

eFinancialPlanner.eu

1

Generated by Doxygen 1.8.6

Wed Nov 18 2015 22:59:22

Contents

1	eFinancialPlanner Prototype V1.0	1
1.1	Introduction	1
1.2	Compilation	1
1.3	Depends on	1
2	Namespace Index	3
2.1	Namespace List	3
3	Hierarchical Index	5
3.1	Class Hierarchy	5
4	Class Index	7
4.1	Class List	7
5	File Index	9
5.1	File List	9
6	Namespace Documentation	11
6.1	bootstrap Namespace Reference	11
6.1.1	Function Documentation	11
6.1.1.1	bs_modal	11
6.1.1.2	bs_navbar_end	11
6.1.1.3	bs_navbar_end1	11
6.1.1.4	bs_navbar_end2	11
6.1.1.5	bs_navbar_item	12
6.1.1.6	bs_navbar_start	12
7	Class Documentation	13
7.1	aGOAL Struct Reference	13
7.1.1	Detailed Description	14
7.1.2	Constructor & Destructor Documentation	14
7.1.2.1	aGOAL	14
7.1.3	Member Function Documentation	15
7.1.3.1	add_to_db	15

7.1.3.2	save	15
7.1.3.3	save_results	15
7.1.3.4	set_amount	15
7.1.3.5	set_currency	15
7.1.3.6	set_description	15
7.1.3.7	set_frequency	15
7.1.3.8	set_from_date	15
7.1.3.9	set_from_db	15
7.1.3.10	set_goal_type	15
7.1.3.11	set_max_shortfall	15
7.1.3.12	set_priority	16
7.1.3.13	set_realization_age	16
7.1.3.14	set_till_date	16
7.1.3.15	set_to_pstmnt	16
7.1.3.16	show_graph_till_MNbr	16
7.1.4	Friends And Related Function Documentation	16
7.1.4.1	operator<<	16
7.1.5	Member Data Documentation	16
7.1.5.1	alpha	16
7.1.5.2	amount	16
7.1.5.3	benchmark	16
7.1.5.4	color	17
7.1.5.5	color_followup	17
7.1.5.6	currency	17
7.1.5.7	description	17
7.1.5.8	frequency	17
7.1.5.9	from_date	17
7.1.5.10	goal_id	17
7.1.5.11	goal_type	17
7.1.5.12	investor	17
7.1.5.13	max_shortfall	17
7.1.5.14	means_goal	17
7.1.5.15	optimal_portf	18
7.1.5.16	priority	18
7.1.5.17	realization_age	18
7.1.5.18	realization_monthNbr	18
7.1.5.19	realized_shortfall	18
7.1.5.20	remarks	18
7.1.5.21	s_exp	18
7.1.5.22	s_goal	18

7.1.5.23	s_high	18
7.1.5.24	s_low	18
7.1.5.25	saving_plan	18
7.1.5.26	simulation_endvalue	19
7.1.5.27	simulation_string	19
7.1.5.28	success	19
7.1.5.29	target_amount	19
7.1.5.30	till_date	19
7.2	asset Class Reference	19
7.2.1	Detailed Description	20
7.2.2	Member Function Documentation	20
7.2.2.1	asset_delete	20
7.2.2.2	exists	20
7.2.2.3	list_assets	20
7.2.2.4	load	20
7.2.2.5	save	20
7.2.2.6	set_amount	20
7.2.2.7	set_asset_class	20
7.2.2.8	set_asset_id	21
7.2.2.9	set_currency	21
7.2.2.10	set_description	21
7.2.2.11	set_investor	21
7.2.2.12	show_edit_form	21
7.2.3	Member Data Documentation	21
7.2.3.1	amount	21
7.2.3.2	asset_class	21
7.2.3.3	asset_id	21
7.2.3.4	currency	21
7.2.3.5	description	21
7.2.3.6	investor	21
7.3	asset_class Class Reference	22
7.3.1	Detailed Description	22
7.3.2	Member Function Documentation	22
7.3.2.1	dropDown_assetClass	22
7.3.2.2	exists	22
7.3.2.3	get_assetList	22
7.3.3	Member Data Documentation	22
7.3.3.1	asset_class_id	22
7.3.3.2	asset_class_name	22
7.3.3.3	E_R	22

7.4	bona Class Reference	23
7.4.1	Detailed Description	23
7.4.2	Member Function Documentation	24
7.4.2.1	d_get	24
7.4.2.2	delete_bona	24
7.4.2.3	exists	24
7.4.2.4	f_get	24
7.4.2.5	i_get	24
7.4.2.6	list	24
7.4.2.7	load_fromDB	24
7.4.2.8	load_fromEnv	24
7.4.2.9	load_fromRecordSet	24
7.4.2.10	s_get	24
7.4.2.11	save	24
7.4.2.12	show	25
7.4.2.13	show_edit_form	25
7.4.2.14	show_tc	25
7.4.2.15	show_th	25
7.4.2.16	verify	25
7.4.3	Member Data Documentation	25
7.4.3.1	action_ref	25
7.4.3.2	class_name	25
7.4.3.3	defList	25
7.4.3.4	order_by	25
7.4.3.5	tbl	25
7.4.3.6	tbl_name	26
7.5	cash_flow Class Reference	26
7.5.1	Detailed Description	26
7.5.2	Constructor & Destructor Documentation	27
7.5.2.1	cash_flow	27
7.5.3	Member Function Documentation	27
7.5.3.1	delete_bona	27
7.5.3.2	exists	27
7.5.3.3	list	27
7.5.3.4	load_fromDB	27
7.5.3.5	load_fromEnv	27
7.5.3.6	save	27
7.5.3.7	show	27
7.5.3.8	show_edit_form	27
7.5.4	Member Data Documentation	28

7.5.4.1	action_ref	28
7.5.4.2	class_name	28
7.5.4.3	defList	28
7.5.4.4	order_by	28
7.5.4.5	tbl	28
7.5.4.6	tbl_name	28
7.6	cls_currency Class Reference	28
7.6.1	Detailed Description	29
7.6.2	Member Function Documentation	29
7.6.2.1	conversion_factor	29
7.6.2.2	dropDown_currency	29
7.6.2.3	set_curr	29
7.6.2.4	to_curr	29
7.6.3	Member Data Documentation	29
7.6.3.1	currency_id	29
7.6.3.2	currency_name	29
7.6.3.3	entity	29
7.7	cls_frequency Class Reference	29
7.7.1	Detailed Description	30
7.7.2	Member Function Documentation	30
7.7.2.1	get_freq_label	30
7.7.2.2	set_freq	30
7.7.3	Member Data Documentation	30
7.7.3.1	frequency_id	30
7.7.3.2	multiplier	30
7.8	column Class Reference	30
7.8.1	Detailed Description	31
7.8.2	Member Function Documentation	31
7.8.2.1	get_dValue	31
7.8.2.2	get_fValue	31
7.8.2.3	get_iValue	31
7.8.2.4	get_sValue	31
7.8.2.5	quote	32
7.8.2.6	set_value	32
7.8.2.7	set_value	32
7.8.2.8	set_value	32
7.8.2.9	show	32
7.8.2.10	show_inputField	32
7.8.3	Member Data Documentation	32
7.8.3.1	col_type	32

7.8.3.2	column_name	32
7.8.3.3	data_align	32
7.8.3.4	drop_down_default	32
7.8.3.5	drop_down_keyField	32
7.8.3.6	drop_down_keyFieldQuote	32
7.8.3.7	drop_down_showField	33
7.8.3.8	drop_down_table	33
7.8.3.9	dValue	33
7.8.3.10	fValue	33
7.8.3.11	input_type	33
7.8.3.12	iValue	33
7.8.3.13	label	33
7.8.3.14	placeholder	33
7.8.3.15	post_code	33
7.8.3.16	show_with_previous	33
7.8.3.17	sValue	33
7.9	config Class Reference	34
7.9.1	Detailed Description	34
7.9.2	Member Data Documentation	35
7.9.2.1	css_dir	35
7.9.2.2	db_host	35
7.9.2.3	db_name	35
7.9.2.4	db_pwd	35
7.9.2.5	db_user	35
7.9.2.6	font_combination	35
7.9.2.7	img_dir	35
7.9.2.8	language	35
7.9.2.9	layout	35
7.9.2.10	merge_ER_vol	35
7.9.2.11	parse_wrapper	36
7.9.2.12	riskFunction	36
7.9.2.13	scripts_dir	36
7.9.2.14	show_aside	36
7.9.2.15	show_social	36
7.9.2.16	simul_show_table	36
7.9.2.17	soft_dir	36
7.9.2.18	soft_file	36
7.9.2.19	soft_name	36
7.9.2.20	soft_server_url	36
7.9.2.21	soft_url	37

7.9.2.22	soft_version	37
7.9.2.23	tbl_prefix	37
7.9.2.24	total_portf_monthNbrs	37
7.9.2.25	wrapper_name	37
7.9.2.26	wrapper_url	37
7.9.2.27	www_dir	37
7.10	db_helper Class Reference	37
7.10.1	Detailed Description	38
7.10.2	Constructor & Destructor Documentation	38
7.10.2.1	db_helper	38
7.10.2.2	~db_helper	38
7.10.3	Member Function Documentation	38
7.10.3.1	getFloat	38
7.10.3.2	getInt	38
7.10.3.3	runSQL	38
7.10.4	Member Data Documentation	38
7.10.4.1	con	38
7.10.4.2	db	38
7.10.4.3	driver	38
7.10.4.4	pstmt	39
7.10.4.5	res	39
7.10.4.6	res2	39
7.10.4.7	stmt	39
7.11	follow_up Class Reference	39
7.11.1	Detailed Description	41
7.11.2	Constructor & Destructor Documentation	41
7.11.2.1	follow_up	41
7.11.3	Member Function Documentation	42
7.11.3.1	add_to_db	42
7.11.3.2	age	42
7.11.3.3	age2monthNbr	42
7.11.3.4	dateStr2Age	42
7.11.3.5	exists	42
7.11.3.6	get_description	42
7.11.3.7	get_from_db	42
7.11.3.8	get_full_name	42
7.11.3.9	get_max_month_for_lower_goals	43
7.11.3.10	get_nbr_goals	43
7.11.3.11	get_nbrMonths2simulate	43
7.11.3.12	get_portf_from_db	43

7.11.3.13 javaGraph	43
7.11.3.14 load_covar	43
7.11.3.15 load_ER	43
7.11.3.16 load_from_db	44
7.11.3.17 load_preferences	44
7.11.3.18 months2simulate	44
7.11.3.19 parse_dashboard	44
7.11.3.20 parse_javaGraph	44
7.11.3.21 plot_market_evol	44
7.11.3.22 prepare_javaVars_colors	44
7.11.3.23 reload_covar	45
7.11.3.24 reload_ER	45
7.11.3.25 save	45
7.11.3.26 save_covar	45
7.11.3.27 save_ER	45
7.11.3.28 save_preferences	45
7.11.3.29 set_followup_color_post	46
7.11.3.30 set_followup_color_prae	46
7.11.3.31 set_max_exposure	46
7.11.3.32 set_mu	46
7.11.3.33 set_scale	46
7.11.3.34 set_sigma	46
7.11.3.35 show_simulation	46
7.11.3.36 simulate	46
7.11.3.37 solve	47
7.11.4 Member Data Documentation	47
7.11.4.1 assetClass_covar	47
7.11.4.2 assetClass_mu	47
7.11.4.3 assetClass_name	47
7.11.4.4 birth_date	47
7.11.4.5 covar	48
7.11.4.6 currency	48
7.11.4.7 description	48
7.11.4.8 desirability	48
7.11.4.9 ER	48
7.11.4.10 experience	48
7.11.4.11 first_name	48
7.11.4.12 goalZ	48
7.11.4.13 investor_id	48
7.11.4.14 knowledge	48

7.11.4.15 last_name	48
7.11.4.16 market_return	48
7.11.4.17 max_exposure	49
7.11.4.18 nbrMonths	49
7.11.4.19 password	49
7.11.4.20 pLogR	49
7.11.4.21 pMu	49
7.11.4.22 portfolio_id	49
7.11.4.23 portfolios	49
7.11.4.24 pSigma	49
7.11.4.25 simulate_till_age	49
7.11.4.26 user_name	49
7.11.4.27 weights	50
7.12 goal Class Reference	50
7.12.1 Detailed Description	50
7.12.2 Constructor & Destructor Documentation	51
7.12.2.1 goal	51
7.12.3 Member Function Documentation	51
7.12.3.1 delete_bona	51
7.12.3.2 exists	51
7.12.3.3 list	51
7.12.3.4 load_fromDB	51
7.12.3.5 load_fromEnv	51
7.12.3.6 save	51
7.12.3.7 show	51
7.12.3.8 show_edit_form	51
7.12.4 Member Data Documentation	52
7.12.4.1 action_ref	52
7.12.4.2 class_name	52
7.12.4.3 defList	52
7.12.4.4 order_by	52
7.12.4.5 tbl	52
7.12.4.6 tbl_name	52
7.13 html_helper Class Reference	52
7.13.1 Detailed Description	54
7.13.2 Member Typedef Documentation	54
7.13.2.1 STEPDEF	54
7.13.2.2 STRMAP	54
7.13.3 Constructor & Destructor Documentation	54
7.13.3.1 html_helper	54

7.13.4	Member Function Documentation	54
7.13.4.1	aside	54
7.13.4.2	collapse_close	54
7.13.4.3	collapse_open	54
7.13.4.4	cookie_policy	55
7.13.4.5	cout_box_color	55
7.13.4.6	footer	55
7.13.4.7	get_box_color_class	55
7.13.4.8	get_glyphicon	55
7.13.4.9	get_next_step	55
7.13.4.10	get_next_sub_step	55
7.13.4.11	get_prev_step	55
7.13.4.12	get_prev_sub_step	55
7.13.4.13	get_step_nbr	56
7.13.4.14	get_string	56
7.13.4.15	get_sub_step_nbr	56
7.13.4.16	header	56
7.13.4.17	header_menu_bar	56
7.13.4.18	navbar_box	56
7.13.4.19	navbar_end	56
7.13.4.20	navbar_item	57
7.13.4.21	navbar_start	57
7.13.4.22	next_step	57
7.13.4.23	parse_breadcrumb	57
7.13.4.24	parse_date_time	57
7.13.4.25	parse_title	57
7.13.4.26	privacy_policy	57
7.13.4.27	show_args	57
7.13.4.28	show_home	57
7.13.4.29	show_social	58
7.13.4.30	show_toolBar	58
7.13.4.31	sub_step_exists	58
7.13.4.32	terms_of_use	58
7.13.5	Member Data Documentation	58
7.13.5.1	assetClassColors	58
7.13.5.2	c_arrow	58
7.13.5.3	c_content_header	58
7.13.5.4	c_doc_type	58
7.13.5.5	c_photos_phdb	58
7.13.5.6	c_photos_phdb_dir	59

7.13.5.7	c_right_menu_names	59
7.13.5.8	c_right_menu_urls	59
7.13.5.9	c_salt	59
7.13.5.10	scale125i2s	59
7.13.5.11	seriesColorsGoalPlot	59
7.13.5.12	step_nbrs	59
7.13.5.13	steps	59
7.13.5.14	sub_step_nbrs	59
7.13.5.15	sub_steps	60
7.14	investment_problem Class Reference	60
7.14.1	Detailed Description	63
7.14.2	Constructor & Destructor Documentation	63
7.14.2.1	investment_problem	63
7.14.3	Member Function Documentation	64
7.14.3.1	add_to_db	64
7.14.3.2	age	64
7.14.3.3	age2monthNbr	64
7.14.3.4	allocate_to_unallocated_goal	64
7.14.3.5	calc_alpha	64
7.14.3.6	calc_risk	64
7.14.3.7	calc_V_high	65
7.14.3.8	dateStr2Age	65
7.14.3.9	exists	65
7.14.3.10	expand_cf	65
7.14.3.11	get_description	65
7.14.3.12	get_from_db	65
7.14.3.13	get_full_name	65
7.14.3.14	get_max_month_for_lower_goals	66
7.14.3.15	get_nbr_goals	66
7.14.3.16	get_optimal_portfolio	66
7.14.3.17	get_portf_from_db	66
7.14.3.18	get_portfolio_riskiest	66
7.14.3.19	get_portfolio_safest	66
7.14.3.20	goal_seek_means	66
7.14.3.21	goal_seek_tmp_means	67
7.14.3.22	javaGraph	67
7.14.3.23	load_covar	67
7.14.3.24	load_ER	67
7.14.3.25	load_from_db	67
7.14.3.26	load_preferences	68

7.14.3.27 months2simulate	68
7.14.3.28 parse_javaGraph	68
7.14.3.29 prepare_javaVars_colors	68
7.14.3.30 reload_covar	68
7.14.3.31 reload_ER	68
7.14.3.32 save	69
7.14.3.33 save_covar	69
7.14.3.34 save_ER	69
7.14.3.35 save_preferences	69
7.14.3.36 set_assets	69
7.14.3.37 set_cfs	69
7.14.3.38 set_followup_color_post	69
7.14.3.39 set_followup_color_prae	70
7.14.3.40 set_goal_color	70
7.14.3.41 set_goal_remarks	70
7.14.3.42 set_goals	70
7.14.3.43 set_max_exposure	70
7.14.3.44 set_means_goal	70
7.14.3.45 set_mu	71
7.14.3.46 set_portfolios	71
7.14.3.47 set_priorityLimits	71
7.14.3.48 set_scale	71
7.14.3.49 set_sigma	71
7.14.3.50 set_unallocated_goal	71
7.14.3.51 solve	71
7.14.4 Member Data Documentation	72
7.14.4.1 assetClass_covar	72
7.14.4.2 assetClass_mu	72
7.14.4.3 assetClass_name	72
7.14.4.4 birth_date	72
7.14.4.5 covar	72
7.14.4.6 currency	72
7.14.4.7 description	73
7.14.4.8 desirability	73
7.14.4.9 ER	73
7.14.4.10 experience	73
7.14.4.11 first_name	73
7.14.4.12 goalZ	73
7.14.4.13 investor_id	73
7.14.4.14 knowledge	73

7.14.4.15 last_name	73
7.14.4.16 max_exposure	73
7.14.4.17 means	73
7.14.4.18 means_block	74
7.14.4.19 means_left	74
7.14.4.20 means_tmp	74
7.14.4.21 means_used	74
7.14.4.22 nbr_assets	74
7.14.4.23 nbr_cfs	74
7.14.4.24 nbr_goals	74
7.14.4.25 nbr_portfolios	74
7.14.4.26 password	74
7.14.4.27 pLogR	75
7.14.4.28 pMu	75
7.14.4.29 portfolio_id	75
7.14.4.30 portfolio_suitable	75
7.14.4.31 portfolios	75
7.14.4.32 priorityMax	75
7.14.4.33 priorityMin	75
7.14.4.34 pSigma	75
7.14.4.35 simulate_till_age	75
7.14.4.36 user_name	75
7.14.4.37 weights	75
7.15 investor Class Reference	76
7.15.1 Detailed Description	77
7.15.2 Constructor & Destructor Documentation	77
7.15.2.1 investor	77
7.15.3 Member Function Documentation	77
7.15.3.1 add_to_db	77
7.15.3.2 age	78
7.15.3.3 age2monthNbr	78
7.15.3.4 dateStr2Age	78
7.15.3.5 exists	78
7.15.3.6 get_from_db	78
7.15.3.7 get_full_name	78
7.15.3.8 load_covar	78
7.15.3.9 load_ER	78
7.15.3.10 load_from_db	79
7.15.3.11 load_preferences	79
7.15.3.12 months2simulate	79

7.15.3.13 reload_covar	79
7.15.3.14 reload_ER	79
7.15.3.15 save	79
7.15.3.16 save_covar	80
7.15.3.17 save_ER	80
7.15.3.18 save_preferences	80
7.15.3.19 set_max_exposure	80
7.15.3.20 set_scale	80
7.15.4 Member Data Documentation	80
7.15.4.1 assetClass_covar	80
7.15.4.2 assetClass_mu	80
7.15.4.3 assetClass_name	80
7.15.4.4 birth_date	81
7.15.4.5 covar	81
7.15.4.6 currency	81
7.15.4.7 desirability	81
7.15.4.8 ER	81
7.15.4.9 experience	81
7.15.4.10 first_name	81
7.15.4.11 investor_id	81
7.15.4.12 knowledge	81
7.15.4.13 last_name	81
7.15.4.14 max_exposure	81
7.15.4.15 password	81
7.15.4.16 simulate_till_age	82
7.15.4.17 user_name	82
7.16 investor_ui Class Reference	82
7.16.1 Detailed Description	85
7.16.2 Constructor & Destructor Documentation	85
7.16.2.1 investor_ui	85
7.16.3 Member Function Documentation	85
7.16.3.1 account_form	85
7.16.3.2 account_save	85
7.16.3.3 add_to_db	85
7.16.3.4 age	85
7.16.3.5 age2monthNbr	85
7.16.3.6 asset_add	86
7.16.3.7 asset_delete	86
7.16.3.8 asset_edit	86
7.16.3.9 asset_save	86

7.16.3.10 button_to	86
7.16.3.11 cash_flow_add	86
7.16.3.12 cash_flow_delete	86
7.16.3.13 cash_flow_edit	86
7.16.3.14 cash_flow_list	86
7.16.3.15 cash_flow_save	86
7.16.3.16 dateStr2Age	87
7.16.3.17 exists	87
7.16.3.18 feedback	87
7.16.3.19 get_from_db	87
7.16.3.20 get_full_name	87
7.16.3.21 goal_add	87
7.16.3.22 goal_delete	87
7.16.3.23 goal_edit	87
7.16.3.24 goal_help_form	87
7.16.3.25 goal_help_save	88
7.16.3.26 goal_list	88
7.16.3.27 goal_save	88
7.16.3.28 home_screen	88
7.16.3.29 is_logged_in	88
7.16.3.30 load_covar	88
7.16.3.31 load_ER	88
7.16.3.32 load_from_db	88
7.16.3.33 load_preferences	89
7.16.3.34 login	89
7.16.3.35 login_exec	89
7.16.3.36 login_form	89
7.16.3.37 logout	89
7.16.3.38 months2simulate	89
7.16.3.39 parse_bm	89
7.16.3.40 parse_form_close_personalize	89
7.16.3.41 parse_form_open_personalize	90
7.16.3.42 parse_goal_box	90
7.16.3.43 parse_savings_plan	90
7.16.3.44 parse_simulation_duration_form	90
7.16.3.45 parse_simulation_insight	90
7.16.3.46 parse_user_info	90
7.16.3.47 personalize_covar_table	90
7.16.3.48 personalize_ER_table	90
7.16.3.49 personalize_ERvol_table	90

7.16.3.50 personalize_form	91
7.16.3.51 personalize_pref_table	91
7.16.3.52 personalize_reload	91
7.16.3.53 personalize_save	91
7.16.3.54 personalize_screen	91
7.16.3.55 personalize_set_corr	91
7.16.3.56 personalize_set_ER	91
7.16.3.57 personalize_set_general	91
7.16.3.58 personalize_set_prefs	91
7.16.3.59 personalize_set_vol	92
7.16.3.60 personalize_vol_table	92
7.16.3.61 register_exec	92
7.16.3.62 register_form	92
7.16.3.63 reload_covar	92
7.16.3.64 reload_ER	92
7.16.3.65 save	92
7.16.3.66 save_covar	93
7.16.3.67 save_ER	93
7.16.3.68 save_preferences	93
7.16.3.69 set_birth_date	93
7.16.3.70 set_currency	93
7.16.3.71 set_first_name	93
7.16.3.72 set_investor_id	93
7.16.3.73 set_investor_id_fromEnv	93
7.16.3.74 set_last_name	93
7.16.3.75 set_max_exposure	94
7.16.3.76 set_password	94
7.16.3.77 set_scale	94
7.16.3.78 set_simulate_till_age	94
7.16.3.79 set_user_name	94
7.16.3.80 show_account_form_body	94
7.16.3.81 show_disclaimer	94
7.16.3.82 show_follow_up	94
7.16.3.83 show_goal_help_form_body	95
7.16.3.84 show_home_screen_form	95
7.16.3.85 show_inventory_form	95
7.16.3.86 show_inventory_form_body	95
7.16.3.87 show_login_form_body	95
7.16.3.88 show_register_form_body	95
7.16.3.89 show_simulation	95

7.16.3.90	show_total_portfolio	96
7.16.3.91	update_last_activity_at	96
7.16.4	Member Data Documentation	96
7.16.4.1	active_action	96
7.16.4.2	asset_class_names	96
7.16.4.3	assetClass_covar	96
7.16.4.4	assetClass_mu	96
7.16.4.5	assetClass_name	96
7.16.4.6	birth_date	96
7.16.4.7	covar	96
7.16.4.8	currency	97
7.16.4.9	desirability	97
7.16.4.10	ER	97
7.16.4.11	experience	97
7.16.4.12	first_name	97
7.16.4.13	hh	97
7.16.4.14	investor_id	97
7.16.4.15	knowledge	97
7.16.4.16	last_name	97
7.16.4.17	max_exposure	97
7.16.4.18	password	97
7.16.4.19	simulate_till_age	97
7.16.4.20	user_name	98
7.17	market Class Reference	98
7.17.1	Detailed Description	99
7.17.2	Constructor & Destructor Documentation	99
7.17.2.1	market	99
7.17.3	Member Function Documentation	99
7.17.3.1	load_covar	99
7.17.3.2	load_ER	99
7.17.3.3	reload_covar	99
7.17.3.4	reload_ER	99
7.17.3.5	save_covar	100
7.17.3.6	save_ER	100
7.17.4	Member Data Documentation	100
7.17.4.1	assetClass_covar	100
7.17.4.2	assetClass_mu	100
7.17.4.3	assetClass_name	100
7.17.4.4	covar	100
7.17.4.5	ER	100

7.18 portfolio Class Reference	100
7.18.1 Detailed Description	101
7.18.2 Member Function Documentation	101
7.18.2.1 get_description	101
7.18.2.2 get_portf_from_db	101
7.18.2.3 set_mu	102
7.18.2.4 set_sigma	102
7.18.3 Member Data Documentation	102
7.18.3.1 description	102
7.18.3.2 pLogR	102
7.18.3.3 pMu	102
7.18.3.4 portfolio_id	102
7.18.3.5 pSigma	102
7.18.3.6 weights	102
7.19 simulation Class Reference	102
7.19.1 Detailed Description	105
7.19.2 Constructor & Destructor Documentation	105
7.19.2.1 simulation	105
7.19.3 Member Function Documentation	105
7.19.3.1 add_to_db	105
7.19.3.2 age	105
7.19.3.3 age2monthNbr	105
7.19.3.4 dateStr2Age	106
7.19.3.5 exists	106
7.19.3.6 get_description	106
7.19.3.7 get_from_db	106
7.19.3.8 get_full_name	106
7.19.3.9 get_max_month_for_lower_goals	106
7.19.3.10 get_nbr_goals	106
7.19.3.11 get_nbrMonths2simulate	106
7.19.3.12 get_portf_from_db	107
7.19.3.13 javaGraph	107
7.19.3.14 load_covar	107
7.19.3.15 load_ER	107
7.19.3.16 load_from_db	107
7.19.3.17 load_preferences	107
7.19.3.18 months2simulate	107
7.19.3.19 parse_javaGraph	108
7.19.3.20 plot_market_evol	108
7.19.3.21 prepare_javaVars_colors	108

7.19.3.22 reload_covar	108
7.19.3.23 reload_ER	108
7.19.3.24 save	109
7.19.3.25 save_covar	109
7.19.3.26 save_ER	109
7.19.3.27 save_preferences	109
7.19.3.28 set_followup_color_post	109
7.19.3.29 set_followup_color_prae	109
7.19.3.30 set_max_exposure	109
7.19.3.31 set_mu	110
7.19.3.32 set_nbrMonths2simulate_from_env	110
7.19.3.33 set_scale	110
7.19.3.34 set_sigma	110
7.19.3.35 show_simulation	110
7.19.3.36 simulate	110
7.19.3.37 simulate_market	110
7.19.3.38 simulate_portfolios	110
7.19.3.39 solve	111
7.19.4 Member Data Documentation	111
7.19.4.1 assetClass_covar	111
7.19.4.2 assetClass_mu	111
7.19.4.3 assetClass_name	111
7.19.4.4 birth_date	111
7.19.4.5 covar	111
7.19.4.6 currency	112
7.19.4.7 description	112
7.19.4.8 desirability	112
7.19.4.9 ER	112
7.19.4.10 experience	112
7.19.4.11 first_name	112
7.19.4.12 goalZ	112
7.19.4.13 investor_id	112
7.19.4.14 knowledge	112
7.19.4.15 last_name	112
7.19.4.16 market_return	112
7.19.4.17 max_exposure	113
7.19.4.18 nbrMonths	113
7.19.4.19 password	113
7.19.4.20 pLogR	113
7.19.4.21 pMu	113

