewEntry),
    (const ViewEntry*)&Studientypen_ViewEntries
}
```

5.9.1.48 const DescriptionEntry Studium_DescriptionEntries[]

Initialisierung:

```
{
  {"stud_id",      "MatrNr",      "Matrikelnummer"},
  {"semester",    "Semester",    "Semester"},
  {"studien_typ", "Studientyp", "Studientyp"},
  {"uni_studium", "Studium",     "Tupel der Studium-Tabelle"}
}
```

5.9.1.49 const SelectEntry Studium_SelectEntries[]

Initialisierung:

```
{
  {"studien_typ", "studium_typ", "studium_txt",      true, &
   Studentypen_ViewTable},
  {"semester",    "semester_txt", "semester_txt",    true, &
   Semestertext_ViewTable},
  {"stud_id",     "stud_id",      "vorname, name, stud_id", false, &
   Studenten_ViewTable}
}
```

5.9.1.50 const ViewEntry Studium_ViewEntries[]

Initialisierung:

```
{
  {
    "Gesamt", "uni_studium", "semester",
    0, NULL,
    sizeof(Studium_SelectEntries) / sizeof(SelectEntry),
    (const SelectEntry*)&Studium_SelectEntries,
    sizeof(Studium_DescriptionEntries) / sizeof(DescriptionEntry),
    (const DescriptionEntry*)&Studium_DescriptionEntries
  }
}
```

5.9.1.51 const ViewTable Studium_ViewTable

Initialisierung:

```
{
  {"stud_id", "semester", NULL, NULL, NULL},
  "uni_studium",
  "Pixmaps/Studium.xpm",
  sizeof(Studium_ViewEntries) / sizeof(ViewEntry),
  (const ViewEntry*)&Studium_ViewEntries
}
```

5.10 paviews.h-Dateireferenz

```
#include "system.h" #include "paclient.h" #include <qstring.h>
```

Klassen

- struct [JoinEntry](#)
Join Entry.
- struct [SelectEntry](#)
Select Entry.
- struct [DescriptionEntry](#)
Description Entry.
- struct [ViewEntry](#)
View Entry.
- struct [ViewTable](#)
View Table.
- struct [OptimizeTable](#)
Optimize Table.

Variablen

- const [ViewTable Pruefer_ViewTable](#)
- const [ViewTable Studenten_ViewTable](#)
- const [ViewTable Statistiken_ViewTable](#)
- const [ViewTable Qualifikattypen_ViewTable](#)
- const [ViewTable Pruefungstypen_ViewTable](#)
- const [OptimizeTable PADB_OptimizeTable](#)

5.10.1 Variablen-Dokumentation

5.10.1.1 const [OptimizeTable PADB_OptimizeTable](#)

Optimierungstabelle.

5.10.1.2 const [ViewTable Pruefer_ViewTable](#)

Sichtentabelle für Prüfer.

5.10.1.3 const [ViewTable Pruefungstypen_ViewTable](#)

Sichtentabelle für Pruefungstypen.

5.10.1.4 const ViewTable Qualifikattypen_ViewTable

Sichtentabelle für Qualifikattypen.

5.10.1.5 const ViewTable Statistiken_ViewTable

Sichtentabelle für Statistiken.

5.10.1.6 const ViewTable Studenten_ViewTable

Sichtentabelle für Studenten.

5.11 sqlconnection.cc-Dateireferenz

```
#include "system.h" #include "sqlconnection.h" #include <qstring.h>
```

5.12 sqlconnection.h-Dateireferenz

```
#include "system.h" #include "sqlconnectioninterface.h" ×  
#include <libpq++.h>
```

Klassen

- class [SQLConnection](#)
SQL Connection.

5.13 sqlconnectioninterface.h-Dateireferenz

```
#include "system.h" #include "sqlexception.h" #include "sqlmonitorinterface.h"
```

Klassen

- class [SQLConnectionInterface](#)
SQL Connection Interface.

5.14 sqlexception.cc-Dateireferenz

```
#include "system.h" #include "sqlexception.h"
```

Funktionen

- ostream & [operator<<](#) (ostream &os, const [SQLException](#) e)

5.14.1 Dokumentation der Funktionen

5.14.1.1 ostream& operator<< (ostream & os, const SQLException e)

Ausgabe-Operator.

5.15 sqlexception.h-Dateireferenz

```
#include "system.h"
```

Klassen

- class [SQLException](#)
SQL Exception.

5.16 sqlmessages.cc-Dateireferenz

```
#include "system.h" #include "sqlmessages.h"
```

Funktionen

- void [sqlWarning](#) (QWidget *widget, const QString &text)
- void [sqlError](#) (QWidget *widget, const QString &text)

5.16.1 Dokumentation der Funktionen

5.16.1.1 void sqlError (QWidget * widget, const QString & text)

Ausgabe eines SQL-Fehlers.

Parameter

<i>widget</i>	Qt-Widget.
<i>text</i>	Fehler-Text.

5.16.1.2 void sqlWarning (QWidget * widget, const QString & text)

Ausgabe einer SQL-Warnung.

Parameter

<i>widget</i>	Qt-Widget.
<i>text</i>	Warnungs-Text.

5.17 sqlmessages.h-Dateireferenz