7.19.4.22 portfolio_id	113
7.19.4.23 portfolios	113
7.19.4.24 pSigma	113
7.19.4.25 simulate_till_age	113
7.19.4.26 user_name	113
7.19.4.27 weights	114
8 File Documentation	115
8.1 aGOAL.struct.cpp File Reference	115
8.1.1 Function Documentation	115
8.1.1.1 operator<<	115
8.2 aGOAL.struct.cpp	115
8.3 asset.class.cpp File Reference	120
8.4 asset.class.cpp	120
8.5 asset_class.class.cpp File Reference	124
8.6 asset_class.class.cpp	124
8.7 bona.class.cpp File Reference	125
8.8 bona.class.cpp	125
8.9 bona_cf.class.cpp File Reference	130
8.10 bona_cf.class.cpp	130
8.11 bona_goal.class.cpp File Reference	132
8.12 bona_goal.class.cpp	132
8.13 bootstrap.namespace.cpp File Reference	133
8.14 bootstrap.namespace.cpp	134
8.15 column.class.cpp File Reference	135
8.16 column.class.cpp	135
8.17 config.class.cpp File Reference	137
8.17.1 Macro Definition Documentation	137
8.17.1.1 LOCAL	137
8.18 config.class.cpp	137
8.19 currency.class.cpp File Reference	138
8.20 currency.class.cpp	138
8.21 db_helper.class.cpp File Reference	139
8.22 db_helper.class.cpp	140
8.23 efp.cpp File Reference	142
8.23.1 Macro Definition Documentation	144
8.23.1.1 ALPHA_LOWER_LIMIT	144
8.23.1.2 ALPHA_PLOT_HIGH	144
8.23.1.3 ALPHA_PLOT_LOW	144
8.23.1.4 ALPHA_UPPER_LIMIT	145

8.23.1.5	CURR_LEN	145
8.23.1.6	DEFAULT_SCALE	145
8.23.1.7	INDEX	145
8.23.1.8	INPUT_WIDTH_FLOAT	145
8.23.1.9	INPUT_WIDTH_INT	145
8.23.1.10	INPUT_WIDTH_TEXT	145
8.23.1.11	MAX_AGE	145
8.23.1.12	MAX_ALPHA	145
8.23.1.13	MAX_MNTHS_2_SIMULATE	146
8.23.1.14	MAX_MONTHS_TO_SAFEST	146
8.23.1.15	MAX_SCALE	146
8.23.1.16	MIN_ALPHA	146
8.23.1.17	MIN_MNTHS_2_SIMULATE	146
8.23.1.18	MIN_SCALE	146
8.23.1.19	MIN_YEARS_TO_SIMULATE	146
8.23.1.20	NBR_ASSET_CLASSES	146
8.23.1.21	NBR_ITERATIONS	146
8.23.1.22	NBR_RIGHT_MENU	147
8.23.1.23	NBR_SCALE	147
8.23.1.24	NBR_SUB_TABS	147
8.23.1.25	NBR_TABS	147
8.23.1.26	PRECISSION	147
8.23.1.27	RAINY_DAY_MONTH_NBR	147
8.23.1.28	RETIREMENT_AGE	147
8.23.1.29	SECONDS_IN_MONTH	147
8.23.1.30	SIMULATE_TILL_DEFAULT	147
8.23.2	Function Documentation	148
8.23.2.1	main	148
8.23.3	Variable Documentation	148
8.23.3.1	cgi	148
8.23.3.2	db	148
8.23.3.3	error_message	149
8.23.3.4	goal_type_i2s	149
8.23.3.5	goal_type_s2i	149
8.23.3.6	hh	149
8.23.3.7	oAsset	149
8.23.3.8	oAssetClass	149
8.23.3.9	oCF	149
8.23.3.10	oConfig	150
8.23.3.11	oCurrency	150

8.23.3.12 oGoal	150
8.23.3.13 oInvestor	150
8.23.3.14 std_message	150
8.24 efp.cpp	150
8.25 follow_up.class.cpp File Reference	153
8.26 follow_up.class.cpp	153
8.27 frequency.class.cpp File Reference	153
8.28 frequency.class.cpp	153
8.29 global_functions.h File Reference	154
8.29.1 Function Documentation	155
8.29.1.1 add0	155
8.29.1.2 curr_format	155
8.29.1.3 currFormat	155
8.29.1.4 date_format_dateStr	155
8.29.1.5 date_format_tm	155
8.29.1.6 dateStr2Mnbr	155
8.29.1.7 dateStr2tm	155
8.29.1.8 flSum	156
8.29.1.9 get_tm	156
8.29.1.10 is_valid_dateStr	156
8.29.1.11 mNbr2dateStr	156
8.29.1.12 months2yms	156
8.29.1.13 round2str	156
8.29.1.14 thousand_separator	156
8.29.1.15 tm2DateStr	156
8.29.1.16 tm2Mnbr	157
8.29.2 Variable Documentation	157
8.29.2.1 Pi	157
8.30 global_functions.h	157
8.31 html_helper.class.cpp File Reference	160
8.32 html_helper.class.cpp	160
8.33 investment_problem.class.cpp File Reference	175
8.34 investment_problem.class.cpp	175
8.35 investor.class.cpp File Reference	190
8.36 investor.class.cpp	190
8.37 investor_ui.class.cpp File Reference	194
8.38 investor_ui.class.cpp	195
8.39 market.class.cpp File Reference	223
8.40 market.class.cpp	224
8.41 normdist.h File Reference	227

8.41.1	Macro Definition Documentation	227
8.41.1.1	HIGH	227
8.41.1.2	LOW	228
8.41.2	Function Documentation	228
8.41.2.1	erfinv	228
8.41.2.2	fast_erfinv	228
8.41.2.3	norminv	228
8.41.2.4	phi	228
8.41.3	Variable Documentation	229
8.41.3.1	a	229
8.41.3.2	b	229
8.41.3.3	c	229
8.41.3.4	d	229
8.42	normdist.h	230
8.43	portfolio.class.cpp File Reference	231
8.44	portfolio.class.cpp	231
8.45	simulation.class.cpp File Reference	233
8.46	simulation.class.cpp	233
8.47	t_disclaimer_Eng_UK.h File Reference	236
8.47.1	Variable Documentation	236
8.47.1.1	t_disclaimer	236
8.48	t_disclaimer_Eng_UK.h	236
8.49	t_personalize_Eng_UK.h File Reference	237
8.49.1	Function Documentation	238
8.49.1.1	init_personalize	238
8.49.2	Variable Documentation	238
8.49.2.1	t_Desirability	238
8.49.2.2	t_Experience	238
8.49.2.3	t_info_bdate	238
8.49.2.4	t_info_curr	238
8.49.2.5	t_info_simtill	238
8.49.2.6	t_Knowledge	238
8.49.2.7	t_Max_Exp	238
8.49.2.8	t_personalize	238
8.49.2.9	t_personalize1	238
8.49.2.10	t_personalize_concepts	239
8.49.2.11	t_personalize_scoring	239
8.49.2.12	t_reload	239
8.50	t_personalize_Eng_UK.h	239
8.51	test.js File Reference	240

8.51.1	Function Documentation	240
8.51.1.1	ready	241
8.52	test.js	241
8.53	texts_Eng_UK.h File Reference	241
8.53.1	Variable Documentation	242
8.53.1.1	t_after_title	242
8.53.1.2	t_and	243
8.53.1.3	t_aside	243
8.53.1.4	t_Asset_Class	243
8.53.1.5	t_author	243
8.53.1.6	t_b_currency	243
8.53.1.7	t_b_date	243
8.53.1.8	t_before_list	243
8.53.1.9	t_breadcrumb	244
8.53.1.10	t_cancel	244
8.53.1.11	t_clear_form	244
8.53.1.12	t_collapse	244
8.53.1.13	t_currency	244
8.53.1.14	t_dashboard	244
8.53.1.15	t_delete	244
8.53.1.16	t_edit	244
8.53.1.17	t_errMsg	244
8.53.1.18	t_extra_info	244
8.53.1.19	t_extraInfo	245
8.53.1.20	t_feedback	246
8.53.1.21	t_feedback_post	246
8.53.1.22	t_hide_extra_info	247
8.53.1.23	t_Login	247
8.53.1.24	t_month	247
8.53.1.25	t_months	247
8.53.1.26	t_next_step	247
8.53.1.27	t_plotlabels_evol	247
8.53.1.28	t_prev_step	247
8.53.1.29	t_regFrm	247
8.53.1.30	t_Register	248
8.53.1.31	t_reset	248
8.53.1.32	t_save	248
8.53.1.33	t_scale125i2s	248
8.53.1.34	t_show_extra_info	248
8.53.1.35	t_showhide_extra_info	248

8.53.1.36 t_sim_till	248
8.53.1.37 t_simulation_duration	248
8.53.1.38 t_title	248
8.53.1.39 t_titles	249
8.53.1.40 t_warning	249
8.53.1.41 t_year	249
8.53.1.42 t_years	249
8.54 texts_Eng_UK.h	249
Index	253

Chapter 1

eFinancialPlanner Prototype V1.0

Philippe J.S. De Brouwer

1.1 Introduction

This software is a first effort to implement Maslowian Portfolio Theory in a fully automated way.

Maslowian Portfolio Theory has been proposed by Prof. Dr. Philippe J.S. De Brouwer in 2008 (Journal of Asset Management, Vol.9, 6, pp. 359–365). This theory is the first effort since Markovitz' Modern Portfolio Theory from 1952 to propose a prescriptive portfolio theory that allows for multi-goal investing (a phenomenon also called "mental accounting") that was well described in the descriptive theory "Behavioural Portfolio Theory" of Hersh Shefrin and Meir Statman in 2001.

The theory ignores the modern myth of "the investor's risk profile". The concept that an investor should have one risk profile as used by Markovitz as a working hypothesis is based on magical thinking and results rather from the need of having such concept than from a real theory underpinning it.

The idea is to start from the investor and his/her world with his/her priorities in life. It is from this life-goals and their parameters that we derive the necessary parameters to find an optimal investment portfolio per investment goal.

1.2 Compilation

```
g++ -Wall -I/usr/include/cppconn efp.cpp -o OUTFILE -lmysqlcppconn -lcgicc
```

or optimal compilation:

```
g++ -Wall -I/usr/include/cppconn efp.cpp -o /usr/lib/cgi-bin/$OUTFILE -lmysqlcppconn /usr/lib/libcgicc.a -std=c++11 -finline-small-functions
```

1.3 Depends on

This software is based on an open source stack. Compiled it runs on a linux (Debian) system and via Apache it provides HTML5 to the browser and uses CSS3 to present the data. It should be fairly straightforward to make it work on other platforms.

- Linux operating system
- gnu C++ compiler collection
 - apt-get install build-essential manpages-dev -y
 - apt-get install cmake -y

- MySQL databae
apt-get install mysql-server -y
apt-get install mysql-client -y
- Apache2 web server
sudo apt-get install apache2
sudo a2enmod cgi
sudo service apache2 restart
- msql connector: mysqlcppconn
sudo apt-get install libmysqlcppconn-dev
- cgicc
sudo apt-get install libcgicc5-dev
- useful:
apt-get install doxygen -y

Chapter 2

Namespace Index

2.1 Namespace List

Here is a list of all namespaces with brief descriptions:

[bootstrap](#) 11

Chapter 3

Hierarchical Index

3.1 Class Hierarchy

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

- aGOAL 13
- asset 19
- asset_class 22
- bona 23
 - cash_flow 26
 - goal 50
- cls_currency 28
- cls_frequency 29
- column 30
- config 34
- db_helper 37
- html_helper 52
- market 98
 - investor 76
 - investment_problem 60
 - simulation 102
 - follow_up 39
 - investor_ui 82
- portfolio 100
 - investment_problem 60

Chapter 4

Class Index

4.1 Class List

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

aGOAL	13
asset	19
asset_class	22
bona	23
cash_flow	26
cls_currency	28
cls_frequency	29
column	30
config	34
db_helper	37
follow_up	39
goal	50
html_helper	52
investment_problem	60
investor	76
investor_ui	82
market	98
portfolio	100
simulation	102

Chapter 5

File Index

5.1 File List

Here is a list of all files with brief descriptions:

aGOAL.struct.cpp	115
asset.class.cpp	120
asset_class.class.cpp	124
bona.class.cpp	125
bona_cf.class.cpp	130
bona_goal.class.cpp	132
bootstrap.namespace.cpp	133
column.class.cpp	135
config.class.cpp	137
currency.class.cpp	138
db_helper.class.cpp	139
efp.cpp	142
follow_up.class.cpp	153
frequency.class.cpp	153
global_functions.h	154
html_helper.class.cpp	160
investment_problem.class.cpp	175
investor.class.cpp	190
investor_ui.class.cpp	194
market.class.cpp	223
normdist.h	227
portfolio.class.cpp	231
simulation.class.cpp	233
t_disclaimer_Eng_UK.h	236
t_personalize_Eng_UK.h	237
test.js	240
texts_Eng_UK.h	241

Chapter 6

Namespace Documentation

6.1 bootstrap Namespace Reference

Functions

- void [bs_modal](#) (string title, string content, string id="myModal", string b1_text="", string b1_url="")
- void [bs_navbar_start](#) ()
- void [bs_navbar_item](#) (string my_text="", string my_url="#", bool is_active=false)
- void [bs_navbar_end](#) ()
- void [bs_navbar_end1](#) ()
- void [bs_navbar_end2](#) ()

6.1.1 Function Documentation

6.1.1.1 void bootstrap::bs_modal (string title, string content, string id = "myModal", string b1_text = "", string b1_url = "") [inline]

bs_modal

bootstrap::modal -> generates a modal box within the bootstrap framework

Definition at line 17 of file [bootstrap.namespace.cpp](#).

6.1.1.2 void bootstrap::bs_navbar_end () [inline]

bs_navbar_end

closes the environment in which the nav-bar can be put

Definition at line 89 of file [bootstrap.namespace.cpp](#).

6.1.1.3 void bootstrap::bs_navbar_end1 () [inline]

bs_navbar_end1

closes the environment in which the nav-bar can be put

Definition at line 69 of file [bootstrap.namespace.cpp](#).

6.1.1.4 void bootstrap::bs_navbar_end2 () [inline]

bs_navbar_end2

closes the environment in which the nav-bar can be put

Definition at line 78 of file [bootstrap.namespace.cpp](#).

6.1.1.5 void bootstrap::bs_navbar_item (string *my_text* = "", string *my_url* = "#", bool *is_active* = false) [inline]

bs_navbar_item

shows one item in a navigation bar

Definition at line 99 of file [bootstrap.namespace.cpp](#).

6.1.1.6 void bootstrap::bs_navbar_start () [inline]

bs_navbar_start

opens the environment in which the nav-bar can be p

Definition at line 44 of file [bootstrap.namespace.cpp](#).

Chapter 7

Class Documentation

7.1 aGOAL Struct Reference

Public Member Functions

- `aGOAL` (int investor_id=0)
constructor
- void `set_from_db` ()
- void `set_to_pstmt` ()
- bool `save` ()
- bool `add_to_db` ()
- void `save_results` ()
- bool `set_goal_type` (string g_type)
saves the benchmark and savings plan, !!! TODO
- bool `set_amount` (string amnt)
Needs goal_type to be set first.
- bool `set_description` (string desc)
- bool `set_currency` (string curr)
- bool `set_priority` (string imp)
check if exists, if not then add the default
- bool `set_from_date` (string f_date)
- bool `set_till_date` (string t_date)
- bool `set_frequency` (string freq)
- bool `set_realization_age` (string real_age, float actual_age)
- bool `set_max_shortfall` (string shortfall)
- int `show_graph_till_MNbr` ()

Public Attributes

- int `goal_id`
variables stored in database before `investment_problem.solve()`:
- int `investor` = 0
- int `goal_type`
(0, "unallocated"), (1, "amount@date"), (2, "income from/to"), (3, "rainy day savings"), (4, "amount asap");
- float `amount`
- float `target_amount`
target_amount is for goal_type amount
- `std::map< int, float >` `means_goal`

the means implied by the goal (eg. the regular income during retirement for goal type income from/to)

- string `description`
- string `currency`
- int `priority`
- string `from_date`
- string `till_date`
- string `frequency`
- float `realization_age`
- float `max_shortfall`
- float `alpha` = 0.1

variables calculated and allocated during `investment_problem.solve()` BEFORE solution:

- int `realization_monthNbr`
- `std::map< int, float >` `saving_plan`

variables calculated and allocated during `investment_problem.solve()` AFTER solution:

- int `optimal_portf`
- float `realized_shortfall`
- `std::map< int, float >` `benchmark`
- `std::string` `remarks`
- string `color`
- string `color_followup`
- bool `success`
- string `simulation_string` = ""

a string containing the simulated values of the advised portfolio [45,1000], [45,083, 1100], etc.]]

- float `simulation_endvalue` = 0

the last value obtained by the simulation (only used in the Follow-Up screens to allow the evolution to start from the end-value of the simulation)

- string `s_low` = ""

javaScript variable for the lower quantile as [[45,100], [45.1,101],...]

- string `s_exp` = ""
- string `s_high` = ""

javaScript variable for the expected evolution

- string `s_goal` = ""

javaScript variable for the upper quantile

Friends

- ostream & `operator<<` (ostream &os, const `aGOAL` &the_goal)

other functions:

7.1.1 Detailed Description

struct `aGOAL`

Definition at line 5 of file `aGOAL.struct.cpp`.

7.1.2 Constructor & Destructor Documentation

7.1.2.1 `aGOAL::aGOAL (int investor_id = 0)` [inline]

constructor

Definition at line 8 of file `aGOAL.struct.cpp`.

7.1.3 Member Function Documentation

7.1.3.1 bool aGOAL::add_to_db ()

Definition at line 280 of file [aGOAL.struct.cpp](#).

7.1.3.2 bool aGOAL::save ()

save

Definition at line 327 of file [aGOAL.struct.cpp](#).

7.1.3.3 void aGOAL::save_results ()

7.1.3.4 bool aGOAL::set_amount (string *amnt*)

Needs goal_type to be set first.

Definition at line 142 of file [aGOAL.struct.cpp](#).

7.1.3.5 bool aGOAL::set_currency (string *curr*)

Definition at line 148 of file [aGOAL.struct.cpp](#).

7.1.3.6 bool aGOAL::set_description (string *desc*)

Definition at line 163 of file [aGOAL.struct.cpp](#).

7.1.3.7 bool aGOAL::set_frequency (string *freq*)

Definition at line 208 of file [aGOAL.struct.cpp](#).

7.1.3.8 bool aGOAL::set_from_date (string *f_date*)

Definition at line 223 of file [aGOAL.struct.cpp](#).

7.1.3.9 void aGOAL::set_from_db ()

set_from_db

assumes that a result set of my_sql is active and loads its content in the object < (0, "unallocated"), (1, "amount@date"), (2, "income from/to"), (3, "rainy day savings"), (4, "amount asap");

Definition at line 362 of file [aGOAL.struct.cpp](#).

7.1.3.10 bool aGOAL::set_goal_type (string *g_type*)

saves the benchmark and savings plan, !!! TODO

Definition at line 126 of file [aGOAL.struct.cpp](#).

7.1.3.11 bool aGOAL::set_max_shortfall (string *shortfall*)

Definition at line 268 of file [aGOAL.struct.cpp](#).

7.1.3.12 `bool aGOAL::set_priority (string imp)`

check if exists, if not then add the default

Definition at line 177 of file [aGOAL.struct.cpp](#).

7.1.3.13 `bool aGOAL::set_realization_age (string real_age, float actual_age)`

`set_realization_age`

this will also set the `from_date = to_date` and frequency NOT to be used for a income-type-goal!!

Definition at line 189 of file [aGOAL.struct.cpp](#).

7.1.3.14 `bool aGOAL::set_till_date (string t_date)`

`set_till_date`

assumes that `from_date` is already set and valid!!!!

Definition at line 242 of file [aGOAL.struct.cpp](#).

7.1.3.15 `void aGOAL::set_to_pstmnt ()`

Definition at line 381 of file [aGOAL.struct.cpp](#).

7.1.3.16 `int aGOAL::show_graph_till_MNbr ()`

7.1.4 Friends And Related Function Documentation

7.1.4.1 `ostream& operator<< (ostream & os, const aGOAL & the_goal)` [`friend`]

other functions:

overload the `<<` operator to output the goal in a readable format

Definition at line 80 of file [aGOAL.struct.cpp](#).

7.1.5 Member Data Documentation

7.1.5.1 `float aGOAL::alpha = 0.1`

variables calculated and allocated during `investment_problem.solve()` BEFORE solution:

Definition at line 29 of file [aGOAL.struct.cpp](#).

7.1.5.2 `float aGOAL::amount`

Definition at line 14 of file [aGOAL.struct.cpp](#).

7.1.5.3 `std::map<int, float> aGOAL::benchmark`

Definition at line 35 of file [aGOAL.struct.cpp](#).

7.1.5.4 string aGOAL::color

Definition at line 37 of file [aGOAL.struct.cpp](#).

7.1.5.5 string aGOAL::color_followup

Definition at line 38 of file [aGOAL.struct.cpp](#).

7.1.5.6 string aGOAL::currency

Definition at line 20 of file [aGOAL.struct.cpp](#).

7.1.5.7 string aGOAL::description

Definition at line 19 of file [aGOAL.struct.cpp](#).

7.1.5.8 string aGOAL::frequency

Definition at line 24 of file [aGOAL.struct.cpp](#).

7.1.5.9 string aGOAL::from_date

Definition at line 22 of file [aGOAL.struct.cpp](#).

7.1.5.10 int aGOAL::goal_id

variables stored in database before [investment_problem.solve\(\)](#):

Definition at line 11 of file [aGOAL.struct.cpp](#).

7.1.5.11 int aGOAL::goal_type

(0, "unallocated"), (1, "amount@date"), (2, "income from/to"), (3, "rainy day savings"), (4, "amount asap");

Definition at line 13 of file [aGOAL.struct.cpp](#).

7.1.5.12 int aGOAL::investor = 0

Definition at line 12 of file [aGOAL.struct.cpp](#).

7.1.5.13 float aGOAL::max_shortfall

Definition at line 26 of file [aGOAL.struct.cpp](#).

7.1.5.14 std::map<int, float> aGOAL::means_goal

the means implied by the goal (eg. the regular income during retirement for goal type income from/to)

Definition at line 17 of file [aGOAL.struct.cpp](#).

7.1.5.15 `int aGOAL::optimal_portf`

Definition at line 33 of file [aGOAL.struct.cpp](#).

7.1.5.16 `int aGOAL::priority`

Definition at line 21 of file [aGOAL.struct.cpp](#).

7.1.5.17 `float aGOAL::realization_age`

Definition at line 25 of file [aGOAL.struct.cpp](#).

7.1.5.18 `int aGOAL::realization_monthNbr`

Definition at line 30 of file [aGOAL.struct.cpp](#).

7.1.5.19 `float aGOAL::realized_shortfall`

Definition at line 34 of file [aGOAL.struct.cpp](#).

7.1.5.20 `std::string aGOAL::remarks`

Definition at line 36 of file [aGOAL.struct.cpp](#).

7.1.5.21 `string aGOAL::s_exp = ""`

Definition at line 43 of file [aGOAL.struct.cpp](#).

7.1.5.22 `string aGOAL::s_goal = ""`

[javaScript](#) variable for the upper quantile

[javaScript](#) variable for the goal

Definition at line 45 of file [aGOAL.struct.cpp](#).

7.1.5.23 `string aGOAL::s_high = ""`

[javaScript](#) variable for the expected evolution

Definition at line 44 of file [aGOAL.struct.cpp](#).

7.1.5.24 `string aGOAL::s_low = ""`

[javaScript](#) variable for the lower quantile as [\[\[45,100\], \[45.1,101\],....\]](#)

Definition at line 42 of file [aGOAL.struct.cpp](#).

7.1.5.25 `std::map<int, float> aGOAL::saving_plan`

variables calculated and allocated during [investment_problem.solve\(\)](#) AFTER solution:

Definition at line 32 of file [aGOAL.struct.cpp](#).

7.1.5.26 float aGOAL::simulation_endvalue = 0

the last value obtained by the simulation (only used in the Follow-Up screens to allow the evolution to start from the end-value of the simulation)

Definition at line 41 of file [aGOAL.struct.cpp](#).

7.1.5.27 string aGOAL::simulation_string = ""

a string containing the simulated values of the advised portfolio [45,1000], [45,083, 1100], etc.]]

Definition at line 40 of file [aGOAL.struct.cpp](#).

7.1.5.28 bool aGOAL::success

Definition at line 39 of file [aGOAL.struct.cpp](#).

7.1.5.29 float aGOAL::target_amount

target_amount is for goal_type amount

for goal_type 1 this is the target amount, for goal_type 2 this is the monthly income, for goal_type 3 this is the fixe amount needed

Date

the same as amount, for "income" goal, it is zero, etc.

Definition at line 16 of file [aGOAL.struct.cpp](#).

7.1.5.30 string aGOAL::till_date

Definition at line 23 of file [aGOAL.struct.cpp](#).

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

- [aGOAL.struct.cpp](#)

7.2 asset Class Reference

Public Member Functions

- bool [set_asset_id](#) (const std::string &s)
- bool [set_investor](#) (const std::string &s)
- bool [set_asset_class](#) (const std::string &s)
- bool [set_description](#) (const std::string &s)
- bool [set_currency](#) (const std::string &s)
- bool [set_amount](#) (const std::string &s)
- void [list_assets](#) (int iid)
- void [show_edit_form](#) (int iid)
- bool [load](#) ()
- bool [save](#) ()
- bool [asset_delete](#) ()

Public Attributes

- int [asset_id](#)
- int [investor](#)
- int [asset_class](#)
- string [description](#)
- string [currency](#)
- float [amount](#)

Private Member Functions

- bool [exists](#) (int aid)

7.2.1 Detailed Description

Definition at line 11 of file [asset.class.cpp](#).

7.2.2 Member Function Documentation

7.2.2.1 bool [asset::asset_delete](#) ()

Definition at line 267 of file [asset.class.cpp](#).

7.2.2.2 bool [asset::exists](#) (int *aid*) [private]

Definition at line 257 of file [asset.class.cpp](#).

7.2.2.3 void [asset::list_assets](#) (int *iid*)

Definition at line 104 of file [asset.class.cpp](#).

7.2.2.4 bool [asset::load](#) ()

Definition at line 42 of file [asset.class.cpp](#).

7.2.2.5 bool [asset::save](#) ()

Definition at line 230 of file [asset.class.cpp](#).

7.2.2.6 bool [asset::set_amount](#) (const std::string & s)

Definition at line 204 of file [asset.class.cpp](#).

7.2.2.7 bool [asset::set_asset_class](#) (const std::string & s)

Definition at line 190 of file [asset.class.cpp](#).

7.2.2.8 `bool asset::set_asset_id (const std::string & s)`

`set_investor`

Definition at line 148 of file [asset.class.cpp](#).

7.2.2.9 `bool asset::set_currency (const std::string & s)`

Definition at line 218 of file [asset.class.cpp](#).

7.2.2.10 `bool asset::set_description (const std::string & s)`

Definition at line 178 of file [asset.class.cpp](#).

7.2.2.11 `bool asset::set_investor (const std::string & s)`

Definition at line 163 of file [asset.class.cpp](#).

7.2.2.12 `void asset::show_edit_form (int iid)`

Definition at line 63 of file [asset.class.cpp](#).

7.2.3 Member Data Documentation

7.2.3.1 `float asset::amount`

Definition at line 19 of file [asset.class.cpp](#).

7.2.3.2 `int asset::asset_class`

Definition at line 16 of file [asset.class.cpp](#).

7.2.3.3 `int asset::asset_id`

Definition at line 14 of file [asset.class.cpp](#).

7.2.3.4 `string asset::currency`

Definition at line 18 of file [asset.class.cpp](#).

7.2.3.5 `string asset::description`

Definition at line 17 of file [asset.class.cpp](#).

7.2.3.6 `int asset::investor`

Definition at line 15 of file [asset.class.cpp](#).

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

- [asset.class.cpp](#)

7.3 `asset_class` Class Reference

Public Member Functions

- string `dropDown_assetClass` (int defaultAC=0)
- bool `exists` (int acID)
- `std::map< int, string >` `get_assetList` ()

Public Attributes

- int `asset_class_id`
- string `asset_class_name`
- float `E_R`

7.3.1 Detailed Description

Definition at line 11 of file `asset_class.class.cpp`.

7.3.2 Member Function Documentation

7.3.2.1 `string asset_class::dropDown_assetClass (int defaultAC = 0)`

Definition at line 44 of file `asset_class.class.cpp`.

7.3.2.2 `bool asset_class::exists (int acID)`

Definition at line 31 of file `asset_class.class.cpp`.

7.3.2.3 `std::map< int, string > asset_class::get_assetList ()`

`get_assetList`

Definition at line 64 of file `asset_class.class.cpp`.

7.3.3 Member Data Documentation

7.3.3.1 `int asset_class::asset_class_id`

Definition at line 14 of file `asset_class.class.cpp`.

7.3.3.2 `string asset_class::asset_class_name`

Definition at line 15 of file `asset_class.class.cpp`.

7.3.3.3 `float asset_class::E_R`

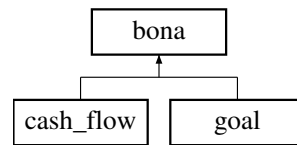
Definition at line 16 of file `asset_class.class.cpp`.

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

- `asset_class.class.cpp`

7.4 bona Class Reference

Inheritance diagram for bona:



Public Member Functions

- void [list](#) (int iid, bool editable=true)
- bool [save](#) ()
- void [show](#) (bool editable=true)
- void [show_edit_form](#) (int iid, bool showExisting=false)
- bool [delete_bona](#) ()
- bool [exists](#) ()
- bool [load_fromEnv](#) ()
- bool [load_fromDB](#) ()

Public Attributes

- std::map< int, [column](#) > [tbl](#)
- string [tbl_name](#) = oConfig.tbl_prefix + "_assets"
- string [class_name](#) = ""
- string [action_ref](#)
- std::map< string, int > [defList](#)

list holding the column names in the table connected to the correct id of the column

Protected Attributes

- string [order_by](#) = ""
- should be the column name eventually followed by DESC/ASC, eg. "priority ASC"*

Private Member Functions

- bool [verify](#) ()
- void [load_fromRecordSet](#) ()
- void [show_th](#) (bool editable)
- void [show_tc](#) ()
- string [s_get](#) (string lbl)
- float [f_get](#) (string lbl)
- int [i_get](#) (string lbl)
- struct tm [d_get](#) (string lbl)

7.4.1 Detailed Description

Definition at line 11 of file [bona.class.cpp](#).

7.4.2 Member Function Documentation

7.4.2.1 `struct tm bona::d_get (string lbl) [private]`

Definition at line 411 of file [bona.class.cpp](#).

7.4.2.2 `bool bona::delete_bona ()`

Definition at line 350 of file [bona.class.cpp](#).

7.4.2.3 `bool bona::exists ()`

Definition at line 382 of file [bona.class.cpp](#).

7.4.2.4 `float bona::f_get (string lbl) [private]`

Definition at line 409 of file [bona.class.cpp](#).

7.4.2.5 `int bona::i_get (string lbl) [private]`

Definition at line 410 of file [bona.class.cpp](#).

7.4.2.6 `void bona::list (int iid, bool editable = true)`

`void list(int investor_id);`

to trigger the scripts to fill out the dates and provide the date-picker

so javascripts can check if this exists to know if they are in the cash flow screen

Definition at line 157 of file [bona.class.cpp](#).

7.4.2.7 `bool bona::load_fromDB ()`

Definition at line 44 of file [bona.class.cpp](#).

7.4.2.8 `bool bona::load_fromEnv ()`

Definition at line 336 of file [bona.class.cpp](#).

7.4.2.9 `void bona::load_fromRecordSet () [private]`

Definition at line 60 of file [bona.class.cpp](#).

7.4.2.10 `string bona::s_get (string lbl) [private]`

Definition at line 408 of file [bona.class.cpp](#).

7.4.2.11 `bool bona::save ()`

Definition at line 276 of file [bona.class.cpp](#).

7.4.2.12 `void bona::show (bool editable = true)`

Definition at line 234 of file [bona.class.cpp](#).

7.4.2.13 `void bona::show_edit_form (int iid, bool showExisting = false)`

to trigger the scripts to fill out the dates and provide the date-picker

Definition at line 108 of file [bona.class.cpp](#).

7.4.2.14 `void bona::show_tc () [private]`

Definition at line 100 of file [bona.class.cpp](#).

7.4.2.15 `void bona::show_th (bool editable) [private]`

Definition at line 71 of file [bona.class.cpp](#).

7.4.2.16 `bool bona::verify () [private]`

Definition at line 369 of file [bona.class.cpp](#).

7.4.3 Member Data Documentation

7.4.3.1 `string bona::action_ref`

Definition at line 17 of file [bona.class.cpp](#).

7.4.3.2 `string bona::class_name = ""`

Definition at line 16 of file [bona.class.cpp](#).

7.4.3.3 `std::map<string, int> bona::defList`

list holding the column names in the table connected to the correct id of the column

Definition at line 26 of file [bona.class.cpp](#).

7.4.3.4 `string bona::order_by = "" [protected]`

should be the column name eventually followed by DESC/ASC, eg. "priority ASC"

Definition at line 28 of file [bona.class.cpp](#).

7.4.3.5 `std::map<int, column> bona::tbl`

Definition at line 14 of file [bona.class.cpp](#).

7.4.3.6 `string bona::tbl_name = oConfig.tbl_prefix + "_assets"`

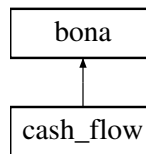
Definition at line 15 of file [bona.class.cpp](#).

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

- [bona.class.cpp](#)

7.5 cash_flow Class Reference

Inheritance diagram for `cash_flow`:



Public Member Functions

- [cash_flow](#) ()
- void [list](#) (int iid, bool editable=true)
- bool [save](#) ()
- void [show](#) (bool editable=true)
- void [show_edit_form](#) (int iid, bool showExisting=false)
- bool [delete_bona](#) ()
- bool [exists](#) ()
- bool [load_fromEnv](#) ()
- bool [load_fromDB](#) ()

Public Attributes

- `std::map< int, column >` [tbl](#)
- `string tbl_name = oConfig.tbl_prefix + "_assets"`
- `string class_name = ""`
- `string action_ref`
- `std::map< string, int >` [defList](#)

list holding the column names in the table connected to the correct id of the column

Protected Attributes

- `string order_by = ""`
should be the column name eventually followed by DESC/ASC, eg. "priority ASC"

7.5.1 Detailed Description

Definition at line 11 of file [bona_cf.class.cpp](#).

7.5.2 Constructor & Destructor Documentation

7.5.2.1 cash_flow::cash_flow ()

Definition at line 32 of file [bona_cf.class.cpp](#).

7.5.3 Member Function Documentation

7.5.3.1 bool bona::delete_bona () [inherited]

Definition at line 350 of file [bona.class.cpp](#).

7.5.3.2 bool bona::exists () [inherited]

Definition at line 382 of file [bona.class.cpp](#).

7.5.3.3 void bona::list (int iid, bool *editable* = true) [inherited]

void list(int investor_id);

to trigger the scripts to fill out the dates and provide the date-picker

so javascripts can check if this exists to know if they are in the cash flow screen

Definition at line 157 of file [bona.class.cpp](#).

7.5.3.4 bool bona::load_fromDB () [inherited]

Definition at line 44 of file [bona.class.cpp](#).

7.5.3.5 bool bona::load_fromEnv () [inherited]

Definition at line 336 of file [bona.class.cpp](#).

7.5.3.6 bool bona::save () [inherited]

Definition at line 276 of file [bona.class.cpp](#).

7.5.3.7 void bona::show (bool *editable* = true) [inherited]

Definition at line 234 of file [bona.class.cpp](#).

7.5.3.8 void bona::show_edit_form (int iid, bool *showExisting* = false) [inherited]

to trigger the scripts to fill out the dates and provide the date-picker

Definition at line 108 of file [bona.class.cpp](#).

7.5.4 Member Data Documentation

7.5.4.1 `string bona::action_ref` [inherited]

Definition at line 17 of file [bona.class.cpp](#).

7.5.4.2 `string bona::class_name = ""` [inherited]

Definition at line 16 of file [bona.class.cpp](#).

7.5.4.3 `std::map<string, int> bona::defList` [inherited]

list holding the column names in the table connected to the correct id of the column

Definition at line 26 of file [bona.class.cpp](#).

7.5.4.4 `string bona::order_by = ""` [protected], [inherited]

should be the column name eventually followed by DESC/ASC, eg. "priority ASC"

Definition at line 28 of file [bona.class.cpp](#).

7.5.4.5 `std::map<int, column> bona::tbl` [inherited]

Definition at line 14 of file [bona.class.cpp](#).

7.5.4.6 `string bona::tbl_name = oConfig.tbl_prefix + "_assets"` [inherited]

Definition at line 15 of file [bona.class.cpp](#).

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

- [bona_cf.class.cpp](#)

7.6 `cls_currency` Class Reference

Public Member Functions

- `string dropDown_currency` (string defaultCurr="EUR")
- `bool set_curr` (string the_curr)
- `float to_curr` (float amnt, string the_curr)
- `float conversion_factor` (string from_curr, string to_curr)

Public Attributes

- `string entity`
- `string currency_name`
- `string currency_id`
eg. EUR, USD, etc.

7.6.1 Detailed Description

Definition at line 1 of file [currency.class.cpp](#).

7.6.2 Member Function Documentation

7.6.2.1 float cls_currency::conversion_factor (string *from_curr*, string *to_curr*)

7.6.2.2 string cls_currency::dropDown_currency (string *defaultCurr* = "EUR")

Definition at line 20 of file [currency.class.cpp](#).

7.6.2.3 bool cls_currency::set_curr (string *the_curr*)

set_curr

checks if the currency_id given is a valid currency, if yes currency_id is set and true returned

Definition at line 41 of file [currency.class.cpp](#).

7.6.2.4 float cls_currency::to_curr (float *amnt*, string *to_curr*) [inline]

to_curr

converts a given amount to the given currency step 1: convert to EUR

step 2: if necessary convert from EUR to another currency

step 3: return the result

Definition at line 67 of file [currency.class.cpp](#).

7.6.3 Member Data Documentation

7.6.3.1 string cls_currency::currency_id

eg. EUR, USD, etc.

Definition at line 7 of file [currency.class.cpp](#).

7.6.3.2 string cls_currency::currency_name

Definition at line 6 of file [currency.class.cpp](#).

7.6.3.3 string cls_currency::entity

Definition at line 5 of file [currency.class.cpp](#).

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

- [currency.class.cpp](#)

7.7 cls_frequency Class Reference

Public Member Functions

- bool [set_freq](#) (string *the_freq*)
- string [get_freq_label](#) ()

Public Attributes

- string [frequency_id](#)
- float [multiplier](#)

7.7.1 Detailed Description

Definition at line 1 of file [frequency.class.cpp](#).

7.7.2 Member Function Documentation

7.7.2.1 string [cls_frequency::get_freq_label](#) ()

[get_freq_label](#)

returns a descriptive label for the

Definition at line 44 of file [frequency.class.cpp](#).

7.7.2.2 bool [cls_frequency::set_freq](#) (string *the_freq*)

[set_freq](#)

sets the [frequency_id](#)

Definition at line 18 of file [frequency.class.cpp](#).

7.7.3 Member Data Documentation

7.7.3.1 string [cls_frequency::frequency_id](#)

Definition at line 5 of file [frequency.class.cpp](#).

7.7.3.2 float [cls_frequency::multiplier](#)

Definition at line 6 of file [frequency.class.cpp](#).

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

- [frequency.class.cpp](#)

7.8 column Class Reference

Public Member Functions

- string [quote](#) ()
- int [get_iValue](#) ()
- string [get_sValue](#) ()

- float [get_fValue](#) ()
- struct tm [get_dValue](#) ()
- void [set_value](#) (int value)
- void [set_value](#) (string value)
- void [set_value](#) (float value)
- void [show_inputField](#) (int iid, bool showExisting=false)
- void [show](#) ()

Public Attributes

- string [column_name](#)
- char [col_type](#)
- string [input_type](#)
- string [drop_down_table](#)
- string [drop_down_keyField](#)
- string [drop_down_keyFieldQuote](#)
- string [drop_down_showField](#)
- string [drop_down_default](#)
- string [label](#)
- string [placeholder](#)
- string [post_code](#)
- bool [show_with_previous](#) = false
- string [data_align](#) = "center"

Private Attributes

- int [iValue](#)
- string [sValue](#)
- float [fValue](#)
- struct tm [dValue](#)

7.8.1 Detailed Description

Definition at line 6 of file [column.class.cpp](#).

7.8.2 Member Function Documentation

7.8.2.1 struct tm column::get_dValue ()

Definition at line 49 of file [column.class.cpp](#).

7.8.2.2 float column::get_fValue ()

Definition at line 48 of file [column.class.cpp](#).

7.8.2.3 int column::get_iValue ()

Definition at line 46 of file [column.class.cpp](#).

7.8.2.4 string column::get_sValue ()

Definition at line 47 of file [column.class.cpp](#).

7.8.2.5 string column::quote ()

Definition at line 71 of file [column.class.cpp](#).

7.8.2.6 void column::set_value (int *value*)

Definition at line 51 of file [column.class.cpp](#).

7.8.2.7 void column::set_value (string *value*)

Definition at line 56 of file [column.class.cpp](#).

7.8.2.8 void column::set_value (float *value*)

Definition at line 65 of file [column.class.cpp](#).

7.8.2.9 void column::show ()

Definition at line 79 of file [column.class.cpp](#).

7.8.2.10 void column::show_inputField (int *iid*, bool *showExisting* = false)

Definition at line 99 of file [column.class.cpp](#).

7.8.3 Member Data Documentation

7.8.3.1 char column::col_type

Definition at line 10 of file [column.class.cpp](#).

7.8.3.2 string column::column_name

Definition at line 9 of file [column.class.cpp](#).

7.8.3.3 string column::data_align = "center"

Definition at line 27 of file [column.class.cpp](#).

7.8.3.4 string column::drop_down_default

Definition at line 21 of file [column.class.cpp](#).

7.8.3.5 string column::drop_down_keyField

Definition at line 18 of file [column.class.cpp](#).

7.8.3.6 string column::drop_down_keyFieldQuote

Definition at line 19 of file [column.class.cpp](#).

7.8.3.7 string column::drop_down_showField

Definition at line 20 of file [column.class.cpp](#).

7.8.3.8 string column::drop_down_table

Definition at line 17 of file [column.class.cpp](#).

7.8.3.9 struct tm column::dValue [private]

Definition at line 43 of file [column.class.cpp](#).

7.8.3.10 float column::fValue [private]

Definition at line 42 of file [column.class.cpp](#).

7.8.3.11 string column::input_type

Definition at line 13 of file [column.class.cpp](#).

7.8.3.12 int column::iValue [private]

Definition at line 40 of file [column.class.cpp](#).

7.8.3.13 string column::label

Definition at line 22 of file [column.class.cpp](#).

7.8.3.14 string column::placeholder

Definition at line 23 of file [column.class.cpp](#).

7.8.3.15 string column::post_code

Definition at line 24 of file [column.class.cpp](#).

7.8.3.16 bool column::show_with_previous = false

Definition at line 25 of file [column.class.cpp](#).

7.8.3.17 string column::sValue [private]

Definition at line 41 of file [column.class.cpp](#).

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

- [column.class.cpp](#)

7.9 config Class Reference

Public Attributes

- char `layout` = '2'
Presentation.
- char `language` = '1'
1=UK English
- bool `show_aside` = false
whether or not to show the side-box
- char `font_combination` = '5'
1=Corben&Nobile, 2=Droid (Serif and Sans), d = default (non-google fonts), 3=Ubuntu+Sans, 4=Ubuntu+Vollkorn, 5=handwritten+sans, 6 = handwritten titles+serif! (pref 3 and 5)
- const bool `parse_wrapper` = false
true will turn of the rendering of the menu related to the soft_server_url
- const string `wrapper_name` = "eFinancialPlanner"
- const string `wrapper_url` = "http://eFinancialPlanner.eu"
- const string `soft_name` = "Goal Ranking 1"
- const string `soft_version` = "V0.4.0"
- std::vector< int > `total_portf_monthNbrs` = {1}
the month numbers used to show the total portfolio composition
- const bool `show_social` = false
- const bool `merge_ER_vol` = true
true to show ER and vol together in one table in the "Personalize" screen
- const bool `simul_show_table` = false
true if we show the table of E[R] and vol in the simulation-overview
- string `riskFunction` = "VaR"
Math.
- const string `db_host` = "tcp://127.0.0.1:3306"
Database.
- const string `db_name` = "efp"
- const string `db_user` = "efp"
- const string `db_pwd` = "efp.cger"
- const string `tbl_prefix` = "pfp3"
tables start with "" + oConfig.tbl_prefix + "_" if the prefix is eg. "tbl" then we have "tbl_invesotr3"
- const string `soft_file` = "pfp5.cgi"
Apache relevant parameters.
- const string `soft_server_url` = "http://localhost"
- const string `www_dir` = "/efp/"
- const string `img_dir` = "/efp/img/"
- const string `css_dir` = "/efp/style/"
- const string `soft_dir` = "/cgi-bin/"
this is the directory used by the webserver to locate the software (not the physical location)
- const string `scripts_dir` = "/efp/scripts/"
- string `soft_url` = this->`soft_server_url` + this->`soft_dir` + this->`soft_file`

7.9.1 Detailed Description

class config

copyright: (c) Philippe De Brouwer 2015

last modification:

Definition at line 10 of file [config.class.cpp](#).

7.9.2 Member Data Documentation

7.9.2.1 `const string config::css_dir = "/efp/style/"`

Definition at line 48 of file [config.class.cpp](#).

7.9.2.2 `const string config::db_host = "tcp://127.0.0.1:3306"`

Database.

Definition at line 33 of file [config.class.cpp](#).

7.9.2.3 `const string config::db_name = "efp"`

Definition at line 34 of file [config.class.cpp](#).

7.9.2.4 `const string config::db_pwd = "efp.cger"`

Definition at line 36 of file [config.class.cpp](#).

7.9.2.5 `const string config::db_user = "efp"`

Definition at line 35 of file [config.class.cpp](#).

7.9.2.6 `char config::font_combination = '5'`

1=Corben&Nobile, 2=Droid (Serif and Sans), d = default (non-google fonts), 3=Ubuntu+Sans, 4=Ubuntu+Vollkorn, 5=handwritten+sans, 6 = handwritten titles+serif! (pref 3 and 5)

Definition at line 18 of file [config.class.cpp](#).

7.9.2.7 `const string config::img_dir = "/efp/img/"`

Definition at line 47 of file [config.class.cpp](#).

7.9.2.8 `char config::language = '1'`

1=UK English

Definition at line 16 of file [config.class.cpp](#).

7.9.2.9 `char config::layout = '2'`

Presentation.

1=Traditional (base2.css) | 2=Responsive (bootstrap)

Definition at line 15 of file [config.class.cpp](#).

7.9.2.10 `const bool config::merge_ER_vol = true`

true to show ER and vol together in one table in the "Personalize" screen

Definition at line 26 of file [config.class.cpp](#).

7.9.2.11 `const bool config::parse_wrapper = false`

true will turn of the rendering of the menu related to the `soft_server_url`

Definition at line 19 of file [config.class.cpp](#).

7.9.2.12 `string config::riskFunction = "VaR"`

Math.

Definition at line 30 of file [config.class.cpp](#).

7.9.2.13 `const string config::scripts_dir = "/efp/scripts/"`

Definition at line 50 of file [config.class.cpp](#).

7.9.2.14 `bool config::show_aside = false`

whether or not to show the side-box

Definition at line 17 of file [config.class.cpp](#).

7.9.2.15 `const bool config::show_social = false`

Definition at line 25 of file [config.class.cpp](#).

7.9.2.16 `const bool config::simul_show_table = false`

true if we show the table of $E[R]$ and vol in the simulation-overview

Definition at line 27 of file [config.class.cpp](#).

7.9.2.17 `const string config::soft_dir = "/cgi-bin/"`

this is the directory used by the webserver to locate the software (not the physical location)

Definition at line 49 of file [config.class.cpp](#).

7.9.2.18 `const string config::soft_file = "pfp5.cgi"`

Apache relevant parameters.

Definition at line 41 of file [config.class.cpp](#).

7.9.2.19 `const string config::soft_name = "Goal Ranking 1"`

Definition at line 22 of file [config.class.cpp](#).

7.9.2.20 `const string config::soft_server_url = "http://localhost"`

Definition at line 45 of file [config.class.cpp](#).

7.9.2.21 `string config::soft_url = this->soft_server_url + this->soft_dir + this->soft_file`

Definition at line 59 of file [config.class.cpp](#).

7.9.2.22 `const string config::soft_version = "V0.4.0"`

Definition at line 23 of file [config.class.cpp](#).

7.9.2.23 `const string config::tbl_prefix = "pfp3"`

tables start with "" + oConfig.tbl_prefix + "_" if the prefix is eg. "tbl" then we have "tbl_invesotrs"

Definition at line 37 of file [config.class.cpp](#).

7.9.2.24 `std::vector<int> config::total_portf_monthNbrs = {1}`

the month numbers used to show the total portfolio composition

Definition at line 24 of file [config.class.cpp](#).

7.9.2.25 `const string config::wrapper_name = "eFinancialPlanner"`

Definition at line 20 of file [config.class.cpp](#).

7.9.2.26 `const string config::wrapper_url = "http://eFinancialPlanner.eu"`

Definition at line 21 of file [config.class.cpp](#).

7.9.2.27 `const string config::www_dir = "/efp/"`

Definition at line 46 of file [config.class.cpp](#).

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

- [config.class.cpp](#)

7.10 db_helper Class Reference

Public Member Functions

- [db_helper](#) ()
- [~db_helper](#) ()
- bool [runSQL](#) (string s)
- float [getFloat](#) (string s)
- int [getInt](#) (string s)

Public Attributes

- `sql::Driver` * [driver](#)
- `sql::Connection` * [con](#)
- `sql::Statement` * [stmt](#)
- `sql::ResultSet` * [res](#)

- `sql::ResultSet * res2`
- `sql::PreparedStatement * pstmt`
- `string db = "efp"`

7.10.1 Detailed Description

Definition at line 11 of file [db_helper.class.cpp](#).

7.10.2 Constructor & Destructor Documentation

7.10.2.1 `db_helper::db_helper ()`

Definition at line 40 of file [db_helper.class.cpp](#).

7.10.2.2 `db_helper::~db_helper ()`

Definition at line 91 of file [db_helper.class.cpp](#).

7.10.3 Member Function Documentation

7.10.3.1 `float db_helper::getFloat (string s) [inline]`

getString returning a float

Definition at line 142 of file [db_helper.class.cpp](#).

7.10.3.2 `int db_helper::getInt (string s) [inline]`

getString returning an int

Definition at line 150 of file [db_helper.class.cpp](#).

7.10.3.3 `bool db_helper::runSQL (string s)`

Definition at line 102 of file [db_helper.class.cpp](#).

7.10.4 Member Data Documentation

7.10.4.1 `sql::Connection* db_helper::con`

Definition at line 16 of file [db_helper.class.cpp](#).

7.10.4.2 `string db_helper::db = "efp"`

Definition at line 25 of file [db_helper.class.cpp](#).

7.10.4.3 `sql::Driver* db_helper::driver`

Definition at line 15 of file [db_helper.class.cpp](#).

7.10.4.4 `sql::PreparedStatement*` `db_helper::pstmt`

Definition at line 20 of file [db_helper.class.cpp](#).

7.10.4.5 `sql::ResultSet*` `db_helper::res`

Definition at line 18 of file [db_helper.class.cpp](#).

7.10.4.6 `sql::ResultSet*` `db_helper::res2`

Definition at line 19 of file [db_helper.class.cpp](#).

7.10.4.7 `sql::Statement*` `db_helper::stmt`

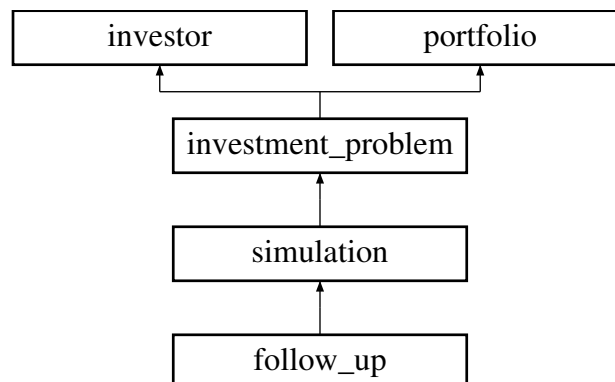
Definition at line 17 of file [db_helper.class.cpp](#).

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

- [db_helper.class.cpp](#)

7.11 follow_up Class Reference

Inheritance diagram for follow_up:



Public Member Functions

- [follow_up](#) (int investor)
- void [parse_dashboard](#) ()
- void [show_simulation](#) ()
- void [simulate](#) ()
- int [get_nbrMonths2simulate](#) ()
returns the number of months to simulate as in "37"
- void [plot_market_evol](#) ()
- void [solve](#) ()
allocates means (with a benchmark) to goals
- int [get_nbr_goals](#) ()
- int [get_max_month_for_lower_goals](#) (int g)
- void [javaGraph](#) (int g, string xtra_var="", string xtra_label="", int followup_mnth=0)
- void [prepare_javaVars_colors](#) (int g, int followup_mnth)

- void `parse_javaGraph` (int g, string xtra_var="", string xtra_label="")
plots the chartOverview
- string `get_full_name` ()
- float `age` (int Mnbr=0)
- int `months2simulate` ()
- int `load_from_db` (string investorID)
- void `load_ER` (int person=0, char person_type= 'i')
- void `save_ER` (int person=0, char person_type= 'i')
- void `load_covar` (int person, char person_type= 'i')
- void `save_covar` (int person, char person_type= 'i')
- bool `get_portf_from_db` (int id)
- string `get_description` ()
- void `set_mu` (float(*AssetClass_mu))
- void `set_sigma` (float(*AssetClass_varCov))

Public Attributes

- std::map< int, aGOAL > `goalZ`
map containing all goals
- int `investor_id`
- string `user_name`
- string `first_name`
- string `last_name`
- string `password`
- string `currency`
the default currency for that customer
- struct tm `birth_date`
- float `simulate_till_age`
- float * `assetClass_covar` = new float[(NBR_ASSET_CLASSES + 1) * (NBR_ASSET_CLASSES + 1)]
- float `assetClass_mu` [(NBR_ASSET_CLASSES+1)]
note: index 0 not used
- std::map< int, string > `assetClass_name`
note: index 0 not used
- std::map< int, float > `ER`
- std::map< int, std::map< int, float > > `covar`
- int `portfolio_id`
- string `description`
- float `weights` [NBR_ASSET_CLASSES]
the weights of the asset classes (ordered as the asset classes)
- float `pMu`
the MONTHLY return
- float `pLogR`
the MONTHLY log-return
- float `pSigma`
the MONTHLY volatility

Protected Member Functions

- void [set_followup_color_prae](#) (float Vlow, float Vmed, float Vhigh, int g)
- void [set_followup_color_post](#) (float V, int g)
- int [age2monthNbr](#) (float theAge)
- string [dateStr2Age](#) (string the_date)
- bool [save](#) ()
- int [get_from_db](#) (string investorID, string password)
- bool [add_to_db](#) ()
- bool [exists](#) (int iid)
- void [load_preferences](#) ()
 - sets the experience, knowledge and desirability vectors*
- bool [save_preferences](#) ()
 - sets the experience, knowledge and desirability vectors*
- void [set_max_exposure](#) ()
 - sets the max_exposure vector based on the experience, knowledge and desirability*
- bool [set_scale](#) (string scale_type, int ac, int val)
- void [reload_ER](#) (int person, char person_type= 'i')
 - TODO: the ER and assetClass_mu as well as the covar and assetClass_covar are redundant ==> eliminate the C-style assetClass_*.*
- void [reload_covar](#) (int person, char what2do, char person_type= 'i')

Protected Attributes

- int [nbrMonths](#)
 - the number of months to simulate, per goal we simulate max(realization_monthNbr, nbrMonths)*
- std::map< int, std::map< int, float > > [market_return](#)
- std::map< int, [portfolio](#) > [portfolios](#)
 - the standard portfolios*
- std::map< int, int > [experience](#)
- std::map< int, int > [knowledge](#)
- std::map< int, int > [desirability](#)
- std::map< int, float > [max_exposure](#)
 - the maximum exposure per asset class*

7.11.1 Detailed Description

[follow_up](#)

class used to simulate behaviour of the investment problem after optimization

Definition at line 7 of file [follow_up.class.cpp](#).

7.11.2 Constructor & Destructor Documentation

7.11.2.1 [follow_up::follow_up](#) (int *investor*)

CONSTRUCTOR of the class simulation

Definition at line 17 of file [follow_up.class.cpp](#).

7.11.3 Member Function Documentation

7.11.3.1 `bool investor::add_to_db ()` [protected],[inherited]

`add_to_db`

does not check data any more, simply tries to push it to the DB only returns 0 if the userID already exists.

Definition at line 313 of file [investor.class.cpp](#).

7.11.3.2 `float investor::age (int Mnbr = 0)` [inline],[inherited]

returns the age of a customer in years (with decimal part)

Definition at line 108 of file [investor.class.cpp](#).

7.11.3.3 `int investor::age2monthNbr (float theAge)` [inline],[protected],[inherited]

`age2monthNbr`

returns the month number for a given age note that this is negative for ages younger than the actual age.

Definition at line 200 of file [investor.class.cpp](#).

7.11.3.4 `string investor::dateStr2Age (string the_date)` [inline],[protected],[inherited]

`dateStr2Age`

Definition at line 209 of file [investor.class.cpp](#).

7.11.3.5 `bool investor::exists (int iid)` [protected],[inherited]

`exists`

checks if a given investor-id really exists

Definition at line 343 of file [investor.class.cpp](#).

7.11.3.6 `string portfolio::get_description ()` [inherited]

returns the label of the portfolio

Definition at line 86 of file [portfolio.class.cpp](#).

7.11.3.7 `int investor::get_from_db (string uid, string password)` [protected],[inherited]

`get_from_db` returns the investor_id if successful (otherwise 0)

Definition at line 249 of file [investor.class.cpp](#).

7.11.3.8 `string investor::get_full_name ()` [inherited]

overloading << operator overload the << operator to output the goal in a readable format `get_full_name` returns the full name of an investor

Definition at line 103 of file [investor.class.cpp](#).