```
#include "system.h" #include <qapp.h> #include <qmessagebox.-
h>
```

Funktionen

- void [sqlWarning](#) (QWidget *widget, const QString &text)
- void [sqlError](#) (QWidget *widget, const QString &text)

5.17.1 Dokumentation der Funktionen

5.17.1.1 void sqlError (QWidget * widget, const QString & text)

Ausgabe eines SQL-Fehlers.

Parameter

<i>widget</i>	Qt-Widget.
<i>text</i>	Fehler-Text.

5.17.1.2 void sqlWarning (QWidget * widget, const QString & text)

Ausgabe einer SQL-Warnung.

Parameter

<i>widget</i>	Qt-Widget.
<i>text</i>	Warnungs-Text.

5.18 sqlmonitorinterface.h-Dateireferenz

```
#include "system.h"
```

Klassen

- class [SQLMonitorInterface](#)
SQL Monitor Interface.

5.19 system.h-Dateireferenz

```
#include <stdio.h> #include <stdlib.h> #include <unistd.-  
h> #include <errno.h> #include <iostream.h> #include  
<string.h> #include <math.h> #include <endian.h>
```

Makrodefinitionen

- #define [_THREAD_SAFE](#)
- #define [_GNU_SOURCE](#)
- #define [USE_PTHREADS](#)
- #define [CPU_BYTEORDER](#) [__BYTE_ORDER](#)

Typdefinitionen

- typedef signed char [sbyte](#)
- typedef unsigned char [ubyte](#)
- typedef signed char [int8](#)
- typedef unsigned char [card8](#)
- typedef signed short [int16](#)
- typedef unsigned short [card16](#)
- typedef signed int [int32](#)
- typedef signed int [integer](#)
- typedef unsigned int [card32](#)
- typedef signed long long [int64](#)
- typedef unsigned long long [card64](#)
- typedef unsigned int [cardinal](#)

5.19.1 Makro-Dokumentation

5.19.1.1 #define [_GNU_SOURCE](#)

5.19.1.2 #define [_THREAD_SAFE](#)

5.19.1.3 #define [CPU_BYTEORDER](#) [__BYTE_ORDER](#)

5.19.1.4 #define [USE_PTHREADS](#)

5.19.2 Dokumentation der benutzerdefinierten Typen**5.19.2.1 typedef unsigned short card16**

Datatype for storing a 16-bit cardinal.

5.19.2.2 typedef unsigned int card32

Datatype for storing a 32-bit cardinal.

5.19.2.3 typedef unsigned long long card64

Datatype for storing a 64-bit cardinal.

5.19.2.4 typedef unsigned char card8

Datatype for storing a 8-bit cardinal.

5.19.2.5 typedef unsigned int cardinal

Datatype for storing a default-sized cardinal (32 bits minimum).

5.19.2.6 typedef signed short int16

Datatype for storing a 16-bit integer.

5.19.2.7 typedef signed int int32

Datatype for storing a 32-bit integer.

5.19.2.8 typedef signed long long int64

Datatype for storing an 64-bit integer.

5.19.2.9 typedef signed char int8

Datatype for storing an 8-bit integer.

5.19.2.10 typedef signed int integer

Datatype for storing a default-sized integer (32 bits minimum).

5.19.2.11 typedef signed char sbyte

Datatype for storing a signed char.

5.19.2.12 typedef unsigned char ubyte

Datatype for storing an unsigned char.

5.20 tableviewer.cc-Dateireferenz

```
#include "system.h" #include "tableviewer.h" #include
"sqlmessages.h" #include <qheader.h> #include <qpushbutton.-
h> #include <qcombobox.h> #include <qhbuttongroup.h> ×
#include <qlayout.h> #include "tableviewer.moc"
```

Funktionen

- [QString trim](#) (const QString &string)

5.20.1 Dokumentation der Funktionen

5.20.1.1 QString trim (const QString & string)

5.21 tableviewer.h-Dateireferenz

```
#include "system.h" #include "paclient.h" #include "paviews.-
h" #include "sqlconnectioninterface.h" #include "tupleeditor.-
h" #include <qapp.h> #include <qlineedit.h> #include
<qlabel.h> #include <qwhatsthis.h> #include <qlistview.-
h> #include <qstring.h> #include <qlist.h> #include <qdict.-
h>
```

Klassen

- class [TableView](#)
Table Viewer.

5.22 textmonitor.cc-Dateireferenz

```
#include "system.h" #include "textmonitor.h" #include
<qapp.h> #include <qlayout.h> #include <qpushbutton.h>
#include <qmultilineedit.h> #include "textmonitor.moc"
```

5.23 textmonitor.h-Dateireferenz

```
#include "system.h"      #include "sqlmonitorinterface.h" ×  
#include <qapp.h> #include <qmultilineedit.h>
```

Klassen

- class [TextMonitor](#)
Text Monitor.

5.24 tupleeditor.cc-Dateireferenz

```
#include "system.h"      #include "tupleeditor.h"      #include  
"sqlmessages.h" #include <qlayout.h> #include <qpushbutton.-  
h> #include <qtabwidget.h> #include <qhbuttongroup.h> ×  
#include "tupleeditor.moc"
```

5.25 tupleeditor.h-Dateireferenz

```
#include "system.h" #include "paclient.h" #include "tableviewer.-  
h" #include "attributeeditor.h" #include <qapp.h> #include  
<qstring.h> #include <qlabel.h> #include <qwhatsthis.h>
```

Klassen

- struct [Tuple](#)
Tuple.
- class [TupleEditor](#)
Tuple Editor.