7.11.3.9 `int investment_problem::get_max_month_for_lower_goals (int g)` [inherited]

returns the maximum term of all the other LOWER ranking goals. The higher ranking goals already have allocated all the means that they require, so we can disregard them here. `goal_type = 0` is the unallocated goal type, so this goal should not be taken into account here.

Definition at line 945 of file [investment_problem.class.cpp](#).

7.11.3.10 `int investment_problem::get_nbr_goals ()` [inherited]

returns the number of goals

Definition at line 936 of file [investment_problem.class.cpp](#).

7.11.3.11 `int simulation::get_nbrMonths2simulate ()` [inherited]

returns the number of months to simulate as in "37"

[get_nbrMonths2simulate\(\)](#)

returns nbrMonths (to simulate)

Definition at line 259 of file [simulation.class.cpp](#).

7.11.3.12 `bool portfolio::get_portf_from_db (int id)` [inherited]

load the definitions from the database note: we have to create a second instance of the sql connection as this is called while the object db is in use!

Definition at line 32 of file [portfolio.class.cpp](#).

7.11.3.13 `void investment_problem::javaGraph (int g, string xtra_var = " ", string xtra_label = " ", int followup_mnth = 0)` [inherited]

`javaGraph`

plots the overview-graph for the given goal

`xtav_Var` and `xtra_label` add a series of data that is delivered in text format `xtra_var = [45,2000], [45.5,2200]` `xtra_label = "history"` `followup_mnth = 0` for feedback and simulation screen, and for the follow up screen it is the number of months we look into the future.

Definition at line 1089 of file [investment_problem.class.cpp](#).

7.11.3.14 `void market::load_covar (int person, char person_type = 'i')` [inherited]

`load_covar` < note: `person_type = 'i'` for persons (later to provide for 'a' advisor, organization, etc.)

< TODO eliminate this

if we did not find a personalized expectation for each asset class, then we load the default values

< TODO eliminate this

Definition at line 135 of file [market.class.cpp](#).

7.11.3.15 `void market::load_ER (int person = 0, char person_type = 'i')` [inherited]

`load_ER` < note: `person_type = 'i'` for persons (later to provide for 'a' advisor, organization, etc.)

Definition at line 61 of file [market.class.cpp](#).

7.11.3.16 `int investor::load_from_db (string uid)` [inherited]

returns the investor_id if successful (otherwise 0) NOTE for user interaction use get_from_db(uid, password) for password verification!!

Definition at line 134 of file [investor.class.cpp](#).

7.11.3.17 `void investor::load_preferences ()` [protected],[inherited]

sets the experience, knowledge and desirability vectors

set_preferences

Definition at line 383 of file [investor.class.cpp](#).

7.11.3.18 `int investor::months2simulate ()` [inline],[inherited]

months2simulate

Definition at line 124 of file [investor.class.cpp](#).

7.11.3.19 `void follow_up::parse_dashboard ()`

parse_dashboard

Definition at line 26 of file [follow_up.class.cpp](#).

7.11.3.20 `void investment_problem::parse_javaGraph (int g, string xtra_var = "", string xtra_label = "")`
[inherited]

plots the chartOverview

parse_javaGraph

parses the line plot of the three scenarios and the goal to the output (webbrowser). It parses as well the that will get become the plot as the <script> with javascript that will produce the plot.

Definition at line 1103 of file [investment_problem.class.cpp](#).

7.11.3.21 `void simulation::plot_market_evol ()` [inherited]

plot_market_evol

plots the evolution of the different asset classes (is goal independent) initialize:

show the table if desired

build the simulation variables:

cout the script

Definition at line 138 of file [simulation.class.cpp](#).

7.11.3.22 `void investment_problem::prepare_javaVars_colors (int g, int followup_mnth)` [inherited]

prepare_javaVars_colors

followup_mnth is used as the startmonth for the followUp-Screen (in all other graphs it will be zero – its default)

exp = expected value med = median low = alpha quantile high = 1- alpha quantile < the weighted average of the number of months that investments are in portfolio

TODO in the function below use another rate for NEGATIVE amounts, to mymic lending!

```
parse_javaGraph(g, s1, s2, s3, s4, xtra_var, xtra_label);
```

< place al the variables

Definition at line 970 of file [investment_problem.class.cpp](#).

7.11.3.23 `void market::reload_covar (int person, char what2do, char person_type = ' i ')` [protected],
[inherited]

reload_vol

the situation here is more complex as the user expect that only the volatilities will be reset to the default values and not the correlations ...

what2do = [v|c|b] as in [vol | corr | both] < both

< volatilities

< correlations

Definition at line 281 of file [market.class.cpp](#).

7.11.3.24 `void market::reload_ER (int person, char person_type = ' i ')` [protected],[inherited]

TODO: the ER and assetClass_mu as well as the covar and assetClass_covar are redundant ==> eliminate the C-style assetClass_*.

set_mu

the expect value for all asset classes set_covar

the expect value for all assets reload_ER

note: person_type = 'i' for persons (later to provide for 'a' advisor, organization, etc.)

this function simply deletes the personalized expected returns. This is sufficient because later we check if they exists and only then load them (see function [load_ER\(\)](#)).

Definition at line 247 of file [market.class.cpp](#).

7.11.3.25 `bool investor::save ()` [protected],[inherited]

save

Definition at line 220 of file [investor.class.cpp](#).

7.11.3.26 `void market::save_covar (int person, char person_type = ' i ')` [inherited]

save_covar

Definition at line 172 of file [market.class.cpp](#).

7.11.3.27 `void market::save_ER (int person = 0, char person_type = ' i ')` [inherited]

save_ER

Definition at line 109 of file [market.class.cpp](#).

7.11.3.28 `bool investor::save_preferences ()` [protected],[inherited]

sets the experience, knowledge and desirability vectors

[save_preferences\(\)](#)

Definition at line 416 of file [investor.class.cpp](#).

7.11.3.29 `void investment_problem::set_followup_color_post (float V, int g) [inline], [protected], [inherited]`

`set_followup_color_post` sets the color of the goal in case the simulation is LONGER than the realization date of the goal

Definition at line 1246 of file [investment_problem.class.cpp](#).

7.11.3.30 `void investment_problem::set_followup_color_prae (float Vlow, float Vmed, float Vhigh, int g) [inline], [protected], [inherited]`

`set_followup_color_prae`

sets the color of the goal in case the simulations ends PRIOR to the realization date of the goal

Definition at line 1234 of file [investment_problem.class.cpp](#).

7.11.3.31 `void investor::set_max_exposure () [protected], [inherited]`

sets the `max_exposure` vector based on the experience, knowledge and desirability

`get_max_exposure`

sets the map `max_exposure` in this object

Definition at line 363 of file [investor.class.cpp](#).

7.11.3.32 `void portfolio::set_mu (float * AssetClass_mu) [inherited]`

sets the expected return for the portfolio

Definition at line 94 of file [portfolio.class.cpp](#).

7.11.3.33 `bool investor::set_scale (string scale_type, int ac, int val) [protected], [inherited]`

`set_scale`

Definition at line 402 of file [investor.class.cpp](#).

7.11.3.34 `void portfolio::set_sigma (float * AssetClass_varCov) [inherited]`

sets the standard deviation for the portfolio: `this->sigma`

Definition at line 112 of file [portfolio.class.cpp](#).

7.11.3.35 `void simulation::show_simulation () [inherited]`

7.11.3.36 `void simulation::simulate () [inherited]`

`simulate`

simulates one possible scenario for what can happen after the goals are set TODO in absense of `gGaoI.load()` we solve again ...!!!!!!!

< does this for each goal and sets `goalZ[g].simulation_string` (requires [solve\(\)](#))

Definition at line 49 of file [simulation.class.cpp](#).

7.11.3.37 void investment_problem::solve () [inherited]

allocates means (with a benchmark) to goals

solve is the main attribute to be used by other classes a [cls_currency](#) object that will facilitate currency conversions

note: set_assets, set_cfs, set_goals, set_mu are done in the constructor

< initialize means_left for g=1 (all in [investor.currency](#))

< initialize means_left for g=1 (all in [investor.currency](#))

< do nothing right now for unallocated goals, so take the next goal (for loop)

try the free means (not used by other –lower ranking– goals)

< initialize

and means that are allocated to this goal):

for the asap-goals we take the first cash flows first

for all other goal types we take the last possible

if the free means are not sufficient, try all the previous also

block the resources for this goal

set the benchmark

set comments and colors:

add the risk info

put all the remaining means into the unallocated goal

Definition at line 396 of file [investment_problem.class.cpp](#).

7.11.4 Member Data Documentation

7.11.4.1 float* market::assetClass_covar = new float[(NBR_ASSET_CLASSES + 1) * (NBR_ASSET_CLASSES + 1)]
[inherited]

Definition at line 16 of file [market.class.cpp](#).

7.11.4.2 float market::assetClass_mu[(NBR_ASSET_CLASSES+1)] [inherited]

note: index 0 not used

Definition at line 17 of file [market.class.cpp](#).

7.11.4.3 std::map<int, string> market::assetClass_name [inherited]

note: index 0 not used

Definition at line 18 of file [market.class.cpp](#).

7.11.4.4 struct tm investor::birth_date [inherited]

Definition at line 20 of file [investor.class.cpp](#).

7.11.4.5 `std::map<int, std::map<int, float>>` `market::covar` [inherited]

Definition at line 26 of file [market.class.cpp](#).

7.11.4.6 `string` `investor::currency` [inherited]

the default currency for that customer

Definition at line 19 of file [investor.class.cpp](#).

7.11.4.7 `string` `portfolio::description` [inherited]

Definition at line 15 of file [portfolio.class.cpp](#).

7.11.4.8 `std::map<int, int>` `investor::desirability` [protected],[inherited]

Definition at line 41 of file [investor.class.cpp](#).

7.11.4.9 `std::map<int, float>` `market::ER` [inherited]

Definition at line 25 of file [market.class.cpp](#).

7.11.4.10 `std::map<int, int>` `investor::experience` [protected],[inherited]

Definition at line 39 of file [investor.class.cpp](#).

7.11.4.11 `string` `investor::first_name` [inherited]

Definition at line 16 of file [investor.class.cpp](#).

7.11.4.12 `std::map<int, aGOAL>` `investment_problem::goalZ` [inherited]

map containing all goals

Definition at line 22 of file [investment_problem.class.cpp](#).

7.11.4.13 `int` `investor::investor_id` [inherited]

Definition at line 14 of file [investor.class.cpp](#).

7.11.4.14 `std::map<int, int>` `investor::knowledge` [protected],[inherited]

Definition at line 40 of file [investor.class.cpp](#).

7.11.4.15 `string` `investor::last_name` [inherited]

Definition at line 17 of file [investor.class.cpp](#).

7.11.4.16 `std::map<int, std::map<int, float>>` `simulation::market_return` [protected],[inherited]

Definition at line 25 of file [simulation.class.cpp](#).

7.11.4.17 `std::map<int, float> investor::max_exposure` [protected],[inherited]

the maximum exposure per asset class

Definition at line 42 of file [investor.class.cpp](#).

7.11.4.18 `int simulation::nbrMonths` [protected],[inherited]

the number of months to simulate, per goal we simulate $\max(\text{realization_monthNbr}, \text{nbrMonths})$

Definition at line 22 of file [simulation.class.cpp](#).

7.11.4.19 `string investor::password` [inherited]

Definition at line 18 of file [investor.class.cpp](#).

7.11.4.20 `float portfolio::pLogR` [inherited]

the MONTHLY log-return

Definition at line 25 of file [portfolio.class.cpp](#).

7.11.4.21 `float portfolio::pMu` [inherited]

the MONTHLY return

Definition at line 24 of file [portfolio.class.cpp](#).

7.11.4.22 `int portfolio::portfolio_id` [inherited]

Definition at line 14 of file [portfolio.class.cpp](#).

7.11.4.23 `std::map<int, portfolio> investment_problem::portfolios` [protected],[inherited]

the standard portfolios

Definition at line 29 of file [investment_problem.class.cpp](#).

7.11.4.24 `float portfolio::pSigma` [inherited]

the MONTHLY volatility

Definition at line 26 of file [portfolio.class.cpp](#).

7.11.4.25 `float investor::simulate_till_age` [inherited]

Definition at line 21 of file [investor.class.cpp](#).

7.11.4.26 `string investor::user_name` [inherited]

Definition at line 15 of file [investor.class.cpp](#).

7.11.4.27 float portfolio::weights[NBR_ASSET_CLASSES] [inherited]

the weights of the asset classes (ordered as the asset classes)

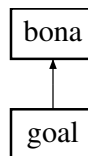
Definition at line 17 of file [portfolio.class.cpp](#).

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

- [follow_up.class.cpp](#)

7.12 goal Class Reference

Inheritance diagram for goal:



Public Member Functions

- [goal](#) ()
- void [list](#) (int iid, bool editable=true)
- bool [save](#) ()
- void [show](#) (bool editable=true)
- void [show_edit_form](#) (int iid, bool showExisting=false)
- bool [delete_bona](#) ()
- bool [exists](#) ()
- bool [load_fromEnv](#) ()
- bool [load_fromDB](#) ()

Public Attributes

- std::map< int, [column](#) > [tbl](#)
- string [tbl_name](#) = oConfig.tbl_prefix + "_assets"
- string [class_name](#) = ""
- string [action_ref](#)
- std::map< string, int > [defList](#)

list holding the column names in the table connected to the correct id of the column

Protected Attributes

- string [order_by](#) = ""
should be the column name eventually followed by DESC/ASC, eg. "priority ASC"

7.12.1 Detailed Description

Definition at line 11 of file [bona_goal.class.cpp](#).

7.12.2 Constructor & Destructor Documentation

7.12.2.1 goal::goal ()

< note: this must be the exact column name in the database, when no ordering is required: leave empty ""

Definition at line 22 of file [bona_goal.class.cpp](#).

7.12.3 Member Function Documentation

7.12.3.1 bool bona::delete_bona () [inherited]

Definition at line 350 of file [bona.class.cpp](#).

7.12.3.2 bool bona::exists () [inherited]

Definition at line 382 of file [bona.class.cpp](#).

7.12.3.3 void bona::list (int iid, bool *editable* = true) [inherited]

void list(int investor_id);

to trigger the scripts to fill out the dates and provide the date-picker

so javascripts can check if this exists to know if they are in the cash flow screen

Definition at line 157 of file [bona.class.cpp](#).

7.12.3.4 bool bona::load_fromDB () [inherited]

Definition at line 44 of file [bona.class.cpp](#).

7.12.3.5 bool bona::load_fromEnv () [inherited]

Definition at line 336 of file [bona.class.cpp](#).

7.12.3.6 bool bona::save () [inherited]

Definition at line 276 of file [bona.class.cpp](#).

7.12.3.7 void bona::show (bool *editable* = true) [inherited]

Definition at line 234 of file [bona.class.cpp](#).

7.12.3.8 void bona::show_edit_form (int iid, bool *showExisting* = false) [inherited]

to trigger the scripts to fill out the dates and provide the date-picker

Definition at line 108 of file [bona.class.cpp](#).

7.12.4 Member Data Documentation

7.12.4.1 `string bona::action_ref` [inherited]

Definition at line 17 of file [bona.class.cpp](#).

7.12.4.2 `string bona::class_name = ""` [inherited]

Definition at line 16 of file [bona.class.cpp](#).

7.12.4.3 `std::map<string, int> bona::defList` [inherited]

list holding the column names in the table connected to the correct id of the column

Definition at line 26 of file [bona.class.cpp](#).

7.12.4.4 `string bona::order_by = ""` [protected],[inherited]

should be the column name eventually followed by DESC/ASC, eg. "priority ASC"

Definition at line 28 of file [bona.class.cpp](#).

7.12.4.5 `std::map<int, column> bona::tbl` [inherited]

Definition at line 14 of file [bona.class.cpp](#).

7.12.4.6 `string bona::tbl_name = oConfig.tbl_prefix + "_assets"` [inherited]

Definition at line 15 of file [bona.class.cpp](#).

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

- [bona_goal.class.cpp](#)

7.13 `html_helper` Class Reference

Public Member Functions

- [html_helper](#) ()
- void [footer](#) (string tab_name, string extraInfo="")
- void [show_home](#) ()
- void [header](#) (int iid=0, bool addSavingsPlan=false, bool addPlots=false)
- string [get_string](#) (const std::string &varName)
- void [show_toolBar](#) (string tab_name, string usrName="")
- void [next_step](#) (int stepNbr, int subStepNbr)
- void [aside](#) (const string &textLine)
- void [cout_box_color](#) (string the_color)
- string [get_box_color_class](#) (string the_color)
- string [get_glyphicon](#) (string the_color)
- string [terms_of_use](#) ()
- string [privacy_policy](#) ()
- string [cookie_policy](#) ()

- string `collapse_open` (string parent_id, string title, int nbr, bool add_triangle=true, bool is_open=true, string xtra_body_class="")
- string `collapse_close` (string parent_id="", int nbr=0, bool add_triangle=true)

Public Attributes

- const string `c_salt` = "k35f9vfgDtjTe1"
- const string `seriesColorsGoalPlot` =["'rgba(160, 0, 0, 0.6)', 'rgba(0, 0, 160, 0.6)', 'rgba(0, 160, 0, 0.6)', '#000', '#663', '#958c12', '#953579', '#4b5de4', '#d8b83f', '#ff5800', '#0085cc'"]
- const string `assetClassColors` =["'#999', '#0c3', '#396', '#0c9', '#3cf', '#69f', '#96f', '#f93', '#fc3', '#f63', '#33f'"]

Private Types

- typedef std::map< string, string > `STRMAP`
- typedef std::map< int, `STRMAP` > `STEPDEF`

Private Member Functions

- void `header_menu_bar` ()
- void `parse_date_time` ()
- void `show_args` (int argc, char **argv)
- void `parse_breadcrumb` (int tab, int sub_tab)
- void `parse_title` (int tab, int sub_tab)
- int `get_prev_step` (int theStep, int theSubStep)
- int `get_prev_sub_step` (int theStep, int theSubStep)
- int `get_next_step` (int theStep, int theSubStep)
- int `get_next_sub_step` (int theStep, int theSubStep)
- bool `sub_step_exists` (int theStep, int theSubStep)
- int `get_step_nbr` (string step_name)
- int `get_sub_step_nbr` (string sub_step_name)
- void `show_social` ()
- void `navbar_start` ()
- void `navbar_item` (int k, string usr_name="", string tab_name="")
- void `navbar_end` ()
- void `navbar_box` (int k, string usr_name="")

Private Attributes

- const std::map< int, string > `c_photos_phdb`
- const string `c_photos_phdb_dir` = "/img/"
- const string `c_content_header` = "Content-type:text/html\r\n\r\n"
- const string `c_doc_type` = "<!DOCTYPE html>"
- std::map< int, string > `scale125i2s`
a scale from 1 to 5, int to string
- `STEPDEF` `steps`
- std::map< int, `STEPDEF` > `sub_steps`
- std::map< string, int > `step_nbrs`
- std::map< string, int > `sub_step_nbrs`
- const string `c_right_menu_names` [`NBR_RIGHT_MENU`] = {"My Account", "Disclaimers", "Log Out"}
- const string `c_right_menu_urls` [`NBR_RIGHT_MENU`]
- const string `c_arrow` = " ";"

7.13.1 Detailed Description

Definition at line 1 of file [html_helper.class.cpp](#).

7.13.2 Member Typedef Documentation

7.13.2.1 `typedef std::map<int, STRMAP> html_helper::STEPDEF [private]`

Definition at line 47 of file [html_helper.class.cpp](#).

7.13.2.2 `typedef std::map<string, string> html_helper::STRMAP [private]`

Definition at line 46 of file [html_helper.class.cpp](#).

7.13.3 Constructor & Destructor Documentation

7.13.3.1 `html_helper::html_helper ()`

CONSTRUCTOR ==> initializations note that these definitions cannot be done at declaration (must be in a function!)

HOME

login & registert

Personalize

Goals

Assets & cash flows

feedback

simulation

followup

Definition at line 125 of file [html_helper.class.cpp](#).

7.13.4 Member Function Documentation

7.13.4.1 `void html_helper::aside (const string & textLine)`

aside

Definition at line 1119 of file [html_helper.class.cpp](#).

7.13.4.2 `string html_helper::collapse_close (string parent_id = "", int nbr = 0, bool add_triangle = true) [inline]`

collapse_close

Definition at line 401 of file [html_helper.class.cpp](#).

7.13.4.3 `string html_helper::collapse_open (string parent_id, string title, int nbr, bool add_triangle = true, bool is_open = true, string xtra_body_class = "") [inline]`

collapse_open < note the "in": the jqplot chart won't render in hidden div's :-)

Definition at line 375 of file [html_helper.class.cpp](#).

7.13.4.4 `string html_helper::cookie_policy ()` [inline]

terms_of_use

Definition at line 1312 of file [html_helper.class.cpp](#).

7.13.4.5 `void html_helper::cout_box_color (string the_color)`

cout_box_color

sends the appropriate class (for the div surrounding a goal)

Definition at line 1233 of file [html_helper.class.cpp](#).

7.13.4.6 `void html_helper::footer (string tab_name, string extraInfo = " ")`

footer

Definition at line 428 of file [html_helper.class.cpp](#).

7.13.4.7 `string html_helper::get_box_color_class (string the_color)`

get_box_color_class

Definition at line 1246 of file [html_helper.class.cpp](#).

7.13.4.8 `string html_helper::get_glyphicon (string the_color)`

get_glyphicon

Definition at line 1261 of file [html_helper.class.cpp](#).

7.13.4.9 `int html_helper::get_next_step (int theStep, int theSubStep)` [inline],[private]

get_next_step

Definition at line 230 of file [html_helper.class.cpp](#).

7.13.4.10 `int html_helper::get_next_sub_step (int theStep, int theSubStep)` [inline],[private]

get_next_sub_step

Definition at line 250 of file [html_helper.class.cpp](#).

7.13.4.11 `int html_helper::get_prev_step (int theStep, int theSubStep)` [inline],[private]

get_prev_step

Definition at line 215 of file [html_helper.class.cpp](#).

7.13.4.12 `int html_helper::get_prev_sub_step (int theStep, int theSubStep)` [inline],[private]

get_prev_sub_step

xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxNOT OK!!!!!!!!!!!! TODO: repair this function, it does not work for the step after one with sub-steps

Definition at line 284 of file [html_helper.class.cpp](#).

7.13.4.13 `int html_helper::get_step_nbr (string step_name) [private]`

`get_step_nbr`

Definition at line 334 of file [html_helper.class.cpp](#).

7.13.4.14 `string html_helper::get_string (const std::string & varName)`

Definition at line 785 of file [html_helper.class.cpp](#).

7.13.4.15 `int html_helper::get_sub_step_nbr (string sub_step_name) [private]`

`get_sub_step_nbr`

Definition at line 354 of file [html_helper.class.cpp](#).

7.13.4.16 `void html_helper::header (int iid = 0, bool addSavingsPlan = false, bool addPlots = false)`

`header`

`addPlots` -> adds the jqPlot sources to display graphs `addSavingsPlan` -> adds the show/hide buttons for the savings plan === jquery

=== jqplot

=== bespoke javascript

=== date-picker

=== the presentation (bootstrap)

=== fonts

=== body

< if it is not in an iframe, then the menu is parsed, otherwise we assume that it is done by the calling webpage

Definition at line 562 of file [html_helper.class.cpp](#).

7.13.4.17 `void html_helper::header_menu_bar () [private]`

`header_menu_bar`

Definition at line 539 of file [html_helper.class.cpp](#).

7.13.4.18 `void html_helper::navbar_box (int k, string usr_name = "") [inline],[private]`

`navbar_box`

parses the box in that contains the username, links to `my_account`, Disclaimers, etc.

Definition at line 948 of file [html_helper.class.cpp](#).

7.13.4.19 `void html_helper::navbar_end () [inline],[private]`

`navbar_end`

ends the navbar environment

Definition at line 929 of file [html_helper.class.cpp](#).

7.13.4.20 `void html_helper::navbar_item (int k, string usr_name = "", string tab_name = "") [inline], [private]`

`navbar_item`

adds one item to the navigation bar

Definition at line 821 of file [html_helper.class.cpp](#).

7.13.4.21 `void html_helper::navbar_start () [inline],[private]`

`navbar_start`

open the environment in which the navigation bar will reside

Definition at line 802 of file [html_helper.class.cpp](#).

7.13.4.22 `void html_helper::next_step (int stepNbr, int subStepNbr)`

`next_step`

Definition at line 1046 of file [html_helper.class.cpp](#).

7.13.4.23 `void html_helper::parse_breadcrumb (int tab, int sub_tab) [private]`

`parse_breadcrumb`

parse the breadcrumb given the location as tab, subtab < if there is a sub-tab, then show it

< if there is a sub-tab, then show it

Definition at line 1158 of file [html_helper.class.cpp](#).

7.13.4.24 `void html_helper::parse_date_time () [private]`

`parse_date_time`

Definition at line 529 of file [html_helper.class.cpp](#).

7.13.4.25 `void html_helper::parse_title (int tab, int sub_tab) [private]`

`parse_title`

parses the title section to cout, given the location as tab, subtab

Definition at line 1198 of file [html_helper.class.cpp](#).

7.13.4.26 `string html_helper::privacy_policy () [inline]`

`privacy_policy`

Definition at line 1303 of file [html_helper.class.cpp](#).

7.13.4.27 `void html_helper::show_args (int argc, char ** argv) [private]`

7.13.4.28 `void html_helper::show_home ()`

`show_home`

the landing page of the software

Definition at line 722 of file [html_helper.class.cpp](#).

7.13.4.29 void `html_helper::show_social ()` [`private`]

`show_social` shows the badges for the social media (with underlying links to follow, etc.)

Definition at line 1138 of file [html_helper.class.cpp](#).

7.13.4.30 void `html_helper::show_toolBar (string tab_name, string usrName = " ")`

`show_toolBar`

Definition at line 988 of file [html_helper.class.cpp](#).

7.13.4.31 bool `html_helper::sub_step_exists (int theStep, int theSubStep)` [`inline`], [`private`]

`sub_step_exists`

Definition at line 308 of file [html_helper.class.cpp](#).

7.13.4.32 string `html_helper::terms_of_use ()` [`inline`]

`terms_of_use`

Definition at line 1294 of file [html_helper.class.cpp](#).

7.13.5 Member Data Documentation

7.13.5.1 const string `html_helper::assetClassColors = "[#999', '#0c3', '#396', '#0c9', '#3cf', '#69f', '#96f', '#f93', '#fc3', '#f63', '#33f']"`

Definition at line 23 of file [html_helper.class.cpp](#).

7.13.5.2 const string `html_helper::c_arrow = " "` [`private`]

Definition at line 112 of file [html_helper.class.cpp](#).

7.13.5.3 const string `html_helper::c_content_header = "Content-type:text/html\r\n\r\n"` [`private`]

Definition at line 42 of file [html_helper.class.cpp](#).

7.13.5.4 const string `html_helper::c_doc_type = "<!DOCTYPE html>"` [`private`]

Definition at line 43 of file [html_helper.class.cpp](#).

7.13.5.5 const std::map<int, string> `html_helper::c_photos_phdb` [`private`]

Initial value:

```
= {{1, "phdb/ZM5K1325_01.jpg"}, {2, "phdb/ZM5K1312_01b.jpg"},
  {3, "phdb/ZM5K1302_01c.jpg"}, {4, "phdb/ZM5K1313_01c.jpg"}, {5, "fvdspieg1.jpg"}, {6, "fvdspieg2.jpg"}}
```

Definition at line 39 of file [html_helper.class.cpp](#).

7.13.5.6 `const string html_helper::c_photos_phdb_dir = "/img/"` `[private]`

Definition at line 41 of file [html_helper.class.cpp](#).

7.13.5.7 `const string html_helper::c_right_menu_names[NBR_RIGHT_MENU] = {"My Account", "Disclaimers", "Log Out"}`
`[private]`

Definition at line 106 of file [html_helper.class.cpp](#).

7.13.5.8 `const string html_helper::c_right_menu_urls[NBR_RIGHT_MENU]` `[private]`

Initial value:

```
= {
    oConfig.soft_url + "?a=ie",
    oConfig.soft_url + "?a=d",
    oConfig.soft_url + "?a=io"
}
```

Definition at line 107 of file [html_helper.class.cpp](#).

7.13.5.9 `const string html_helper::c_salt = "k35f9vfgDtjTe1"`

Definition at line 9 of file [html_helper.class.cpp](#).

7.13.5.10 `std::map<int, string> html_helper::scale125i2s` `[private]`

a scale from 1 to 5, int to string

Definition at line 45 of file [html_helper.class.cpp](#).

7.13.5.11 `const string html_helper::seriesColorsGoalPlot = "['rgba(160, 0, 0, 0.6)', 'rgba(0, 0, 160, 0.6)', 'rgba(0, 160, 0, 0.6)',
'\#000\','\#663\','\#958c12\','\#953579\','\#4b5de4\','\#d8b83f\','\#ff5800\','\#0085cc\']"`

Definition at line 19 of file [html_helper.class.cpp](#).

7.13.5.12 `std::map<string, int> html_helper::step_nbrs` `[private]`

Definition at line 55 of file [html_helper.class.cpp](#).

7.13.5.13 `STEPDEF html_helper::steps` `[private]`

Definition at line 48 of file [html_helper.class.cpp](#).

7.13.5.14 `std::map<string, int> html_helper::sub_step_nbrs` `[private]`

Definition at line 56 of file [html_helper.class.cpp](#).

7.13.5.15 `std::map<int, STEPDEF> html_helper::sub_steps` [private]

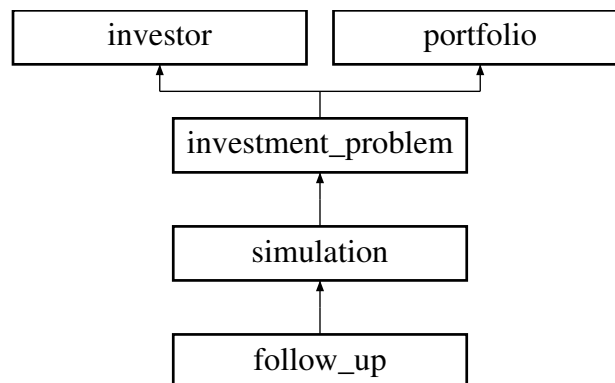
Definition at line 49 of file [html_helper.class.cpp](#).

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

- [html_helper.class.cpp](#)

7.14 investment_problem Class Reference

Inheritance diagram for `investment_problem`:



Public Member Functions

- [investment_problem](#) (int investor)
Constructor.
- void [solve](#) ()
allocates means (with a benchmark) to goals
- int [get_nbr_goals](#) ()
- int [get_max_month_for_lower_goals](#) (int g)
- void [javaGraph](#) (int g, string xtra_var="", string xtra_label="", int followup_mnth=0)
- void [prepare_javaVars_colors](#) (int g, int followup_mnth)
- void [parse_javaGraph](#) (int g, string xtra_var="", string xtra_label="")
plots the chartOverview
- string [get_full_name](#) ()
- float [age](#) (int Mnbr=0)
- int [months2simulate](#) ()
- int [load_from_db](#) (string investorID)
- void [load_ER](#) (int person=0, char person_type='i')
- void [save_ER](#) (int person=0, char person_type='i')
- void [load_covar](#) (int person, char person_type='i')
- void [save_covar](#) (int person, char person_type='i')
- bool [get_portf_from_db](#) (int id)
- string [get_description](#) ()
- void [set_mu](#) (float(*AssetClass_mu))
- void [set_sigma](#) (float(*AssetClass_varCov))

Public Attributes

- `std::map< int, aGOAL > goalZ`
map containing all goals
- `int investor_id`
- `string user_name`
- `string first_name`
- `string last_name`
- `string password`
- `string currency`
the default currency for that customer
- `struct tm birth_date`
- `float simulate_till_age`
- `float * assetClass_covar = new float[(NBR_ASSET_CLASSES + 1) * (NBR_ASSET_CLASSES + 1)]`
- `float assetClass_mu [(NBR_ASSET_CLASSES+1)]`
note: index 0 not used
- `std::map< int, string > assetClass_name`
note: index 0 not used
- `std::map< int, float > ER`
- `std::map< int, std::map< int, float > > covar`
- `int portfolio_id`
- `string description`
- `float weights [NBR_ASSET_CLASSES]`
the weights of the asset classes (ordered as the asset classes)
- `float pMu`
the MONTHLY return
- `float pLogR`
the MONTHLY log-return
- `float pSigma`
the MONTHLY volatility

Protected Member Functions

- `void set_followup_color_prae (float Vlow, float Vmed, float Vhigh, int g)`
- `void set_followup_color_post (float V, int g)`
- `int age2monthNbr (float theAge)`
- `string dateStr2Age (string the_date)`
- `bool save ()`
- `int get_from_db (string investorID, string password)`
- `bool add_to_db ()`
- `bool exists (int iid)`
- `void load_preferences ()`
sets the experience, knowledge and desirability vectors
- `bool save_preferences ()`
sets the experience, knowledge and desirability vectors
- `void set_max_exposure ()`
sets the max_exposure vector based on the experience, knowledge and desirability
- `bool set_scale (string scale_type, int ac, int val)`
- `void reload_ER (int person, char person_type= 'i')`
TODO: the ER and assetClass_mu as well as the covar and assetClass_covar are redundant ==> eliminate the C-style assetClass_.*
- `void reload_covar (int person, char what2do, char person_type= 'i')`

Protected Attributes

- `std::map< int, portfolio > portfolios`
the standard portfolios
- `std::map< int, int > experience`
- `std::map< int, int > knowledge`
- `std::map< int, int > desirability`
- `std::map< int, float > max_exposure`
the maximum exposure per asset class

Private Member Functions

- `void set_assets ()`
add the assets to means[1]
- `void set_cfs ()`
- `void set_goals ()`
- `void set_means_goal (int g)`
set the cash flows that are implied by the goal (eg. regular income from portfolio)
- `void expand_cf (float amount, string freq="M", string fromStr="", string tillStr="", int goal_nbr=0)`
- `void set_portfolios ()`
sets the portfolios
- `float get_optimal_portfolio (int g)`
finds the portfolio with minimal investment
- `float calc_alpha (int priority, int monthNbr)`
estimates \$\$
- `void set_priorityLimits ()`
sets priorityMax and priorityMin
- `float goal_seek_means (int g, int p)`
finds the minimal means needed (first free, then remaining, etc); sets tmp_means and calls the next function
- `void goal_seek_tmp_means (float *perc, int g, int p)`
finds the minimal percentage needed of tmp_means
- `float calc_risk (int g, int p)`
calculate the risk
- `float calc_V_high (int g, int p=0)`
calculates the 1-alpha quantile portfolio for the goal alpha at realization_monthNbr
- `void set_goal_color (int g, float opt_perc)`
allocates a color to each goal after the benchmark is found
- `void set_goal_remarks (int g)`
adds remarks about the benchmark after the benchmark is found
- `void allocate_to_unallocated_goal (int unallocated_goal_nbr)`
if necessary create an unallocated goal and put the means_left there
- `int get_portfolio_safest ()`
returns the portfolio id of the safest acceptable portfolio
- `int get_portfolio_riskiest ()`
returns the portfolio id of the riskiest acceptable portfolio
- `bool set_unallocated_goal ()`

Private Attributes

- `std::map< int, float > means`
sum of cash flows and assets per month (not to be changed)
- `std::map< int, float > means_left`
the percentage left to be invested of each cash-flow (carried over to next goal)
- `std::map< int, float > means_used`
the means already used for THIS particular goal
- `std::map< int, float > means_block`
next block of means to be used for a certain percentage
- `std::map< int, float > means_tmp`
copy of block for goal-seek
- `int nbr_portfolios`
the number of standard portfolios
- `int nbr_goals`
the total number of goals
- `int nbr_assets = 0`
the total number of assets
- `int nbr_cfs`
the total number of cash flows
- `std::map< int, bool > portfolio_suitable`
- `int priorityMax`
- `int priorityMin`

7.14.1 Detailed Description

class [investment_problem](#)

copyright: (c) Philippe De Brouwer 2014

last modification: 2014-08-25

This class is the cornerstone of this software. This class solves the investment problem. The key attribute to access this class is the function [solve\(\)](#), that can be called once an investor is loaded and he/she has stated goals and reported assets and savings capacity.

Definition at line 15 of file [investment_problem.class.cpp](#).

7.14.2 Constructor & Destructor Documentation

7.14.2.1 `investment_problem::investment_problem (int investor)`

Constructor.

CONSTRUCTOR : [investment_problem](#) initialize assets, cash flows and goals ==> all but the goals are converted to the base currency for the customer.

initialize the market parameters

some testing

Definition at line 85 of file [investment_problem.class.cpp](#).

7.14.3 Member Function Documentation

7.14.3.1 `bool investor::add_to_db ()` `[protected]`, `[inherited]`

`add_to_db`

does not check data any more, simply tries to push it to the DB only returns 0 if the userID already exists.

Definition at line 313 of file [investor.class.cpp](#).

7.14.3.2 `float investor::age (int Mnbr = 0)` `[inline]`, `[inherited]`

returns the age of a customer in years (with decimal part)

Definition at line 108 of file [investor.class.cpp](#).

7.14.3.3 `int investor::age2monthNbr (float theAge)` `[inline]`, `[protected]`, `[inherited]`

`age2monthNbr`

returns the month number for a given age note that this is negative for ages younger than the actual age.

Definition at line 200 of file [investor.class.cpp](#).

7.14.3.4 `void investment_problem::allocate_to_unallocated_goal (int unallocated_goal_nbr)` `[private]`

if necessary create an unallocated goal and put the means_left there

`allocate_to_unallocated_goal`

if necessary create one goal of type "unallocated" and allocate all remaining resources to that goal

Definition at line 604 of file [investment_problem.class.cpp](#).

7.14.3.5 `float investment_problem::calc_alpha (int priority, int monthNbr)` `[inline]`, `[private]`

estimates \$\$

`calc_alpha` estimates \$\$ based on the priority rank as into

Definition at line 318 of file [investment_problem.class.cpp](#).

7.14.3.6 `float investment_problem::calc_risk (int g, int p)` `[private]`

calculate the risk

`calc_risk`

calculates the risk (Expected Shortfall) < the alpha'th quantile of the portfolio

< expected value of the portfolio

< the cash flows for a month (= means_used[m] + means_tmp[m] - goalZ[g].means_goal[m];)

< the weighted average standard deviation of the portfolio

< a reweight of the month number to account for large cash flows

Parameters

<i>g</i>	the number of the goal
<i>p</i>	the number of the portfolio

Definition at line 809 of file [investment_problem.class.cpp](#).

7.14.3.7 float investment_problem::calc_V_high (int *g*, int *p* = 0) [private]

calculates the 1-alpha quantile portfolio for the goal alpha at realization_monthNbr

calc_V_high

calculates the 1-alpha quantile (for the highest limit of alpha that is plotted) at goal expiration.

Definition at line 1260 of file [investment_problem.class.cpp](#).

7.14.3.8 string investor::dateStr2Age (string *the_date*) [inline],[protected],[inherited]

dateStr2Age

Definition at line 209 of file [investor.class.cpp](#).

7.14.3.9 bool investor::exists (int *iid*) [protected],[inherited]

exists

checks if a given investor-id really exists

Definition at line 343 of file [investor.class.cpp](#).

7.14.3.10 void investment_problem::expand_cf (float *amount*, string *freq* = "M", string *fromStr* = "", string *tillStr* = "", int *goal_nbr* = 0) [private]

expand_CF

the string from and till are in the format YYYY-MM-DD

Definition at line 155 of file [investment_problem.class.cpp](#).

7.14.3.11 string portfolio::get_description () [inherited]

returns the label of the portfolio

Definition at line 86 of file [portfolio.class.cpp](#).

7.14.3.12 int investor::get_from_db (string *uid*, string *password*) [protected],[inherited]

get_from_db returns the investor_id if successful (otherwise 0)

Definition at line 249 of file [investor.class.cpp](#).

7.14.3.13 string investor::get_full_name () [inherited]

overloading << operator overload the << operator to output the goal in a readable format get_full_name returns the full name of an investor

Definition at line 103 of file [investor.class.cpp](#).

7.14.3.14 `int investment_problem::get_max_month_for_lower_goals (int g)`

returns the maximum term of all the other LOWER ranking goals. The higher ranking goals already have allocated all the means that they require, so we can disregard them here. `goal_type = 0` is the unallocated goal type, so this goal should not be taken into account here.

Definition at line 945 of file [investment_problem.class.cpp](#).

7.14.3.15 `int investment_problem::get_nbr_goals ()`

returns the number of goals

Definition at line 936 of file [investment_problem.class.cpp](#).

7.14.3.16 `float investment_problem::get_optimal_portfolio (int g) [private]`

finds the portfolio with minimal investment

`get_optimal_portfolio`

finds the optimal portfolio by finding that one that needs the least resources to satisfy the investment problem and adapts `means_left` accordingly calculate for each portfolio how much means are used:

find the optimal portfolio

< NOTE: the consequence of the = sign is that this will select the last portfolio if the goal is not attainable!! (eg. the 100% equity portfolio)

Definition at line 677 of file [investment_problem.class.cpp](#).

7.14.3.17 `bool portfolio::get_portf_from_db (int id) [inherited]`

load the definitions from the database note: we have to create a second instance of the sql connection as this is called while the object db is in use!

Definition at line 32 of file [portfolio.class.cpp](#).

7.14.3.18 `int investment_problem::get_portfolio_riskiest () [private]`

returns the portfolio id of the riskiest acceptable portfolio

`get_portfolio_riskiest`

Definition at line 1363 of file [investment_problem.class.cpp](#).

7.14.3.19 `int investment_problem::get_portfolio_safest () [private]`

returns the portfolio id of the safest acceptable portfolio

`get_portfolio_safest`

Definition at line 1345 of file [investment_problem.class.cpp](#).

7.14.3.20 `float investment_problem::goal_seek_means (int g, int p) [private]`

finds the minimal means needed (first free, then remaining, etc); sets `tmp_means` and calls the next function

`goal_seek_means`

returns the percentage of the given block of *this->means_block* that is needed to satisfy the given goal and the given portfolio. One should seek to get all these percentages and then find the lowest one (that will then indicate which portfolio to use)

NOTE: it leaves *means_block* untouched try if 0% works (if goal is in the past, etc.)

try if 100% is sufficient, if yes iterate, if not return 100%

if we're still here, then it is possible and we goal-seek the percentage

TODO: FOR INCOME FROM/TO WE SHOULD FIRST TRY IF ONE CAN SAVE ONLY BEFORE THE START_DATE

Parameters

<i>g</i>	the goal to be satisfied
<i>p</i>	the portfolio to be used

Definition at line 745 of file [investment_problem.class.cpp](#).

7.14.3.21 `void investment_problem::goal_seek_tmp_means (float * perc, int g, int p) [inline],[private]`

finds the minimal percentage needed of *tmp_means*

goal_seek_tmp_means this if statement could be omitted but it should save time for the asap goals

Definition at line 779 of file [investment_problem.class.cpp](#).

7.14.3.22 `void investment_problem::javaGraph (int g, string xtra_var = "", string xtra_label = "", int followup_mnth = 0)`

javaGraph

plots the overview-graph for the given goal

xtav_Var and *xtra_label* add a series of data that is delivered in text format *xtra_var* = [45,2000], [45.5,2200] *xtra_label* = "history" *followup_mnth* = 0 for feedback and simulation screen, and for the follow up screen it is the number of months we look into the future.

Definition at line 1089 of file [investment_problem.class.cpp](#).

7.14.3.23 `void market::load_covar (int person, char person_type = 'i') [inherited]`

load_covar < note: *person_type* = 'i' for persons (later to provide for 'a' advisor, organization, etc.)

< TODO eliminate this

if we did not find a personalized expectation for each asset class, then we load the default values

< TODO eliminate this

Definition at line 135 of file [market.class.cpp](#).

7.14.3.24 `void market::load_ER (int person = 0, char person_type = 'i') [inherited]`

load_ER < note: *person_type* = 'i' for persons (later to provide for 'a' advisor, organization, etc.)

Definition at line 61 of file [market.class.cpp](#).

7.14.3.25 `int investor::load_from_db (string uid) [inherited]`

returns the *investor_id* if successful (otherwise 0) NOTE for user interaction use *get_from_db(uid, password)* for password verification!!

Definition at line 134 of file [investor.class.cpp](#).

7.14.3.26 `void investor::load_preferences () [protected],[inherited]`

sets the experience, knowledge and desirability vectors

set_preferences

Definition at line 383 of file [investor.class.cpp](#).

7.14.3.27 `int investor::months2simulate () [inline],[inherited]`

months2simulate

Definition at line 124 of file [investor.class.cpp](#).

7.14.3.28 `void investment_problem::parse_javaGraph (int g, string xtra_var = "", string xtra_label = "")`

plots the chartOverview

parse_javaGraph

parses the line plot of the three scenarios and the goal to the output (webbrowser). It parses as well the that will get become the plot as the `<script>` with javascript that will produce the plot.

Definition at line 1103 of file [investment_problem.class.cpp](#).

7.14.3.29 `void investment_problem::prepare_javaVars_colors (int g, int followup_mnth)`

prepare_javaVars_colors

followup_mnth is used as the startmonth for the followUp-Screen (in all other graphs it will be zero – its default)

exp = expected value med = median low = alpha quantile high = 1- alpha quantile < the weighted average of the number of months that investments are in portfolio

TODO in the function below use another rate for NEGATIVE amounts, to mymic lending!

parse_javaGraph(g, s1, s2, s3, s4, xtra_var, xtra_label);

< place al the variables

Definition at line 970 of file [investment_problem.class.cpp](#).

7.14.3.30 `void market::reload_covar (int person, char what2do, char person_type = ' i ') [protected],[inherited]`

reload_vol

the situation here is more complex as the user expect that only the volatilities will be reset to the default values and not the correlations ...

what2do = [v|c|b] as in [vol | corr | both] < both

< volatilities

< correlations

Definition at line 281 of file [market.class.cpp](#).

7.14.3.31 `void market::reload_ER (int person, char person_type = ' i ') [protected],[inherited]`

TODO: the ER and assetClass_mu as well as the covar and assetClass_covar are redundant ==> eliminate the C-style assetClass_*.

set_mu

the expect value for all asset classes `set_covar`

the expect value for all assets `reload_ER`

note: `person_type = 'i'` for persons (later to provide for 'a' advisor, organization, etc.)

this function simply deletes the personalized expected returns. This is sufficient because later we check if they exists and only then load them (see function [load_ER\(\)](#)).

Definition at line 247 of file [market.class.cpp](#).

7.14.3.32 `bool investor::save ()` `[protected]`, `[inherited]`

`save`

Definition at line 220 of file [investor.class.cpp](#).

7.14.3.33 `void market::save_covar (int person, char person_type = 'i')` `[inherited]`

`save_covar`

Definition at line 172 of file [market.class.cpp](#).

7.14.3.34 `void market::save_ER (int person = 0, char person_type = 'i')` `[inherited]`

`save_ER`

Definition at line 109 of file [market.class.cpp](#).

7.14.3.35 `bool investor::save_preferences ()` `[protected]`, `[inherited]`

sets the experience, knowledge and desirability vectors

[save_preferences\(\)](#)

Definition at line 416 of file [investor.class.cpp](#).

7.14.3.36 `void investment_problem::set_assets ()` `[private]`

add the assets to `means[1]`

`set_assets`

puts the value of the today's savings into `means[1]` (all converted to the base currency of the investor)

Definition at line 122 of file [investment_problem.class.cpp](#).

7.14.3.37 `void investment_problem::set_cfs ()` `[private]`

`set_cfs`

load, expand and sum the expansion of all cash flows into the variable `means[]` (all converted to the base currency of the investor)

Definition at line 202 of file [investment_problem.class.cpp](#).

7.14.3.38 `void investment_problem::set_followup_color_post (float V, int g)` `[inline]`, `[protected]`

`set_followup_color_post` sets the color of the goal in case the simulation is LONGER than the realization date of the goal

Definition at line 1246 of file [investment_problem.class.cpp](#).

7.14.3.39 `void investment_problem::set_followup_color_prae (float Vlow, float Vmed, float Vhigh, int g)` `[inline]`,
`[protected]`

`set_followup_color_prae`

sets the color of the goal in case the simulations ends PRIOR to the realization date of the goal

Definition at line 1234 of file [investment_problem.class.cpp](#).

7.14.3.40 `void investment_problem::set_goal_color (int g, float opt_perc)` `[private]`

allocates a color to each goal after the benchmark is found

`set_color_info()` set the appropriate color and remarks

Definition at line 546 of file [investment_problem.class.cpp](#).

7.14.3.41 `void investment_problem::set_goal_remarks (int g)` `[private]`

adds remarks about the benchmark after the benchmark is found

`set_goal_remarks`

Definition at line 588 of file [investment_problem.class.cpp](#).

7.14.3.42 `void investment_problem::set_goals ()` `[private]`

loads all goals into `this->goalZ[]` the `realization_monthNbr` and `target_amount` have to be set in function of the `goal_type`

calculate the means that are implied by the goal (eg as income "from/till"

and means that are allocated to this goal)

if someone has changed his/her age this has to be corrected for the `means[m]` vectors not to run out of index

Definition at line 225 of file [investment_problem.class.cpp](#).

7.14.3.43 `void investor::set_max_exposure ()` `[protected]`, `[inherited]`

sets the `max_exposure` vector based on the experience, knowledge and desirability

`get_max_exposure`

sets the map `max_exposure` in this object

Definition at line 363 of file [investor.class.cpp](#).

7.14.3.44 `void investment_problem::set_means_goal (int g)` `[private]`

set the cash flows that are implied by the goal (eg. regular income from portfolio)

`set_goalZ[g].means_goal(int g)`

only to be used for goals of type "income from/to"

creates the cash flow related to the goal and puts it into `goalZ[g].means_goal` `add_to_means` is set to false → the amount will be added to `goalZ[g].means_goal`

< initialize means_left for g=1 (all in [investor.currency](#))
 < initialize means_left for g=1 (all in [investor.currency](#))
 < do nothing right now for unallocated goals, so take the next goal (for loop)
 try the free means (not used by other –lower ranking– goals)
 < initialize
 and means that are allocated to this goal):
 for the asap-goals we take the first cash flows first
 for all other goal types we take the last possible
 if the free means are not sufficient, try all the previous also
 block the resources for this goal
 set the benchmark
 set comments and colors:
 add the risk info
 put all the remaining means into the unallocated goal
 Definition at line 396 of file [investment_problem.class.cpp](#).

7.14.4 Member Data Documentation

7.14.4.1 `float* market::assetClass_covar = new float[(NBR_ASSET_CLASSES + 1) * (NBR_ASSET_CLASSES + 1)]`
 [inherited]

Definition at line 16 of file [market.class.cpp](#).

7.14.4.2 `float market::assetClass_mu[(NBR_ASSET_CLASSES+1)]` [inherited]

note: index 0 not used

Definition at line 17 of file [market.class.cpp](#).

7.14.4.3 `std::map<int, string> market::assetClass_name` [inherited]

note: index 0 not used

Definition at line 18 of file [market.class.cpp](#).

7.14.4.4 `struct tm investor::birth_date` [inherited]

Definition at line 20 of file [investor.class.cpp](#).

7.14.4.5 `std::map<int, std::map<int, float>> market::covar` [inherited]

Definition at line 26 of file [market.class.cpp](#).

7.14.4.6 `string investor::currency` [inherited]

the default currency for that customer

Definition at line 19 of file [investor.class.cpp](#).

7.14.4.7 `string portfolio::description` [inherited]

Definition at line 15 of file [portfolio.class.cpp](#).

7.14.4.8 `std::map<int, int> investor::desirability` [protected], [inherited]

Definition at line 41 of file [investor.class.cpp](#).

7.14.4.9 `std::map<int, float> market::ER` [inherited]

Definition at line 25 of file [market.class.cpp](#).

7.14.4.10 `std::map<int, int> investor::experience` [protected], [inherited]

Definition at line 39 of file [investor.class.cpp](#).

7.14.4.11 `string investor::first_name` [inherited]

Definition at line 16 of file [investor.class.cpp](#).

7.14.4.12 `std::map<int, aGOAL> investment_problem::goalZ`

map containing all goals

Definition at line 22 of file [investment_problem.class.cpp](#).

7.14.4.13 `int investor::investor_id` [inherited]

Definition at line 14 of file [investor.class.cpp](#).

7.14.4.14 `std::map<int, int> investor::knowledge` [protected], [inherited]

Definition at line 40 of file [investor.class.cpp](#).

7.14.4.15 `string investor::last_name` [inherited]

Definition at line 17 of file [investor.class.cpp](#).

7.14.4.16 `std::map<int, float> investor::max_exposure` [protected], [inherited]

the maximum exposure per asset class

Definition at line 42 of file [investor.class.cpp](#).

7.14.4.17 `std::map<int, float> investment_problem::means` [private]

sum of cash flows and assets per month (not to be changed)

Definition at line 35 of file [investment_problem.class.cpp](#).

7.14.4.18 `std::map<int, float> investment_problem::means_block` [private]

next block of means to be used for a certain percentage

Definition at line 38 of file [investment_problem.class.cpp](#).

7.14.4.19 `std::map<int, float> investment_problem::means_left` [private]

the percentage left to be invested of each cash-flow (carried over to next goal)

Definition at line 36 of file [investment_problem.class.cpp](#).

7.14.4.20 `std::map<int, float> investment_problem::means_tmp` [private]

copy of block for goal-seek

Definition at line 39 of file [investment_problem.class.cpp](#).

7.14.4.21 `std::map<int, float> investment_problem::means_used` [private]

the means already used for THIS particular goal

Definition at line 37 of file [investment_problem.class.cpp](#).

7.14.4.22 `int investment_problem::nbr_assets = 0` [private]

the total number of assets

Definition at line 44 of file [investment_problem.class.cpp](#).

7.14.4.23 `int investment_problem::nbr_cfs` [private]

the total number of cash flows

Definition at line 45 of file [investment_problem.class.cpp](#).

7.14.4.24 `int investment_problem::nbr_goals` [private]

the total number of goals

Definition at line 43 of file [investment_problem.class.cpp](#).

7.14.4.25 `int investment_problem::nbr_portfolios` [private]

the number of standard portfolios

Definition at line 41 of file [investment_problem.class.cpp](#).

7.14.4.26 `string investor::password` [inherited]

Definition at line 18 of file [investor.class.cpp](#).

7.14.4.27 float portfolio::pLogR [inherited]

the MONTHLY log-return

Definition at line 25 of file [portfolio.class.cpp](#).

7.14.4.28 float portfolio::pMu [inherited]

the MONTHLY return

Definition at line 24 of file [portfolio.class.cpp](#).

7.14.4.29 int portfolio::portfolio_id [inherited]

Definition at line 14 of file [portfolio.class.cpp](#).

7.14.4.30 std::map<int, bool> investment_problem::portfolio_suitable [private]

Definition at line 67 of file [investment_problem.class.cpp](#).

7.14.4.31 std::map<int, portfolio> investment_problem::portfolios [protected]

the standard portfolios

Definition at line 29 of file [investment_problem.class.cpp](#).

7.14.4.32 int investment_problem::priorityMax [private]

Definition at line 72 of file [investment_problem.class.cpp](#).

7.14.4.33 int investment_problem::priorityMin [private]

Definition at line 73 of file [investment_problem.class.cpp](#).

7.14.4.34 float portfolio::pSigma [inherited]

the MONTHLY volatility

Definition at line 26 of file [portfolio.class.cpp](#).

7.14.4.35 float investor::simulate_till_age [inherited]

Definition at line 21 of file [investor.class.cpp](#).

7.14.4.36 string investor::user_name [inherited]

Definition at line 15 of file [investor.class.cpp](#).

7.14.4.37 float portfolio::weights[NBR_ASSET_CLASSES] [inherited]

the weights of the asset classes (ordered as the asset classes)

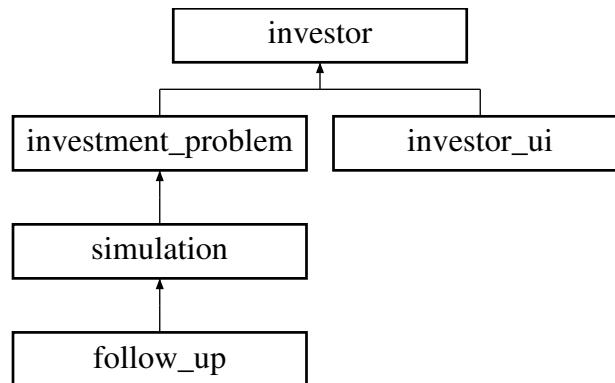
Definition at line 17 of file [portfolio.class.cpp](#).

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

- [investment_problem.class.cpp](#)

7.15 investor Class Reference

Inheritance diagram for investor:



Public Member Functions

- string [get_full_name](#) ()
- float [age](#) (int Mnbr=0)
- int [months2simulate](#) ()
- int [load_from_db](#) (string investorID)
- void [load_ER](#) (int person=0, char person_type= 'i')
- void [save_ER](#) (int person=0, char person_type= 'i')
- void [load_covar](#) (int person, char person_type= 'i')
- void [save_covar](#) (int person, char person_type= 'i')

Public Attributes

- int [investor_id](#)
- string [user_name](#)
- string [first_name](#)
- string [last_name](#)
- string [password](#)
- string [currency](#)
the default currency for that customer
- struct tm [birth_date](#)
- float [simulate_till_age](#)
- float * [assetClass_covar](#) = new float[(NBR_ASSET_CLASSES + 1) * (NBR_ASSET_CLASSES + 1)]
- float [assetClass_mu](#) [(NBR_ASSET_CLASSES+1)]
note: index 0 not used
- std::map< int, string > [assetClass_name](#)
note: index 0 not used
- std::map< int, float > [ER](#)
- std::map< int, std::map< int, float > > [covar](#)

Protected Member Functions

- int [age2monthNbr](#) (float theAge)
- string [dateStr2Age](#) (string the_date)
- bool [save](#) ()
- int [get_from_db](#) (string investorID, string [password](#))
- bool [add_to_db](#) ()
- bool [exists](#) (int iid)
- void [load_preferences](#) ()
 - sets the experience, knowledge and desirability vectors*
- bool [save_preferences](#) ()
 - sets the experience, knowledge and desirability vectors*
- void [set_max_exposure](#) ()
 - sets the max_exposure vector based on the experience, knowledge and desirability*
- bool [set_scale](#) (string scale_type, int ac, int val)
- [investor](#) ()
- void [reload_ER](#) (int person, char person_type= 'i')
 - TODO: the ER and assetClass_mu as well as the covar and assetClass_covar are redundant ==> eliminate the C-style assetClass_*.*
- void [reload_covar](#) (int person, char what2do, char person_type= 'i')

Protected Attributes

- std::map< int, int > [experience](#)
- std::map< int, int > [knowledge](#)
- std::map< int, int > [desirability](#)
- std::map< int, float > [max_exposure](#)
 - the maximum exposure per asset class*

7.15.1 Detailed Description

class investor

copyright: (c) Philippe De Brouwer 2014

last modification:

Definition at line 11 of file [investor.class.cpp](#).

7.15.2 Constructor & Destructor Documentation

7.15.2.1 `investor::investor ()` [`inline`], [`protected`]

CONSTRUCTOR

needed because initalization of vectors cannot be done inside a class outside of a method

Definition at line 60 of file [investor.class.cpp](#).

7.15.3 Member Function Documentation

7.15.3.1 `bool investor::add_to_db ()` [`protected`]

`add_to_db`

does not check data any more, simply tries to push it to the DB only returns 0 if the userID already exists.

Definition at line 313 of file [investor.class.cpp](#).

7.15.3.2 `float investor::age (int Mnbr = 0) [inline]`

returns the age of a customer in years (with decimal part)

Definition at line 108 of file [investor.class.cpp](#).

7.15.3.3 `int investor::age2monthNbr (float theAge) [inline], [protected]`

age2monthNbr

returns the month number for a given age note that this is negative for ages younger than the actual age.

Definition at line 200 of file [investor.class.cpp](#).

7.15.3.4 `string investor::dateStr2Age (string the_date) [inline], [protected]`

dateStr2Age

Definition at line 209 of file [investor.class.cpp](#).

7.15.3.5 `bool investor::exists (int iid) [protected]`

exists

checks if a given investor-id really exists

Definition at line 343 of file [investor.class.cpp](#).

7.15.3.6 `int investor::get_from_db (string uid, string password) [protected]`

get_from_db returns the investor_id if successful (otherwise 0)

Definition at line 249 of file [investor.class.cpp](#).

7.15.3.7 `string investor::get_full_name ()`

overloading << operator overload the << operator to output the goal in a readable format get_full_name returns the full name of an investor

Definition at line 103 of file [investor.class.cpp](#).

7.15.3.8 `void market::load_covar (int person, char person_type = 'i') [inherited]`

load_covar < note: person_type = 'i' for persons (later to provide for 'a' advisor, organization, etc.)

< TODO eliminate this

if we did not find a personalized expectation for each asset class, then we load the default values

< TODO eliminate this

Definition at line 135 of file [market.class.cpp](#).

7.15.3.9 `void market::load_ER (int person = 0, char person_type = 'i') [inherited]`

load_ER < note: person_type = 'i' for persons (later to provide for 'a' advisor, organization, etc.)

Definition at line 61 of file [market.class.cpp](#).

7.15.3.10 `int investor::load_from_db (string uid)`

returns the investor_id if successful (otherwise 0) NOTE for user interaction use get_from_db(uid, password) for password verification!!

Definition at line 134 of file [investor.class.cpp](#).

7.15.3.11 `void investor::load_preferences () [protected]`

sets the experience, knowledge and desirability vectors

set_preferences

Definition at line 383 of file [investor.class.cpp](#).

7.15.3.12 `int investor::months2simulate () [inline]`

months2simulate

Definition at line 124 of file [investor.class.cpp](#).

7.15.3.13 `void market::reload_covar (int person, char what2do, char person_type = ' i') [protected], [inherited]`

reload_vol

the situation here is more complex as the user expect that only the volatilities will be reset to the default values and not the correlations ...

what2do = [v|c|b] as in [vol | corr | both] < both

< volatilities

< correlations

Definition at line 281 of file [market.class.cpp](#).

7.15.3.14 `void market::reload_ER (int person, char person_type = ' i') [protected], [inherited]`

TODO: the ER and assetClass_mu as well as the covar and assetClass_covar are redundant ==> eliminate the C-style assetClass_*.

set_mu

the expect value for all asset classes set_covar

the expect value for all assets reload_ER

note: person_type = 'i' for persons (later to provide for 'a' advisor, organization, etc.)

this function simply deletes the personalized expected returns. This is sufficient because later we check if they exists and only then load them (see function [load_ER\(\)](#)).

Definition at line 247 of file [market.class.cpp](#).

7.15.3.15 `bool investor::save () [protected]`

save

Definition at line 220 of file [investor.class.cpp](#).

7.15.3.16 `void market::save_covar (int person, char person_type = ' i')` [inherited]

`save_covar`

Definition at line 172 of file [market.class.cpp](#).

7.15.3.17 `void market::save_ER (int person = 0, char person_type = ' i')` [inherited]

`save_ER`

Definition at line 109 of file [market.class.cpp](#).

7.15.3.18 `bool investor::save_preferences ()` [protected]

sets the experience, knowledge and desirability vectors

[save_preferences\(\)](#)

Definition at line 416 of file [investor.class.cpp](#).

7.15.3.19 `void investor::set_max_exposure ()` [protected]

sets the max_exposure vector based on the experience, knowledge and desirability

`get_max_exposure`

sets the map max_exposure in this object

Definition at line 363 of file [investor.class.cpp](#).

7.15.3.20 `bool investor::set_scale (string scale_type, int ac, int val)` [protected]

`set_scale`

Definition at line 402 of file [investor.class.cpp](#).

7.15.4 Member Data Documentation

7.15.4.1 `float* market::assetClass_covar = new float[(NBR_ASSET_CLASSES + 1) * (NBR_ASSET_CLASSES + 1)]`
[inherited]

Definition at line 16 of file [market.class.cpp](#).

7.15.4.2 `float market::assetClass_mu[(NBR_ASSET_CLASSES+1)]` [inherited]

note: index 0 not used

Definition at line 17 of file [market.class.cpp](#).

7.15.4.3 `std::map<int, string> market::assetClass_name` [inherited]

note: index 0 not used

Definition at line 18 of file [market.class.cpp](#).

7.15.4.4 struct tm investor::birth_date

Definition at line 20 of file [investor.class.cpp](#).

7.15.4.5 std::map<int, std::map<int, float> > market::covar [inherited]

Definition at line 26 of file [market.class.cpp](#).

7.15.4.6 string investor::currency

the default currency for that customer

Definition at line 19 of file [investor.class.cpp](#).

7.15.4.7 std::map<int, int> investor::desirability [protected]

Definition at line 41 of file [investor.class.cpp](#).

7.15.4.8 std::map<int, float> market::ER [inherited]

Definition at line 25 of file [market.class.cpp](#).

7.15.4.9 std::map<int, int> investor::experience [protected]

Definition at line 39 of file [investor.class.cpp](#).

7.15.4.10 string investor::first_name

Definition at line 16 of file [investor.class.cpp](#).

7.15.4.11 int investor::investor_id

Definition at line 14 of file [investor.class.cpp](#).

7.15.4.12 std::map<int, int> investor::knowledge [protected]

Definition at line 40 of file [investor.class.cpp](#).

7.15.4.13 string investor::last_name

Definition at line 17 of file [investor.class.cpp](#).

7.15.4.14 std::map<int, float> investor::max_exposure [protected]

the maximum exposure per asset class

Definition at line 42 of file [investor.class.cpp](#).

7.15.4.15 string investor::password

Definition at line 18 of file [investor.class.cpp](#).

7.15.4.16 float investor::simulate_till_age

Definition at line 21 of file [investor.class.cpp](#).

7.15.4.17 string investor::user_name

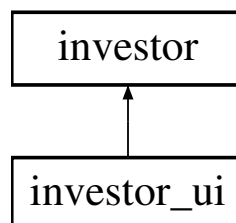
Definition at line 15 of file [investor.class.cpp](#).

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

- [investor.class.cpp](#)

7.16 investor_ui Class Reference

Inheritance diagram for investor_ui:



Public Member Functions

- [investor_ui](#) ()
- bool [set_investor_id](#) (const string &s)
- bool [set_user_name](#) (const string &s)
- bool [set_first_name](#) (const string &s)
- bool [set_last_name](#) (const string &s)
- bool [set_password](#) (const string &s)
- bool [set_currency](#) (const string &s)
- bool [set_birth_date](#) (const string &s)
- bool [set_simulate_till_age](#) (const string &s)
- bool [is_logged_in](#) ()
- void [logout](#) ()
- bool [set_investor_id_fromEnv](#) ()
- void [login](#) ()
- void [update_last_activity_at](#) ()
- string [parse_user_info](#) (bool parse_wrapper=true)
- void [home_screen](#) ()
- void [show_home_screen_form](#) ()
- void [register_form](#) ()
- void [register_exec](#) ()
- void [account_form](#) ()
- void [account_save](#) ()
- void [login_form](#) ()
- void [login_exec](#) ()
- void [show_disclaimer](#) ()
- void [personalize_screen](#) ()
- void [personalize_save](#) ()

- void [personalize_reload](#) (string action)
- void [show_inventory_form](#) ()
- void [asset_add](#) ()
- void [asset_edit](#) ()
- void [asset_save](#) ()
- void [asset_delete](#) ()
- void [goal_help_form](#) ()
- void [goal_help_save](#) ()
- void [goal_list](#) ()
- void [goal_add](#) ()
- void [goal_edit](#) ()
- void [goal_save](#) ()
- void [goal_delete](#) ()
- void [cash_flow_list](#) ()
- void [cash_flow_add](#) ()
- void [cash_flow_edit](#) ()
- void [cash_flow_save](#) ()
- void [cash_flow_delete](#) ()
- void [feedback](#) ()
- void [show_simulation](#) ()
- void [show_follow_up](#) ()
- void [show_total_portfolio](#) (investment_problem *oInvProbl)
- void [parse_savings_plan](#) (investment_problem *oInvProbl, int g)
- string [get_full_name](#) ()
- float [age](#) (int Mnbr=0)
- int [months2simulate](#) ()
- int [load_from_db](#) (string investorID)
- void [load_ER](#) (int person=0, char person_type= 'i')
- void [save_ER](#) (int person=0, char person_type= 'i')
- void [load_covar](#) (int person, char person_type= 'i')
- void [save_covar](#) (int person, char person_type= 'i')

Public Attributes

- int [active_action](#) = 1
- [html_helper](#) hh
- int [investor_id](#)
- string [user_name](#)
- string [first_name](#)
- string [last_name](#)
- string [password](#)
- string [currency](#)
the default currency for that customer
- struct tm [birth_date](#)
- float [simulate_till_age](#)
- float * [assetClass_covar](#) = new float[(NBR_ASSET_CLASSES + 1) * (NBR_ASSET_CLASSES + 1)]
- float [assetClass_mu](#) [(NBR_ASSET_CLASSES+1)]
note: index 0 not used
- std::map< int, string > [assetClass_name](#)
note: index 0 not used
- std::map< int, float > [ER](#)
- std::map< int, std::map< int, float > > [covar](#)

Protected Member Functions

- int [age2monthNbr](#) (float theAge)
- string [dateStr2Age](#) (string the_date)
- bool [save](#) ()
- int [get_from_db](#) (string investorID, string [password](#))
- bool [add_to_db](#) ()
- bool [exists](#) (int iid)
- void [load_preferences](#) ()
 - sets the experience, knowledge and desirability vectors*
- bool [save_preferences](#) ()
 - sets the experience, knowledge and desirability vectors*
- void [set_max_exposure](#) ()
 - sets the max_exposure vector based on the experience, knowledge and desirability*
- bool [set_scale](#) (string scale_type, int ac, int val)
- void [reload_ER](#) (int person, char person_type= 'i')
 - TODO: the ER and assetClass_mu as well as the covar and assetClass_covar are redundant ==> eliminate the C-style assetClass_*.*
- void [reload_covar](#) (int person, char what2do, char person_type= 'i')

Protected Attributes

- std::map< int, int > [experience](#)
- std::map< int, int > [knowledge](#)
- std::map< int, int > [desirability](#)
- std::map< int, float > [max_exposure](#)
 - the maximum exposure per asset class*

Private Member Functions

- void [show_register_form_body](#) ()
- void [show_account_form_body](#) ()
- void [show_goal_help_form_body](#) ()
- void [show_login_form_body](#) ()
- void [show_inventory_form_body](#) ()
- void [parse_bm](#) (std::map< int, float > benchmark, int ref_number)
- void [personalize_form](#) ()
- void [personalize_pref_table](#) ()
- void [personalize_ER_table](#) ()
- void [personalize_vol_table](#) ()
- void [personalize_ERvol_table](#) ()
- void [personalize_covar_table](#) ()
- void [parse_simulation_duration_form](#) (int nbrMonths, string action="s")
- void [parse_simulation_insight](#) (simulation *oSim)
- void [parse_form_open_personalize](#) (string the_content)
- void [parse_form_close_personalize](#) (string the_content, bool show_reload=true)
- string [button_to](#) (string go_to, string description, char type_redir)
- void [personalize_set_general](#) ()
- void [personalize_set_prefs](#) ()
- void [personalize_set_ER](#) ()
- void [personalize_set_vol](#) ()
- void [personalize_set_corr](#) ()
- void [parse_goal_box](#) (investment_problem *oInvProbl, int g, string s_var="", string s_desc="")

Private Attributes

- `std::map< int, string > asset_class_names`
this variable is to be filled out when needed

7.16.1 Detailed Description

class investor

copyright: (c) Philippe De Brouwer 2014

last modification: 2014-08-25

This class holds the basic functionality to use an investor: retrieve it from database, save, edit and load the _goal essential parameters related to the investor him/herself.

Definition at line 14 of file [investor_ui.class.cpp](#).

7.16.2 Constructor & Destructor Documentation

7.16.2.1 investor_ui::investor_ui ()

CONSTRUCTOR

Definition at line 126 of file [investor_ui.class.cpp](#).

7.16.3 Member Function Documentation

7.16.3.1 void investor_ui::account_form ()

account_form

Definition at line 383 of file [investor_ui.class.cpp](#).

7.16.3.2 void investor_ui::account_save ()

account_save saves the account information change password!!!!!!!

Definition at line 470 of file [investor_ui.class.cpp](#).

7.16.3.3 bool investor::add_to_db () `[protected]`, `[inherited]`

add_to_db

does not check data any more, simply tries to push it to the DB only returns 0 if the userID already exists.

Definition at line 313 of file [investor.class.cpp](#).

7.16.3.4 float investor::age (int Mnbr = 0) `[inline]`, `[inherited]`

returns the age of a customer in years (with decimal part)

Definition at line 108 of file [investor.class.cpp](#).

7.16.3.5 int investor::age2monthNbr (float theAge) `[inline]`, `[protected]`, `[inherited]`

age2monthNbr

returns the month number for a given age note that this is negative for ages younger than the actual age.

Definition at line 200 of file [investor.class.cpp](#).

7.16.3.6 void investor_ui::asset_add ()

Definition at line 1036 of file [investor_ui.class.cpp](#).

7.16.3.7 void investor_ui::asset_delete ()

Definition at line 1110 of file [investor_ui.class.cpp](#).

7.16.3.8 void investor_ui::asset_edit ()

asset_edit

Definition at line 1061 of file [investor_ui.class.cpp](#).

7.16.3.9 void investor_ui::asset_save ()

Definition at line 1084 of file [investor_ui.class.cpp](#).

7.16.3.10 string investor_ui::button_to (string go_to, string description, char type_redir) [inline], [private]

button_to

type_redir = {a = action, u = url, s = secondary action (variable a2 is set)} go_to is the action or the url

Definition at line 170 of file [investor_ui.class.cpp](#).

7.16.3.11 void investor_ui::cash_flow_add ()

Definition at line 1244 of file [investor_ui.class.cpp](#).

7.16.3.12 void investor_ui::cash_flow_delete ()

Definition at line 1301 of file [investor_ui.class.cpp](#).

7.16.3.13 void investor_ui::cash_flow_edit ()

Definition at line 1265 of file [investor_ui.class.cpp](#).

7.16.3.14 void investor_ui::cash_flow_list ()

Definition at line 1228 of file [investor_ui.class.cpp](#).

7.16.3.15 void investor_ui::cash_flow_save ()

Definition at line 1285 of file [investor_ui.class.cpp](#).

7.16.3.16 `string investor::dateStr2Age (string the_date)` `[inline]`, `[protected]`, `[inherited]`

dateStr2Age

Definition at line 209 of file [investor.class.cpp](#).

7.16.3.17 `bool investor::exists (int iid)` `[protected]`, `[inherited]`

exists

checks if a given investor-id really exists

Definition at line 343 of file [investor.class.cpp](#).

7.16.3.18 `void investor_ui::feedback ()`

feedback

provides the feedback about the goals, risks savings plans, etc.

Definition at line 1642 of file [investor_ui.class.cpp](#).

7.16.3.19 `int investor::get_from_db (string uid, string password)` `[protected]`, `[inherited]`

get_from_db returns the investor_id if successful (otherwise 0)

Definition at line 249 of file [investor.class.cpp](#).

7.16.3.20 `string investor::get_full_name ()` `[inherited]`

overloading << operator overload the << operator to output the goal in a readable format get_full_name returns the full name of an investor

Definition at line 103 of file [investor.class.cpp](#).

7.16.3.21 `void investor_ui::goal_add ()`

Definition at line 1149 of file [investor_ui.class.cpp](#).

7.16.3.22 `void investor_ui::goal_delete ()`

Definition at line 1209 of file [investor_ui.class.cpp](#).

7.16.3.23 `void investor_ui::goal_edit ()`

Definition at line 1173 of file [investor_ui.class.cpp](#).

7.16.3.24 `void investor_ui::goal_help_form ()`

goal_help

Definition at line 2106 of file [investor_ui.class.cpp](#).

7.16.3.25 void investor_ui::goal_help_save ()

goal_help_save saves the account information — Retirement

— Rainy Day

– Kids

Foreseeable expenses

Life Dreams

Definition at line 2118 of file [investor_ui.class.cpp](#).

7.16.3.26 void investor_ui::goal_list ()

goal_list < list existing and show input form for new items

Definition at line 1133 of file [investor_ui.class.cpp](#).

7.16.3.27 void investor_ui::goal_save ()

Definition at line 1193 of file [investor_ui.class.cpp](#).

7.16.3.28 void investor_ui::home_screen ()

home_screen

displays the landing screen for the software

Definition at line 255 of file [investor_ui.class.cpp](#).

7.16.3.29 bool investor_ui::is_logged_in ()

is_logged_in returns true if the user is logged in

Definition at line 1463 of file [investor_ui.class.cpp](#).

7.16.3.30 void market::load_covar (int *person*, char *person_type* = 'i') [inherited]

load_covar < note: person_type = 'i' for persons (later to provide for 'a' advisor, organization, etc.)

< TODO eliminate this

if we did not find a personalized expectation for each asset class, then we load the default values

< TODO eliminate this

Definition at line 135 of file [market.class.cpp](#).

7.16.3.31 void market::load_ER (int *person* = 0, char *person_type* = 'i') [inherited]

load_ER < note: person_type = 'i' for persons (later to provide for 'a' advisor, organization, etc.)

Definition at line 61 of file [market.class.cpp](#).

7.16.3.32 int investor::load_from_db (string *uid*) [inherited]

returns the investor_id if successful (otherwise 0) NOTE for user interaction use get_from_db(uid, password) for password verification!!

Definition at line 134 of file [investor.class.cpp](#).

7.16.3.33 void investor::load_preferences () [protected],[inherited]

sets the experience, knowledge and desirability vectors

set_preferences

Definition at line 383 of file [investor.class.cpp](#).

7.16.3.34 void investor_ui::login ()

login

executes the login of the user (updates the table " + oConfig.tbl_prefix + "_investors and adds the information that the user is logged in

Definition at line 1523 of file [investor_ui.class.cpp](#).

7.16.3.35 void investor_ui::login_exec ()

Definition at line 566 of file [investor_ui.class.cpp](#).

7.16.3.36 void investor_ui::login_form ()

Definition at line 552 of file [investor_ui.class.cpp](#).

7.16.3.37 void investor_ui::logout ()

logout logs out the currently logged in user then loads the home screen

Definition at line 1484 of file [investor_ui.class.cpp](#).

7.16.3.38 int investor::months2simulate () [inline],[inherited]

months2simulate

Definition at line 124 of file [investor.class.cpp](#).

7.16.3.39 void investor_ui::parse_bm (std::map< int, float > *benchmark*, int *goal_number*) [private]

parse_bm

the_goal

Definition at line 1881 of file [investor_ui.class.cpp](#).

7.16.3.40 void investor_ui::parse_form_close_personalize (string *the_content*, bool *show_reload* = true) [private]

parse_form_close_personalize

Definition at line 804 of file [investor_ui.class.cpp](#).

7.16.3.41 `void investor_ui::parse_form_open_personalize (string the_content) [private]`

`parse_form_open_personalize`

Definition at line 774 of file [investor_ui.class.cpp](#).

7.16.3.42 `void investor_ui::parse_goal_box (investment_problem * oInvProbl, int g, string s_var = "", string s_desc = "") [private]`

`parse_goal_box`

g = the goal number

Definition at line 1549 of file [investor_ui.class.cpp](#).

7.16.3.43 `void investor_ui::parse_savings_plan (investment_problem * oInvProbl, int g)`

`parse_savings_plan`

prints the savings plan to the screen

Definition at line 2314 of file [investor_ui.class.cpp](#).

7.16.3.44 `void investor_ui::parse_simulation_duration_form (int nbrMonths, string action = "s") [private]`

`parse_simulation_duration_form`

Definition at line 2398 of file [investor_ui.class.cpp](#).

7.16.3.45 `void investor_ui::parse_simulation_insight (simulation * oSim) [private]`

`parse_simulation_insight`

Definition at line 2425 of file [investor_ui.class.cpp](#).

7.16.3.46 `string investor_ui::parse_user_info (bool parse_wrapper = true)`

`parse_user_info`

Definition at line 133 of file [investor_ui.class.cpp](#).

7.16.3.47 `void investor_ui::personalize_covar_table () [private]`

`personalize_covar_table`

Definition at line 678 of file [investor_ui.class.cpp](#).

7.16.3.48 `void investor_ui::personalize_ER_table () [private]`

`personalize_ER_table`

Definition at line 623 of file [investor_ui.class.cpp](#).

7.16.3.49 `void investor_ui::personalize_ERvol_table () [private]`

`personalize_ERvol_table`

Definition at line 657 of file [investor_ui.class.cpp](#).

7.16.3.50 void investor_ui::personalize_form () [private]

personalize_form

Definition at line 712 of file [investor_ui.class.cpp](#).

7.16.3.51 void investor_ui::personalize_pref_table () [private]

personalize_pref_table

Definition at line 595 of file [investor_ui.class.cpp](#).

7.16.3.52 void investor_ui::personalize_reload (string action)

personalize_reload

reloads for the investor the system defaults of ER, vol and/or corr

Definition at line 957 of file [investor_ui.class.cpp](#).

7.16.3.53 void investor_ui::personalize_save ()

personalize_save std_message, error_message

Definition at line 838 of file [investor_ui.class.cpp](#).

7.16.3.54 void investor_ui::personalize_screen ()

personalize_screen

Definition at line 826 of file [investor_ui.class.cpp](#).

7.16.3.55 void investor_ui::personalize_set_corr () [private]

personalize_set_corr

Definition at line 935 of file [investor_ui.class.cpp](#).

7.16.3.56 void investor_ui::personalize_set_ER () [private]

personalize_set_ER

Definition at line 910 of file [investor_ui.class.cpp](#).

7.16.3.57 void investor_ui::personalize_set_general () [private]

personalize_set_general

Definition at line 865 of file [investor_ui.class.cpp](#).

7.16.3.58 void investor_ui::personalize_set_prefs () [private]

personalize_set_general

Definition at line 889 of file [investor_ui.class.cpp](#).

7.16.3.59 void investor_ui::personalize_set_vol () [private]

personalize_set_vol

Definition at line 922 of file [investor_ui.class.cpp](#).

7.16.3.60 void investor_ui::personalize_vol_table () [private]

personalize_vol_table

Definition at line 640 of file [investor_ui.class.cpp](#).

7.16.3.61 void investor_ui::register_exec ()

register_exec

executes the registration of a new user and if successful shows the asset-screen xxx goal_help is not defined et TODO

Definition at line 409 of file [investor_ui.class.cpp](#).

7.16.3.62 void investor_ui::register_form ()

Definition at line 322 of file [investor_ui.class.cpp](#).

7.16.3.63 void market::reload_covar (int person, char what2do, char person_type = ' i') [protected],
[inherited]

reload_vol

the situation here is more complex as the user expect that only the volatilities will be reset to the default values and not the correlations ...

what2do = [v|c|b] as in [vol | corr | both] < both

< volatilities

< correlations

Definition at line 281 of file [market.class.cpp](#).

7.16.3.64 void market::reload_ER (int person, char person_type = ' i') [protected],[inherited]

TODO: the ER and assetClass_mu as well as the covar and assetClass_covar are redundant ==> eliminate the C-style assetClass_*.

set_mu

the expect value for all asset classes set_covar

the expect value for all assets reload_ER

note: person_type = 'i' for persons (later to provide for 'a' advisor, organization, etc.)

this function simply deletes the personalized expected returns. This is sufficient because later we check if they exists and only then load them (see function [load_ER\(\)](#)).

Definition at line 247 of file [market.class.cpp](#).

7.16.3.65 bool investor::save () [protected],[inherited]

save

Definition at line 220 of file [investor.class.cpp](#).

7.16.3.66 void market::save_covar (int *person*, char *person_type* = ' i ') [inherited]

save_covar

Definition at line 172 of file [market.class.cpp](#).

7.16.3.67 void market::save_ER (int *person* = 0, char *person_type* = ' i ') [inherited]

save_ER

Definition at line 109 of file [market.class.cpp](#).

7.16.3.68 bool investor::save_preferences () [protected],[inherited]

sets the experience, knowledge and desirability vectors

[save_preferences\(\)](#)

Definition at line 416 of file [investor.class.cpp](#).

7.16.3.69 bool investor_ui::set_birth_date (const string & s)

bool set_birth_date(string s);

Definition at line 1430 of file [investor_ui.class.cpp](#).

7.16.3.70 bool investor_ui::set_currency (const string & s)

bool set_currency(const std::string& s);

sets the currency

Definition at line 1389 of file [investor_ui.class.cpp](#).

7.16.3.71 bool investor_ui::set_first_name (const string & s)

Definition at line 1343 of file [investor_ui.class.cpp](#).

7.16.3.72 bool investor_ui::set_investor_id (const string & s)

bool set_investor_id (const std::string& s);

sets the investor id in the object for an EXISTING user

Definition at line 1410 of file [investor_ui.class.cpp](#).

7.16.3.73 bool investor_ui::set_investor_id_fromEnv ()

Definition at line 1494 of file [investor_ui.class.cpp](#).

7.16.3.74 bool investor_ui::set_last_name (const string & s)

Definition at line 1358 of file [investor_ui.class.cpp](#).

7.16.3.75 `void investor::set_max_exposure ()` [protected],[inherited]

sets the max_exposure vector based on the experience, knowledge and desirability

get_max_exposure

sets the map max_exposure in this object

Definition at line 363 of file [investor.class.cpp](#).

7.16.3.76 `bool investor_ui::set_password (const string & s)`

Definition at line 1373 of file [investor_ui.class.cpp](#).

7.16.3.77 `bool investor::set_scale (string scale_type, int ac, int val)` [protected],[inherited]

set_scale

Definition at line 402 of file [investor.class.cpp](#).

7.16.3.78 `bool investor_ui::set_simulate_till_age (const string & s)`

`bool set_simulate_till_age(string s);`

Definition at line 1447 of file [investor_ui.class.cpp](#).

7.16.3.79 `bool investor_ui::set_user_name (const string & s)`

the set functions that follow here could be defined within the investor class, however they are only used in the user interface, so no need to load them for the simulation for example

Definition at line 1325 of file [investor_ui.class.cpp](#).

7.16.3.80 `void investor_ui::show_account_form_body ()` [private]

show_account_form shows the account form to be edited

Definition at line 335 of file [investor_ui.class.cpp](#).

7.16.3.81 `void investor_ui::show_disclaimer ()`

show disclaimer

Definition at line 395 of file [investor_ui.class.cpp](#).

7.16.3.82 `void investor_ui::show_follow_up ()`

[follow_up](#)

shows a demo-follow up screen

- eventually with a scenario that the user can fill out: TODO

show a field for the user to change the simulation horizon

show the market evolution

show the dashboard

show detailed feedback per goal

Definition at line 1804 of file [investor_ui.class.cpp](#).

7.16.3.83 void investor_ui::show_goal_help_form_body () [private]

show_goal_help shows the screen that helps to create common goals Retirement

Rainy Day Savings

Insurances

Savings for (grand-) children

Foreseeable expenses

Life Dream

Definition at line 1934 of file [investor_ui.class.cpp](#).

7.16.3.84 void investor_ui::show_home_screen_form ()

show_home_screen_form

Definition at line 191 of file [investor_ui.class.cpp](#).

7.16.3.85 void investor_ui::show_inventory_form ()

Definition at line 981 of file [investor_ui.class.cpp](#).

7.16.3.86 void investor_ui::show_inventory_form_body () [private]

Definition at line 993 of file [investor_ui.class.cpp](#).

7.16.3.87 void investor_ui::show_login_form_body () [private]

Definition at line 527 of file [investor_ui.class.cpp](#).

7.16.3.88 void investor_ui::show_register_form_body () [private]

register_form_body

shows the register form on the screen

Definition at line 270 of file [investor_ui.class.cpp](#).

7.16.3.89 void investor_ui::show_simulation ()

simulation < prepare a series of random returns for each asset class and history for each goal (ie. set goal-Z[g].simulation_string)

show a field for the user to change the simulation horizon

– report the market evolution

now show the feedback for each goal

Definition at line 1731 of file [investor_ui.class.cpp](#).

7.16.3.90 `void investor_ui::show_total_portfolio (investment_problem * oInvProbl)`

`show_total_portfolio`

shows the benchmark of the the portfolio in month 6

Definition at line 2356 of file [investor_ui.class.cpp](#).

7.16.3.91 `void investor_ui::update_last_activity_at ()`

`update_last_activity_at`

updates the information in the database when the user had his/her last last_activity

Definition at line 1537 of file [investor_ui.class.cpp](#).

7.16.4 Member Data Documentation

7.16.4.1 `int investor_ui::active_action = 1`

Definition at line 27 of file [investor_ui.class.cpp](#).

7.16.4.2 `std::map<int, string> investor_ui::asset_class_names [private]`

this variable is to be filled out when needed

Definition at line 118 of file [investor_ui.class.cpp](#).

7.16.4.3 `float* market::assetClass_covar = new float[(NBR_ASSET_CLASSES + 1) * (NBR_ASSET_CLASSES + 1)]`
`[inherited]`

Definition at line 16 of file [market.class.cpp](#).

7.16.4.4 `float market::assetClass_mu[(NBR_ASSET_CLASSES+1)] [inherited]`

note: index 0 not used

Definition at line 17 of file [market.class.cpp](#).

7.16.4.5 `std::map<int, string> market::assetClass_name [inherited]`

note: index 0 not used

Definition at line 18 of file [market.class.cpp](#).

7.16.4.6 `struct tm investor::birth_date [inherited]`

Definition at line 20 of file [investor.class.cpp](#).

7.16.4.7 `std::map<int, std::map<int, float> > market::covar [inherited]`

Definition at line 26 of file [market.class.cpp](#).

7.16.4.8 `string investor::currency` [inherited]

the default currency for that customer

Definition at line 19 of file [investor.class.cpp](#).

7.16.4.9 `std::map<int, int> investor::desirability` [protected], [inherited]

Definition at line 41 of file [investor.class.cpp](#).

7.16.4.10 `std::map<int, float> market::ER` [inherited]

Definition at line 25 of file [market.class.cpp](#).

7.16.4.11 `std::map<int, int> investor::experience` [protected], [inherited]

Definition at line 39 of file [investor.class.cpp](#).

7.16.4.12 `string investor::first_name` [inherited]

Definition at line 16 of file [investor.class.cpp](#).

7.16.4.13 `html_helper investor_ui::hh`

Definition at line 89 of file [investor_ui.class.cpp](#).

7.16.4.14 `int investor::investor_id` [inherited]

Definition at line 14 of file [investor.class.cpp](#).

7.16.4.15 `std::map<int, int> investor::knowledge` [protected], [inherited]

Definition at line 40 of file [investor.class.cpp](#).

7.16.4.16 `string investor::last_name` [inherited]

Definition at line 17 of file [investor.class.cpp](#).

7.16.4.17 `std::map<int, float> investor::max_exposure` [protected], [inherited]

the maximum exposure per asset class

Definition at line 42 of file [investor.class.cpp](#).

7.16.4.18 `string investor::password` [inherited]

Definition at line 18 of file [investor.class.cpp](#).

7.16.4.19 `float investor::simulate_till_age` [inherited]

Definition at line 21 of file [investor.class.cpp](#).

7.16.4.20 string investor::user_name [inherited]

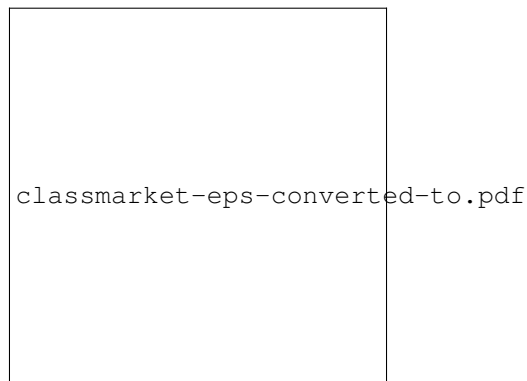
Definition at line 15 of file [investor.class.cpp](#).

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

- [investor_ui.class.cpp](#)

7.17 market Class Reference

Inheritance diagram for market:



Public Member Functions

- [market](#) (int person=0)
- void [load_ER](#) (int person=0, char person_type= 'i')
- void [save_ER](#) (int person=0, char person_type= 'i')
- void [load_covar](#) (int person, char person_type= 'i')
- void [save_covar](#) (int person, char person_type= 'i')

Public Attributes

- float * [assetClass_covar](#) = new float[(NBR_ASSET_CLASSES + 1) * (NBR_ASSET_CLASSES + 1)]
- float [assetClass_mu](#) [(NBR_ASSET_CLASSES+1)]
note: index 0 not used
- std::map< int, string > [assetClass_name](#)
note: index 0 not used
- std::map< int, float > [ER](#)
- std::map< int, std::map< int, float > > [covar](#)

Protected Member Functions

- void [reload_ER](#) (int person, char person_type= 'i')
TODO: the ER and assetClass_mu as well as the covar and assetClass_covar are redundant ==> eliminate the C-style assetClass_.*
- void [reload_covar](#) (int person, char what2do, char person_type= 'i')

7.17.1 Detailed Description

market

this class provides all market information (expecter return, covariances)

(c) Philippe J.S. De Brouwer

last modification: 2015-02

Definition at line 11 of file [market.class.cpp](#).

7.17.2 Constructor & Destructor Documentation

7.17.2.1 market::market (int *person* = 0)

market CONSTRUCTOR

Definition at line 44 of file [market.class.cpp](#).

7.17.3 Member Function Documentation

7.17.3.1 void market::load_covar (int *person*, char *person_type* = ' i ')

load_covar < note: *person_type* = 'i' for persons (later to provide for 'a' advisor, organization, etc.)

< TODO eliminate this

if we did not find a personalized expectation for each asset class, then we load the default values

< TODO eliminate this

Definition at line 135 of file [market.class.cpp](#).

7.17.3.2 void market::load_ER (int *person* = 0, char *person_type* = ' i ')

load_ER < note: *person_type* = 'i' for persons (later to provide for 'a' advisor, organization, etc.)

Definition at line 61 of file [market.class.cpp](#).

7.17.3.3 void market::reload_covar (int *person*, char *what2do*, char *person_type* = ' i ') [protected]

reload_vol

the situation here is more complex as the user expect that only the volatilities will be reset to the default values and not the correlations ...

what2do = [v|c|b] as in [vol | corr | both] < both

< volatilities

< correlations

Definition at line 281 of file [market.class.cpp](#).

7.17.3.4 void market::reload_ER (int *person*, char *person_type* = ' i ') [protected]

TODO: the ER and assetClass_mu as well as the covar and assetClass_covar are redundant ==> eliminate the C-style assetClass_*.

set_mu

the expect value for all asset classes set_covar

the expect value for all assets reload_ER

note: person_type = 'i' for persons (later to provide for 'a' advisor, organization, etc.)

this function simply deletes the personalized expected returns. This is sufficient because later we check if they exists and only then load them (see function [load_ER\(\)](#)).

Definition at line 247 of file [market.class.cpp](#).

7.17.3.5 void market::save_covar (int person, char person_type = ' i ')

save_covar

Definition at line 172 of file [market.class.cpp](#).

7.17.3.6 void market::save_ER (int person = 0, char person_type = ' i ')

save_ER

Definition at line 109 of file [market.class.cpp](#).

7.17.4 Member Data Documentation

7.17.4.1 float* market::assetClass_covar = new float[(NBR_ASSET_CLASSES + 1) * (NBR_ASSET_CLASSES + 1)]

Definition at line 16 of file [market.class.cpp](#).

7.17.4.2 float market::assetClass_mu[(NBR_ASSET_CLASSES+1)]

note: index 0 not used

Definition at line 17 of file [market.class.cpp](#).

7.17.4.3 std::map<int, string> market::assetClass_name

note: index 0 not used

Definition at line 18 of file [market.class.cpp](#).

7.17.4.4 std::map<int, std::map<int, float> > market::covar

Definition at line 26 of file [market.class.cpp](#).

7.17.4.5 std::map<int, float> market::ER

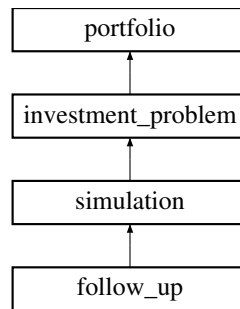
Definition at line 25 of file [market.class.cpp](#).

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

- [market.class.cpp](#)

7.18 portfolio Class Reference

Inheritance diagram for portfolio:



Public Member Functions

- bool [get_portf_from_db](#) (int id)
- string [get_description](#) ()
- void [set_mu](#) (float(*AssetClass_mu))
- void [set_sigma](#) (float(*AssetClass_varCov))

Public Attributes

- int [portfolio_id](#)
- string [description](#)
- float [weights](#) [NBR_ASSET_CLASSES]
the weights of the asset classes (ordered as the asset classes)
- float [pMu](#)
the MONTHLY return
- float [pLogR](#)
the MONTHLY log-return
- float [pSigma](#)
the MONTHLY volatility

7.18.1 Detailed Description

class portfolio

copyright: (c) Philippe De Brouwer 2014

last modification:

Definition at line 11 of file [portfolio.class.cpp](#).

7.18.2 Member Function Documentation

7.18.2.1 string portfolio::get_description ()

returns the label of the portfolio

Definition at line 86 of file [portfolio.class.cpp](#).

7.18.2.2 bool portfolio::get_portf_from_db (int id)

load the definitions from the database note: we have to create a second instance of the sql connection as this is called while the object db is in use!

Definition at line 32 of file [portfolio.class.cpp](#).

7.18.2.3 void portfolio::set_mu (float * *AssetClass_mu*)

sets the expected return for the portfolio

Definition at line 94 of file [portfolio.class.cpp](#).

7.18.2.4 void portfolio::set_sigma (float * *AssetClass_varCov*)

sets the standard deviation for the portfolio: this->sigma

Definition at line 112 of file [portfolio.class.cpp](#).

7.18.3 Member Data Documentation

7.18.3.1 string portfolio::description

Definition at line 15 of file [portfolio.class.cpp](#).

7.18.3.2 float portfolio::pLogR

the MONTHLY log-return

Definition at line 25 of file [portfolio.class.cpp](#).

7.18.3.3 float portfolio::pMu

the MONTHLY return

Definition at line 24 of file [portfolio.class.cpp](#).

7.18.3.4 int portfolio::portfolio_id

Definition at line 14 of file [portfolio.class.cpp](#).

7.18.3.5 float portfolio::pSigma

the MONTHLY volatility

Definition at line 26 of file [portfolio.class.cpp](#).

7.18.3.6 float portfolio::weights[NBR_ASSET_CLASSES]

the weights of the asset classes (ordered as the asset classes)

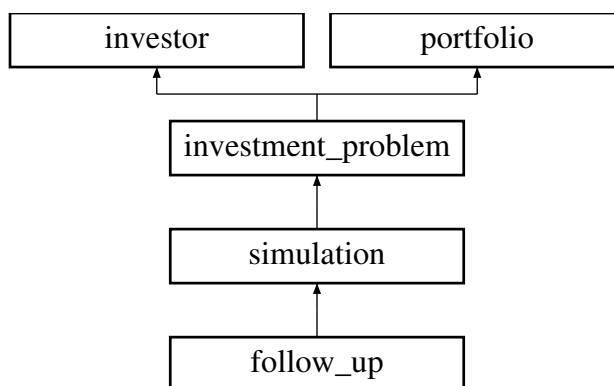
Definition at line 17 of file [portfolio.class.cpp](#).

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

- [portfolio.class.cpp](#)

7.19 simulation Class Reference

Inheritance diagram for simulation:



Public Member Functions

- void [show_simulation](#) ()
- [simulation](#) (int investor)
- void [simulate](#) ()
- int [get_nbrMonths2simulate](#) ()
 - returns the number of months to simulate as in "37"*
- void [plot_market_evolution](#) ()
- void [solve](#) ()
 - allocates means (with a benchmark) to goals*
- int [get_nbr_goals](#) ()
- int [get_max_month_for_lower_goals](#) (int g)
- void [javaGraph](#) (int g, string xtra_var="", string xtra_label="", int followup_mnth=0)
- void [prepare_javaVars_colors](#) (int g, int followup_mnth)
- void [parse_javaGraph](#) (int g, string xtra_var="", string xtra_label="")
 - plots the chartOverview*
- string [get_full_name](#) ()
- float [age](#) (int Mnbr=0)
- int [months2simulate](#) ()
- int [load_from_db](#) (string investorID)
- void [load_ER](#) (int person=0, char person_type='i')
- void [save_ER](#) (int person=0, char person_type='i')
- void [load_covar](#) (int person, char person_type='i')
- void [save_covar](#) (int person, char person_type='i')
- bool [get_portf_from_db](#) (int id)
- string [get_description](#) ()
- void [set_mu](#) (float(*AssetClass_mu))
- void [set_sigma](#) (float(*AssetClass_varCov))

Public Attributes

- std::map< int, [aGOAL](#) > [goalZ](#)
 - map containing all goals*
- int [investor_id](#)
- string [user_name](#)
- string [first_name](#)
- string [last_name](#)
- string [password](#)
- string [currency](#)
 - the default currency for that customer*

- struct tm [birth_date](#)
- float [simulate_till_age](#)
- float * [assetClass_covar](#) = new float[([NBR_ASSET_CLASSES](#) + 1) * ([NBR_ASSET_CLASSES](#) + 1)]
- float [assetClass_mu](#) [([NBR_ASSET_CLASSES](#)+1)]
 - note: index 0 not used*
- std::map< int, string > [assetClass_name](#)
 - note: index 0 not used*
- std::map< int, float > [ER](#)
- std::map< int, std::map< int, float > > [covar](#)
- int [portfolio_id](#)
- string [description](#)
- float [weights](#) [[NBR_ASSET_CLASSES](#)]
 - the weights of the asset classes (ordered as the asset classes)*
- float [pMu](#)
 - the MONTHLY return*
- float [pLogR](#)
 - the MONTHLY log-return*
- float [pSigma](#)
 - the MONTHLY volatility*

Protected Member Functions

- void [set_followup_color_prae](#) (float Vlow, float Vmed, float Vhigh, int g)
- void [set_followup_color_post](#) (float V, int g)
- int [age2monthNbr](#) (float theAge)
- string [dateStr2Age](#) (string the_date)
- bool [save](#) ()
- int [get_from_db](#) (string investorID, string [password](#))
- bool [add_to_db](#) ()
- bool [exists](#) (int iid)
- void [load_preferences](#) ()
 - sets the experience, knowledge and desirability vectors*
- bool [save_preferences](#) ()
 - sets the experience, knowledge and desirability vectors*
- void [set_max_exposure](#) ()
 - sets the max_exposure vector based on the experience, knowledge and desirability*
- bool [set_scale](#) (string scale_type, int ac, int val)
- void [reload_ER](#) (int person, char person_type= 'i')
 - TODO: the ER and assetClass_mu as well as the covar and assetClass_covar are redundant ==> eliminate the C-style assetClass_*.*
- void [reload_covar](#) (int person, char what2do, char person_type= 'i')

Protected Attributes

- int [nbrMonths](#)
 - the number of months to simulate, per goal we simulate max(realization_monthNbr, nbrMonths)*
- std::map< int, std::map< int, float > > [market_return](#)
- std::map< int, [portfolio](#) > [portfolios](#)
 - the standard portfolios*
- std::map< int, int > [experience](#)

- `std::map< int, int >` [knowledge](#)
- `std::map< int, int >` [desirability](#)
- `std::map< int, float >` [max_exposure](#)
the maximum exposure per asset class

Private Member Functions

- void [set_nbrMonths2simulate_from_env](#) ()
- void [simulate_market](#) ()
sets nbrMonths
- void [simulate_portfolios](#) ()

7.19.1 Detailed Description

simulation

class used to simulate behaviour of the investment problem after optimization

(c) Philippe J.S. De Brouwer

last modification: 2014-08

Definition at line 12 of file [simulation.class.cpp](#).

7.19.2 Constructor & Destructor Documentation

7.19.2.1 `simulation::simulation (int investor)`

CONSTRUCTOR of the class simulation

Definition at line 37 of file [simulation.class.cpp](#).

7.19.3 Member Function Documentation

7.19.3.1 `bool investor::add_to_db ()` `[protected]`, `[inherited]`

`add_to_db`

does not check data any more, simply tries to push it to the DB only returns 0 if the userID already exists.

Definition at line 313 of file [investor.class.cpp](#).

7.19.3.2 `float investor::age (int Mnbr = 0)` `[inline]`, `[inherited]`

returns the age of a customer in years (with decimal part)

Definition at line 108 of file [investor.class.cpp](#).

7.19.3.3 `int investor::age2monthNbr (float theAge)` `[inline]`, `[protected]`, `[inherited]`

`age2monthNbr`

returns the month number for a given age note that this is negative for ages younger than the actual age.

Definition at line 200 of file [investor.class.cpp](#).

7.19.3.4 `string investor::dateStr2Age (string the_date)` `[inline]`, `[protected]`, `[inherited]`

`dateStr2Age`

Definition at line 209 of file [investor.class.cpp](#).

7.19.3.5 `bool investor::exists (int iid)` `[protected]`, `[inherited]`

`exists`

checks if a given investor-id really exists

Definition at line 343 of file [investor.class.cpp](#).

7.19.3.6 `string portfolio::get_description ()` `[inherited]`

returns the label of the portfolio

Definition at line 86 of file [portfolio.class.cpp](#).

7.19.3.7 `int investor::get_from_db (string uid, string password)` `[protected]`, `[inherited]`

`get_from_db` returns the `investor_id` if successful (otherwise 0)

Definition at line 249 of file [investor.class.cpp](#).

7.19.3.8 `string investor::get_full_name ()` `[inherited]`

overloading << operator overload the << operator to output the goal in a readable format `get_full_name` returns the full name of an investor

Definition at line 103 of file [investor.class.cpp](#).

7.19.3.9 `int investment_problem::get_max_month_for_lower_goals (int g)` `[inherited]`

returns the maximum term of all the other LOWER ranking goals. The higher ranking goals already have allocated all the means that they require, so we can disregard them here. `goal_type = 0` is the unallocated goal type, so this goal should not be taken into account here.

Definition at line 945 of file [investment_problem.class.cpp](#).

7.19.3.10 `int investment_problem::get_nbr_goals ()` `[inherited]`

returns the number of goals

Definition at line 936 of file [investment_problem.class.cpp](#).

7.19.3.11 `int simulation::get_nbrMonths2simulate ()`

returns the number of months to simulate as in "37"

[get_nbrMonths2simulate\(\)](#)

returns `nbrMonths` (to simulate)

Definition at line 259 of file [simulation.class.cpp](#).

7.19.3.12 `bool portfolio::get_portf_from_db (int id) [inherited]`

load the definitions from the database note: we have to create a second instance of the sql connection as this is called while the object db is in use!

Definition at line 32 of file [portfolio.class.cpp](#).

7.19.3.13 `void investment_problem::javaGraph (int g, string xtra_var = " ", string xtra_label = " ", int followup_mnth = 0) [inherited]`

javaGraph

plots the overview-graph for the given goal

x_{tav}_Var and x_{tra_label} add a series of data that is delivered in text format x_{tra_var} = [45,2000], [45.5,2200] x_{tra_label} = "history" followup_mnth = 0 for feedback and simulation screen, and for the follow up screen it is the number of months we look into the future.

Definition at line 1089 of file [investment_problem.class.cpp](#).

7.19.3.14 `void market::load_covar (int person, char person_type = 'i') [inherited]`

load_covar < note: person_type = 'i' for persons (later to provide for 'a' advisor, organization, etc.)

< TODO eliminate this

if we did not find a personalized expectation for each asset class, then we load the default values

< TODO eliminate this

Definition at line 135 of file [market.class.cpp](#).

7.19.3.15 `void market::load_ER (int person = 0, char person_type = 'i') [inherited]`

load_ER < note: person_type = 'i' for persons (later to provide for 'a' advisor, organization, etc.)

Definition at line 61 of file [market.class.cpp](#).

7.19.3.16 `int investor::load_from_db (string uid) [inherited]`

returns the investor_id if successful (otherwise 0) NOTE for user interaction use get_from_db(uid, password) for password verification!!

Definition at line 134 of file [investor.class.cpp](#).

7.19.3.17 `void investor::load_preferences () [protected],[inherited]`

sets the experience, knowledge and desirability vectors

set_preferences

Definition at line 383 of file [investor.class.cpp](#).

7.19.3.18 `int investor::months2simulate () [inline],[inherited]`

months2simulate

Definition at line 124 of file [investor.class.cpp](#).

7.19.3.19 `void investment_problem::parse_javaGraph (int g, string xtra_var = "", string xtra_label = "")`
 [inherited]

plots the chartOverview

parse_javaGraph

parses the line plot of the three scenarios and the goal to the output (webbrowser). It parses as well the that will get become the plot as the <script> with javascript that will produce the plot.

Definition at line 1103 of file [investment_problem.class.cpp](#).

7.19.3.20 `void simulation::plot_market_evol ()`

plot_market_evol

plots the evolution of the different asset classes (is goal independent) initialize:

show the table if desired

build the simulation variables:

cout the script

Definition at line 138 of file [simulation.class.cpp](#).

7.19.3.21 `void investment_problem::prepare_javaVars_colors (int g, int followup_mnth)` [inherited]

prepare_javaVars_colors

followup_mnth is used as the startmonth for the followUp-Screen (in all other graphs it will be zero – its default)

exp = expected value med = median low = alpha quantile high = 1- alpha quantile < the weighted average of the number of months that investments are in portfolio

TODO in the function below use another rate for NEGATIVE amounts, to mymic lending!

parse_javaGraph(g, s1, s2, s3, s4, xtra_var, xtra_label);

< place al the variables

Definition at line 970 of file [investment_problem.class.cpp](#).

7.19.3.22 `void market::reload_covar (int person, char what2do, char person_type = ' i')` [protected],
 [inherited]

reload_vol

the situation here is more complex as the user expect that only the volatilities will be reset to the default values and not the correlations ...

what2do = [v|c|b] as in [vol | corr | both] < both

< volatilities

< correlations

Definition at line 281 of file [market.class.cpp](#).

7.19.3.23 `void market::reload_ER (int person, char person_type = ' i')` [protected], [inherited]

TODO: the ER and assetClass_mu as well as the covar and assetClass_covar are redundant ==> eliminate the C-style assetClass_*.

set_mu

the expect value for all asset classes `set_covar`

the expect value for all assets `reload_ER`

note: `person_type = 'i'` for persons (later to provide for 'a' advisor, organization, etc.)

this function simply deletes the personalized expected returns. This is sufficient because later we check if they exists and only then load them (see function [load_ER\(\)](#)).

Definition at line 247 of file [market.class.cpp](#).

7.19.3.24 `bool investor::save ()` `[protected]`, `[inherited]`

`save`

Definition at line 220 of file [investor.class.cpp](#).

7.19.3.25 `void market::save_covar (int person, char person_type = ' i ')` `[inherited]`

`save_covar`

Definition at line 172 of file [market.class.cpp](#).

7.19.3.26 `void market::save_ER (int person = 0, char person_type = ' i ')` `[inherited]`

`save_ER`

Definition at line 109 of file [market.class.cpp](#).

7.19.3.27 `bool investor::save_preferences ()` `[protected]`, `[inherited]`

sets the experience, knowledge and desirability vectors

[save_preferences\(\)](#)

Definition at line 416 of file [investor.class.cpp](#).

7.19.3.28 `void investment_problem::set_followup_color_post (float V, int g)` `[inline]`, `[protected]`, `[inherited]`

`set_followup_color_post` sets the color of the goal in case the simulation is LONGER than the realization date of the goal

Definition at line 1246 of file [investment_problem.class.cpp](#).

7.19.3.29 `void investment_problem::set_followup_color_prae (float Vlow, float Vmed, float Vhigh, int g)` `[inline]`, `[protected]`, `[inherited]`

`set_followup_color_prae`

sets the color of the goal in case the simulations ends PRIOR to the realization date of the goal

Definition at line 1234 of file [investment_problem.class.cpp](#).

7.19.3.30 `void investor::set_max_exposure ()` `[protected]`, `[inherited]`

sets the `max_exposure` vector based on the experience, knowledge and desirability

`get_max_exposure`

sets the map `max_exposure` in this object

Definition at line 363 of file [investor.class.cpp](#).

7.19.3.31 `void portfolio::set_mu (float * AssetClass_mu)` [inherited]

sets the expected return for the portfolio

Definition at line 94 of file [portfolio.class.cpp](#).

7.19.3.32 `void simulation::set_nbrMonths2simulate_from_env ()` [private]

`get_nbrMonths2simulate_from_env`

returns nothing sets this->nbrMonths if nMonths is supplied, then test and use it, otherwise use defaultValue

Definition at line 223 of file [simulation.class.cpp](#).

7.19.3.33 `bool investor::set_scale (string scale_type, int ac, int val)` [protected],[inherited]

`set_scale`

Definition at line 402 of file [investor.class.cpp](#).

7.19.3.34 `void portfolio::set_sigma (float * AssetClass_varCov)` [inherited]

sets the standard deviation for the portfolio: this->sigma

Definition at line 112 of file [portfolio.class.cpp](#).

7.19.3.35 `void simulation::show_simulation ()`

7.19.3.36 `void simulation::simulate ()`

`simulate`

simulates one possible scenario for what can happen after the goals are set TODO in absense of gGaoI.load() we solve again ...!!!!!!!

< does this for each goal and sets goalZ[g].simulation_string (requires [solve\(\)](#))

Definition at line 49 of file [simulation.class.cpp](#).

7.19.3.37 `void simulation::simulate_market ()` [private]

sets nbrMonths

`simulate_market`

simulates one possible scenario for the market evolution

Definition at line 62 of file [simulation.class.cpp](#).

7.19.3.38 `void simulation::simulate_portfolios ()` [private]

`simulate_portfolios`

simulates the evolutions of the portfolios given a certain market scenario

Definition at line 103 of file [simulation.class.cpp](#).

7.19.3.39 `void investment_problem::solve ()` [inherited]

allocates means (with a benchmark) to goals

solve is the main attribute to be used by other classes a [cls_currency](#) object that will facilitate currency conversions

note: `set_assets`, `set_cfs`, `set_goals`, `set_mu` are done in the constructor

< initialize means_left for g=1 (all in [investor.currency](#))

< initialize means_left for g=1 (all in [investor.currency](#))

< do nothing right now for unallocated goals, so take the next goal (for loop)

try the free means (not used by other –lower ranking– goals)

< initialize

and means that are allocated to this goal):

for the asap-goals we take the first cash flows first

for all other goal types we take the last possible

if the free means are not sufficient, try all the previous also

block the resources for this goal

set the benchmark

set comments and colors:

add the risk info

put all the remaining means into the unallocated goal

Definition at line 396 of file [investment_problem.class.cpp](#).

7.19.4 Member Data Documentation

7.19.4.1 `float* market::assetClass_covar = new float[(NBR_ASSET_CLASSES + 1) * (NBR_ASSET_CLASSES + 1)]`
[inherited]

Definition at line 16 of file [market.class.cpp](#).

7.19.4.2 `float market::assetClass_mu[(NBR_ASSET_CLASSES+1)]` [inherited]

note: index 0 not used

Definition at line 17 of file [market.class.cpp](#).

7.19.4.3 `std::map<int, string> market::assetClass_name` [inherited]

note: index 0 not used

Definition at line 18 of file [market.class.cpp](#).

7.19.4.4 `struct tm investor::birth_date` [inherited]

Definition at line 20 of file [investor.class.cpp](#).

7.19.4.5 `std::map<int, std::map<int, float> > market::covar` [inherited]

Definition at line 26 of file [market.class.cpp](#).

7.19.4.6 `string investor::currency` [inherited]

the default currency for that customer

Definition at line 19 of file [investor.class.cpp](#).

7.19.4.7 `string portfolio::description` [inherited]

Definition at line 15 of file [portfolio.class.cpp](#).

7.19.4.8 `std::map<int, int> investor::desirability` [protected],[inherited]

Definition at line 41 of file [investor.class.cpp](#).

7.19.4.9 `std::map<int, float> market::ER` [inherited]

Definition at line 25 of file [market.class.cpp](#).

7.19.4.10 `std::map<int, int> investor::experience` [protected],[inherited]

Definition at line 39 of file [investor.class.cpp](#).

7.19.4.11 `string investor::first_name` [inherited]

Definition at line 16 of file [investor.class.cpp](#).

7.19.4.12 `std::map<int, aGOAL> investment_problem::goalZ` [inherited]

map containing all goals

Definition at line 22 of file [investment_problem.class.cpp](#).

7.19.4.13 `int investor::investor_id` [inherited]

Definition at line 14 of file [investor.class.cpp](#).

7.19.4.14 `std::map<int, int> investor::knowledge` [protected],[inherited]

Definition at line 40 of file [investor.class.cpp](#).

7.19.4.15 `string investor::last_name` [inherited]

Definition at line 17 of file [investor.class.cpp](#).

7.19.4.16 `std::map<int, std::map<int, float>> simulation::market_return` [protected]

Definition at line 25 of file [simulation.class.cpp](#).

7.19.4.17 `std::map<int, float> investor::max_exposure` [protected],[inherited]

the maximum exposure per asset class

Definition at line 42 of file [investor.class.cpp](#).

7.19.4.18 `int simulation::nbrMonths` [protected]

the number of months to simulate, per goal we simulate $\max(\text{realization_monthNbr}, \text{nbrMonths})$

Definition at line 22 of file [simulation.class.cpp](#).

7.19.4.19 `string investor::password` [inherited]

Definition at line 18 of file [investor.class.cpp](#).

7.19.4.20 `float portfolio::pLogR` [inherited]

the MONTHLY log-return

Definition at line 25 of file [portfolio.class.cpp](#).

7.19.4.21 `float portfolio::pMu` [inherited]

the MONTHLY return

Definition at line 24 of file [portfolio.class.cpp](#).

7.19.4.22 `int portfolio::portfolio_id` [inherited]

Definition at line 14 of file [portfolio.class.cpp](#).

7.19.4.23 `std::map<int, portfolio> investment_problem::portfolios` [protected],[inherited]

the standard portfolios

Definition at line 29 of file [investment_problem.class.cpp](#).

7.19.4.24 `float portfolio::pSigma` [inherited]

the MONTHLY volatility

Definition at line 26 of file [portfolio.class.cpp](#).

7.19.4.25 `float investor::simulate_till_age` [inherited]

Definition at line 21 of file [investor.class.cpp](#).

7.19.4.26 `string investor::user_name` [inherited]

Definition at line 15 of file [investor.class.cpp](#).

7.19.4.27 float portfolio::weights[NBR_ASSET_CLASSES] [inherited]

the weights of the asset classes (ordered as the asset classes)

Definition at line 17 of file [portfolio.class.cpp](#).

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

- [simulation.class.cpp](#)

Chapter 8

File Documentation

8.1 aGOAL.struct.cpp File Reference

Classes

- struct [aGOAL](#)

Functions

- ostream & [operator<<](#) (ostream &os, const [aGOAL](#) &the_goal)

8.1.1 Function Documentation

8.1.1.1 ostream& operator<< (ostream & os, const aGOAL & the_goal)

overload the << operator to output the goal in a readable format

Definition at line 80 of file [aGOAL.struct.cpp](#).

8.2 aGOAL.struct.cpp

```
00001
00005 struct aGOAL
00006 {
00007     public:
00008         aGOAL(int investor_id = 0) {this->investor = investor_id;}
00009
00010         int    goal_id;
00011         int    investor = 0;
00012         int    goal_type;
00013         float  amount;
00014         float  target_amount;
00015         std::map<int, float> means_goal;
00016
00017         string description;
00018         string currency;
00019         int    priority;
00020         string from_date;
00021         string till_date;
00022         string frequency;
00023         float  realization_age;
00024         float  max_shortfall;
00025
00026         float  alpha = 0.1;
00027         int    realization_monthNbr;
00028         std::map<int, float> saving_plan;
00029         int    optimal_portf;
00030         float  realized_shortfall;
```

```

00035     std::map<int, float> benchmark;
00036     std::string remarks;
00037     string color;
00038     string color_followup;
00039     bool success;
00040     string simulation_string = "";
00041     float simulation_endvalue = 0;
00042     string s_low = "";
00043     string s_exp = "";
00044     string s_high = "";
00045     string s_goal = "";
00046
00047
00048     friend ostream& operator<<(ostream& os, const aGOAL& the_goal);
00049
00050
00051 /* void save_this(string g_type, string descr, string curr, string priority,
00052                  string f_date, string t_date, string f, string r_age,
00053                  string amount, string m_shortfall);
00054 */
00055
00056 //     bool set_unallocated_goal(); --> moved to investment_problem
00057 void set_from_db();
00058 void set_to_pstmnt();
00059     bool save();
00060 bool add_to_db();
00061 void save_results();
00062
00063     bool set_goal_type      (string g_type);
00064     bool set_amount        (string amnt);
00065     bool set_description   (string desc);
00066     bool set_currency      (string curr);
00067     bool set_priority      (string imp);
00068     bool set_from_date     (string f_date);
00069     bool set_till_date     (string t_date);
00070     bool set_frequency     (string freq);
00071     bool set_realization_age (string real_age, float actual_age);
00072     bool set_max_shortfall (string shortfall);
00073     int  show_graph_till_MNbr ();
00074 };
00075
00076
00077 ostream& operator<<(ostream& os, const aGOAL& the_goal)
00078 {
00079     if (the_goal.goal_type == goal_type_s2i["unallocated"])
00080     {
00081         os << "The goal <b><i>&quot;" << the_goal.description << "&quot;</i></b> is to show you the
00082         resources that are not used by your other goals.";
00083     }
00084     else if (the_goal.goal_type == goal_type_s2i["income from/to"])
00085     {
00086         cls_frequency the_freq;
00087         the_freq.set_freq(the_goal.frequency);
00088         string s = the_freq.get_freq_label();
00089         std::transform(s.begin(), s.end(), s.begin(), ::tolower);
00090
00091         os << setiosflags(ios::fixed) << setprecision(2)
00092         << "The goal <b><i>&quot;" << the_goal.description << "&quot;</i></b> is to achieve"
00093         << " an income of " << curr_format(the_goal.amount, the_goal.
00094         currency)
00095         << " (" << s << ") "
00096         << " from " << date_format_dateStr(the_goal.from_date)
00097         << " to " << date_format_dateStr(the_goal.till_date) << ".<br>"
00098         << "This goal's relative priority is " << the_goal.priority;
00099     }
00100     else if (the_goal.goal_type == goal_type_s2i["rainy day savings"])
00101     {
00102         os << setiosflags(ios::fixed) << setprecision(2)
00103         << "This goal <b><i>&quot;" << the_goal.description << "&quot;</i></b> is to have a
00104         safety layer (\\"rainy day savings\\") always available."
00105         << "The amount that we try to keep available is " << curr_format(the_goal.
00106         amount, the_goal.currency)
00107         << ".<br>This goal's relative priority is " << the_goal.priority;
00108     }
00109     else if (the_goal.goal_type == goal_type_s2i["amount asap"])
00110     {
00111         os << setiosflags(ios::fixed) << setprecision(2)
00112         << "This goal <b><i>&quot;" << the_goal.description << "&quot;</i></b> is to obtain the
00113         sum of " << curr_format(the_goal.amount, the_goal.currency)
00114         << " asap.<br>This goal's relative priority is " << the_goal.priority;
00115     }
00116     else
00117     {
00118         os << setiosflags(ios::fixed) << setprecision(2)
00119         << "The goal <b><i>&quot;" << the_goal.description << "&quot;</i></b> is to achieve"
00120         << " an amount of " << curr_format(the_goal.amount, the_goal.
00121         currency)
00122         << " at the age of " << setprecision(0) << the_goal.realization_age << ".<br>" <<

```

```

        setprecision(2)
00120     << "This goal's relative priority is " << the_goal.priority
00121     << " and we accept a maximum shortfall of " << curr_format(the_goal.
max_shortfall, the_goal.currency) << ".";
00122     }
00123     return os;
00124 }
00125
00126 bool aGOAL::set_goal_type(string g_type)
00127 {
00128     int g_type_id = atoi(g_type.c_str());
00129     unordered_map<int, string>::const_iterator got = goal_type_i2s.find(g_type_id);
00130     if (got == goal_type_i2s.end())
00131     {
00132         error_message = error_message + "<li>the goal_type is not one of the
supported types</li>";
00133         return false;
00134     }
00135     else
00136     {
00137         this->goal_type = g_type_id;
00138         return true;
00139     }
00140 }
00141
00142 bool aGOAL::set_amount(string amnt)
00143 {
00144     this->amount = atof(amnt.c_str()); // returns 0.0 if no valid number is found
00145     return true;
00146 }
00147
00148 bool aGOAL::set_currency(string curr)
00149 {
00150     cls_currency o_curr;
00151     if (o_curr.set_curr(curr))
00152     {
00153         this->currency = curr;
00154         return true;
00155     }
00156     else
00157     {
00158         error_message = error_message + "<li>error occured saving the currency ---
not a valid currency</li>";
00159         return false;
00160     }
00161 }
00162
00163 bool aGOAL::set_description(string desc)
00164 {
00165     if (desc == "")
00166     {
00167         this->description = "Please change this text with your own goal-description!";
00168     }
00169     else
00170     {
00171         this->description = desc;
00172     }
00173     return true;
00174 }
00175
00176
00177 bool aGOAL::set_priority(string imp)
00178 {
00179     this->priority = atoi(imp.c_str());
00180     return true;
00181 }
00182
00189 bool aGOAL::set_realization_age(string real_age, float actual_age)
00190 {
00191     float the_age;
00192     the_age = atof(real_age.c_str());
00193     if ((the_age < actual_age) || (the_age > (MAX_AGE + 0.0)))
00194     {
00195         error_message = error_message + "<li>the realization_age of the goal should
be a number between your actual age and " + to_string(MAX_AGE + 0) + "(provided: realization age = " +
real_age + "; compared to actual age = " + to_string(actual_age) + "</li>";
00196         return false;
00197     }
00198     else
00199     {
00200         this->realization_age = the_age;
00201         //NOT this->from_date = tm2DateStr(get_tm((the_age - actual_age) * 12));
00202         //NOT this->till_date = this->from_date;
00203         //NOT override! this->frequency = "0";
00204         return true;
00205     }
00206 }

```

```

00207
00208 bool aGOAL::set_frequency(string freq)
00209 {
00210     cls_frequency o_freq;
00211     if (o_freq.set_freq(freq))
00212     {
00213         this->frequency = freq;
00214         return true;
00215     }
00216     else
00217     {
00218         error_message = error_message + "<li>error saving the frequency
information</li>";
00219         return false;
00220     }
00221 }
00222
00223 bool aGOAL::set_from_date(string f_date)
00224 {
00225     if (is_valid_dateStr(f_date))
00226     {
00227         from_date = f_date;
00228         return true;
00229     }
00230     else
00231     {
00232         error_message = error_message + "<li>from_date is not a valid date</li>";
00233         return false;
00234     }
00235 }
00236
00242 bool aGOAL::set_till_date(string t_date)
00243 {
00244     struct tm tm_from_date, tm_till_date;
00245     dateStr2tm(this->from_date, &tm_from_date);
00246     dateStr2tm(t_date, &tm_till_date);
00247     if (is_valid_dateStr(t_date))
00248     {
00249         if (difftime(mktime(&tm_till_date),mktime(&tm_from_date)) >= 0)
00250         {
00251             till_date = t_date;
00252             return true;
00253         }
00254         else
00255         {
00256             error_message = error_message + "<li>the till_date must be after the
from_date (it cannot precede the from_date)</li>";
00257             return false;
00258         }
00259     }
00260     else
00261     {
00262         error_message = error_message + "<li>till_date is not a valid date</li>";
00263         return false;
00264     }
00265 }
00266 }
00267
00268 bool aGOAL::set_max_shortfall(string shortfall)
00269 {
00270     float the_sf = atof(shortfall.c_str());
00271     if ((this->goal_type != goal_type_s2i["income from/to"]) && (the_sf > this->
amount))
00272     {
00273         std_message = std_message + "<li>are you sure that the shortfall can be higher
than the target amount (note: this is saved, but you might want to reconsider)</li>";
00274     }
00275     this->max_shortfall = the_sf;
00276     return true;
00277 }
00278
00279
00280 bool aGOAL::add_to_db()
00281 {
00282     if (this->investor == 0)
00283     {
00284         error_message = error_message + "<li>Failed to add_to_db, because investor id
not set</li>.";
00285         return false;
00286     }
00287     string s;
00288     s = "INSERT INTO " + oConfig.tbl_prefix + "_goals (investor, goal_type, description,
need_level, realization_age, from_date, ";
00289     s = s + "till_date, frequency, amount, currency, max_shortfall, priority) VALUES
(?, ?, ?, ?, ?, ?, ?, ?, ?, ?)";
00290
00291     try

```

```

00292 {
00293     db.pstmt = db.con->prepareStatement(s);
00294     db.pstmt->setInt ( 1,investor);
00295     db.pstmt->setInt ( 2,this->goal_type);
00296     db.pstmt->setString( 3,this->description);
00297     db.pstmt->setInt ( 4,1);
00298     db.pstmt->setDouble( 5,this->realization_age);
00299     db.pstmt->setString( 6,this->from_date);
00300     db.pstmt->setString( 7,this->till_date);
00301     db.pstmt->setString( 8,this->frequency);
00302     db.pstmt->setDouble( 9,this->amount);
00303     db.pstmt->setString(10,this->currency);
00304     db.pstmt->setDouble(11,this->max_shortfall);
00305     db.pstmt->setInt (12,this->priority);
00306     db.pstmt->execute();
00307 }
00308
00309 catch (sql::SQLException &e)
00310 {
00311     #ifdef DEBUG
00312         debug_info = debug_info + "# ERR: SQLException in " + __FILE__ + "(" + __FUNCTION__
00313             + ") on line " + to_string(__LINE__) + "<br># ERR: " + e.what()
00314             + " (MySQL error code: " + to_string(e.getErrorCode())
00315             + ", SQLState: " + e.getSQLState() + ")<br>";
00316     #endif
00317     error_message = error_message + "<li>Sorry, I failed to add the goal
<b>&quot;" + this->description + "&quot;</b> to the database. Please, retry!</li>";
00318     return false;
00319 }
00320 return true; // if we're still here, then no exception was thrown and the INSERT INTO was successful.
00321 }
00322
00327 bool aGOAL::save()
00328 {
00329     string s;
00330     s = "UPDATE " + oConfig.tbl_prefix + "_goals SET , goal_type = ?, description = ?,
need_level = ?, realization_age = ?, from_date = ?, ";
00331     s = s + "till_date = ?, frequency = ?, amount = ?, currency = ?, max_shortfall = ?, priority = ?";
00332     s = s + " WHERE investor_id = " + to_string(this->investor) + ";";
00333
00334     try
00335     {
00336         db.pstmt = db.con->prepareStatement(s);
00337         set_to_pstmt();
00338     }
00339
00340     catch (sql::SQLException &e)
00341     {
00342         #ifdef DEBUG
00343             debug_info = debug_info + "# ERR: SQLException in " + __FILE__ + "(" + __FUNCTION__
00344                 + ") on line " + to_string(__LINE__) + "<br># ERR: " + e.what()
00345                 + " (MySQL error code: " + to_string(e.getErrorCode())
00346                 + ", SQLState: " + e.getSQLState() + ")<br>";
00347         #endif
00348         error_message = error_message + "<li>Sorry, I failed to save the goal
<b>&quot;" + this->description + "&quot;</b> the remains unchanged!</li>";
00349         return false;
00350     }
00351     return true; // if we're still here, then no exception was thrown and the INSERT INTO was successful.
00352 }
00353 }
00354
00355
00356
00362 void aGOAL::set_from_db()
00363 {
00364     this->goal_id = db.res->getInt("goal_id");
00365     this->investor = db.res->getInt("investor");
00366     this->amount = db.res->getDouble("amount");
00367     this->goal_type = db.res->getInt("goal_type");
00368     this->description = db.res->getString("description");
00369     this->realization_age = atof(db.res->getString("realization_age").c_str());
00370     this->currency = db.res->getString("currency");
00371     this->priority = db.res->getInt("priority");
00372     this->from_date = db.res->getString("from_date");
00373     this->till_date = db.res->getString("till_date");
00374     this->frequency = db.res->getString("frequency");
00375     this->max_shortfall = atof(db.res->getString("max_shortfall").c_str());
00376     // not loaded:
00377     // need_level INT(10) REFERENCES " + oConfig.tbl_prefix + "_need_levels(need_level_id) ON DELETE
RESTRICT,
00378
00379 }
00380
00381 void aGOAL::set_to_pstmt()
00382 {
00383     db.pstmt->setInt ( 1,investor);

```

```

00384     db.pstmt->setInt    ( 2,this->goal_type);
00385     db.pstmt->setString( 3,this->description);
00386     db.pstmt->setInt    ( 4,1); // need level not used right now
00387     db.pstmt->setDouble( 5,this->realization_age);
00388     db.pstmt->setString( 6,this->from_date);
00389     db.pstmt->setString( 7,this->till_date);
00390     db.pstmt->setString( 8,this->frequency);
00391     db.pstmt->setDouble( 9,this->amount);
00392     db.pstmt->setString(10,this->currency);
00393     db.pstmt->setDouble(11,this->max_shortfall);
00394     db.pstmt->setInt    (12,this->priority);
00395     db.pstmt->execute();
00396 }
00397

```

8.3 asset.class.cpp File Reference

Classes

- class `asset`

8.4 asset.class.cpp

```

00001 /*
00002  *
00003  * class asset
00004  *
00005  * copyright: (c) Philippe De Brouwer 2014
00006  *
00007  * last modification:
00008  *
00009  */
00010
00011 class asset
00012 {
00013 public:
00014     int     asset_id;
00015     int     investor;
00016     int     asset_class;
00017     string  description;
00018     string  currency;
00019     float   amount;
00020
00021     bool set_asset_id   (const std::string& s);
00022     bool set_investor   (const std::string& s);
00023     bool set_asset_class(const std::string& s);
00024     bool set_description(const std::string& s);
00025     bool set_currency   (const std::string& s);
00026     bool set_amount     (const std::string& s);
00027     void list_assets    (int iid);
00028     void show_edit_form(int iid);
00029     bool load();
00030     bool save();
00031     bool asset_delete ();
00032
00033 private:
00034     bool exists (int aid);
00035 };
00036
00037
00038 /*****
00039  *         load
00040  *
00041  * *****/
00042 bool asset::load()
00043 {
00044     bool rc = false;
00045     string s = "";
00046     s = "SELECT * FROM " + oConfig.tbl_prefix + "_assets WHERE asset_id = " + std::to_string
(asset_id) + ";";
00047     db.res = db.stmt->executeQuery(s);
00048     while (db.res->next())
00049     {
00050         if (set_investor   (db.res->getString("investor"))   &&
00051             set_asset_class(db.res->getString("asset_class")) &&
00052             set_description(db.res->getString("description")) &&
00053             set_currency   (db.res->getString("currency"))   &&

```

```

00054         set_amount      (db.res->getString("amount")) rc = true;
00055     }
00056     return rc;
00057 }
00058
00059 /*****
00060 *   show_edit
00061 *
00062 *   *****/
00063 void asset::show_edit_form(int iid)
00064 {
00065
00066     cout << "<form method=\"post\" action=\"" + oConfig.soft_url + "?a=as\">";
00067     cout << "<table class=\"asset\">\n";
00068     cout << "<tr>";
00069     cout << "<th>Description</th><th>Asset Class</th><th>Amount</th><th>Currency</th><th>Actions</th>";
00070     /* cout << "</tr>\n";
00071     oAsset.list_assets(investor_id);
00072     */
00073     cout << "</tr>\n<tr>\n";
00074     cout << "<td><input type=\"text\" name=\"description\" required value=\"" <<
description << "\"></td>" ;
00075     cout << "<td>" << oAssetClass.dropDown_assetClass(
asset_class << "</td>" ;
00076     cout << "<td><input type=\"number\" min=\"0\" name=\"amount\" required title=\"Please enter a positive
number\" value=\"" << amount << "\"></td>" ;
00077     cout << "<td>" << oCurrency.dropDown_currency(
currency) << "</td>" ;
00078
00079     switch (oConfig.layout)
00080     {
00081     case '1' :
00082         cout << "<td><button type=\"submit\" value=\"Submit\"><img src=\"" <<
oConfig.css_dir << "check.svg\" alt=\"add\" width=\"16\" height=\"16\" alt=\"Save\"
></button><a href=\"" + oConfig.soft_url + "?a=if&iid=" << iid << "\" class=\"button\"><img src=\""
<< oConfig.css_dir << "arrow_left.svg\" alt=\"Go back without saving the changes!\" width=\"
16\" height=\"16\"></a></td>" ;
00083         break;
00084     case '2' :
00085         cout << "<td><button type=\"submit\" value=\"Submit\" data-toggle=\"tooltip\" title=\"" <<
t_save << "\"><span class=\"glyphicon glyphicon-ok\" style=\"color:green\"></span></button>";
00086         cout << "<a data-toggle=\"tooltip\" title=\"" << t_cancel << "\" href=\"" +
oConfig.soft_url + "?a=if&iid=" << iid << "\"><span class=\"glyphicon
glyphicon-chevron-left\" style=\"color:blue\"></span></a></td>" ;
00087         break;
00088     default :
00089         cout << "<td><button type=\"submit\" value=\"Submit\"><img src=\"" <<
oConfig.css_dir << "check.svg\" alt=\"add\" width=\"16\" height=\"16\" alt=\"Save\"
></button><a href=\"" + oConfig.soft_url + "?a=if&iid=" << iid << "\" class=\"button\"><img src=\""
<< oConfig.css_dir << "arrow_left.svg\" alt=\"Go back without saving the changes!\" width=\"
16\" height=\"16\"></a></td>" ;
00090         break;
00091     }
00092
00093     cout << "</tr></table>\n";
00094     cout << "<input type=\"hidden\" name=\"iid\" value=\"" << iid << "\" />\n";
00095     cout << "<input type=\"hidden\" name=\"aid\" value=\"" << asset_id << "\" />\n";
00096     cout << "<p>All fields are required.</p>";
00097     cout << "</form>";
00098 }
00099
00100 /*****
00101 *   void    list_assets(investor_id);
00102 *
00103 *   *****/
00104 void asset::list_assets(int iid)
00105 {
00106     string s = "";
00107     s = "SELECT A.*, AC.asset_class_name FROM " + oConfig.tbl_prefix + "_assets AS A, " +
oConfig.tbl_prefix + "_asset_classes AS AC WHERE A.asset_class = AC.asset_class_id &&
A.investor = " + std::to_string(iid) + ";";
00108     //cout << s;
00109     db.res = db.stmt->executeQuery(s);
00110
00111     s = "";
00112     while (db.res->next())
00113     {
00114         s = s + "<tr>";
00115         s = s + "<td>" + db.res->getString("description") + "</td>";
00116         s = s + "<td>" + db.res->getString("asset_class_name") + "</td>";
00117         s = s + "<td>" + db.res->getString("amount") + "</td>";
00118         s = s + "<td>" + db.res->getString("currency") + "</td>";
00119
00120         switch (oConfig.layout)
00121         {
00122         case '1' :
00123             s = s + "<td><a href=\"" + oConfig.soft_url + "?a=ae&iid=" + std::to_string(iid)

```

```

    + "&aid=" + db.res->getString("asset_id") + "\"><img src=\"" + oConfig.
css_dir + "pen.svg\" alt=\"edit\" width=\"16\" height=\"16\"></a>&nbsp;";
00124     s = s + "<a href=\"" + oConfig.soft_url + "?a=ad&aid=" + std::to_string(iid) + "
&aid=" + db.res->getString("asset_id") + "\"><img src=\"" + oConfig.
css_dir + "x.svg\" alt=\"delete\" width=\"16\" height=\"16\"></a></td>";
00125     break;
00126     case '2' :
00127         s = s + "<td><a class=\"btn btn-default\" href=\"" + oConfig.
soft_url + "?a=ae&aid=" + std::to_string(iid) + "&aid=" + db.
res->getString("asset_id") + "\" data-toggle=\"tooltip\" title=\"" + t_edit + "\"><span class=\"
glyphicon glyphicon-pencil\" style=\"color:blue\"></span></a>&nbsp;";
00128         s = s + "<a href=\"" + oConfig.soft_url + "?a=ad&aid=" + std::to_string(iid) + "
&aid=" + db.res->getString("asset_id") + "\" data-toggle=\"tooltip\" title=\"" +
t_delete + "\"><span class=\"glyphicon glyphicon-remove\" style=\"color:red\"></span></a></td>";
00129     break;
00130     default :
00131         s = s + "<td><a href=\"" + oConfig.soft_url + "?a=ae&aid=" + std::to_string(iid)
+ "&aid=" + db.res->getString("asset_id") + "\"><img src=\"" + oConfig.
css_dir + "pen.svg\" alt=\"edit\" width=\"16\" height=\"16\"></a>&nbsp;";
00132         s = s + "<a href=\"" + oConfig.soft_url + "?a=ad&aid=" + std::to_string(iid) + "
&aid=" + db.res->getString("asset_id") + "\"><img src=\"" + oConfig.
css_dir + "x.svg\" alt=\"delete\" width=\"16\" height=\"16\"></a></td>";
00133     break;
00134     }
00135     s = s + "</tr>";
00136     }
00137     cout << s;
00138     }
00139
00140     /*****
00141     * set functions
00142     * *****/
00143
00144 bool asset::set_asset_id(const std::string& s)
00145 {
00146     int a = atoi(s.c_str());
00147     bool rc = true;
00148
00149     if ((a < 0) || (a > 1000000)) //TODO : ad real check if investor_id exists
00150                                     // and if not, block IP and report!!!
00151     {
00152         rc = false;
00153         error_message = error_message + "<li>Sorry, an error ocured. The asset
cannot be found.</li>";
00154     }
00155     else {asset_id = a;}
00156     return rc;
00157 }
00158
00159 bool asset::set_investor(const std::string& s)
00160 {
00161     int a = atoi(s.c_str());
00162     bool rc = true;
00163
00164     if ((a < 0) || (a > 1000000)) //TODO : ad real check if investor_id exists
00165                                     // and if not, block IP and report!!!
00166     {
00167         rc = false;
00168         error_message = error_message + "<li>...please contact me, I know ...</li>";
00169     }
00170     else {investor = a;}
00171     return rc;
00172 }
00173
00174 bool asset::set_description(const std::string& s)
00175 {
00176     bool rc = true;
00177     if (s.length() < 2)
00178     {
00179         rc = false;
00180         error_message = error_message + "<li>a description of the asset is
required</li>";
00181     }
00182     else {description = s;}
00183     return rc;
00184 }
00185
00186 bool asset::set_asset_class(const std::string& s)
00187 {
00188     int a = atoi(s.c_str());
00189     bool rc = true;
00190
00191     if ((a < 0) || (a > 100)) //TODO : ad real check if asset_class exists
00192     {
00193         rc = false;
00194         error_message = error_message + "<li>...please contact me, I know ...</li>";
00195     }
00196     }
00197 }

```

```

00200     else {asset_class = a;}
00201     return rc;
00202 }
00203
00204 bool asset::set_amount(const std::string& s)
00205 {
00206     int a = atof(s.c_str());
00207     bool rc = true;
00208
00209     if ((a < 0) || (a > 100000000))
00210     {
00211         rc = false;
00212         error_message = error_message + "<li>the amount seems wrong (please correct
or email the " + t_author + ")</li>";
00213     }
00214     else {amount = a;}
00215     return rc;
00216 }
00217
00218 bool asset::set_currency(const std::string& s)
00219 {
00220     bool rc = true;
00221     if (s.length() != 3) //TODO: do a real check if the currency exists
00222     {
00223         rc = false;
00224         error_message = error_message + "<li>Invalid currency ... how did you do
that?</li>";
00225     }
00226     else {currency = s;}
00227     return rc;
00228 }
00229
00230 bool asset::save()
00231 {
00232     string s;
00233     if (!exists(asset_id))
00234     {
00235         s = "INSERT INTO " + oConfig.tbl_prefix + "_assets (investor, asset_class,
description, currency, amount) ";
00236         s = s + "value (" + std::to_string(investor) + ", " + std::to_string(
asset_class) + ", '" + description + "', ";
00237         s = s + "'" + currency + "', " + std::to_string(amount) + ")";
00238     }
00239     else
00240     {
00241         s = "UPDATE " + oConfig.tbl_prefix + "_assets SET asset_class=" + std::to_string(
asset_class);
00242         s = s + ", description='" + description + "', currency='" +
currency + "', amount=" + std::to_string(amount);
00243         s = s + " WHERE asset_id=" + std::to_string(asset_id) + ";";
00244     }
00245     if (db.runSQL(s))
00246     {return true;}
00247     else
00248     {
00249 #ifdef DEBUG
00250     cout << s;
00251 #endif
00252         error_message = error_message + "Sorry, I failed to add/edit the asset :-/";
00253         return false;
00254     }
00255 }
00256
00257 bool asset::exists(int aid)
00258 {
00259     string s;
00260     bool rc = false;
00261     s = "SELECT COUNT(*) AS CNT FROM " + oConfig.tbl_prefix + "_assets WHERE asset_id = " +
std::to_string(aid) + ";";
00262     db.res = db.stmt->executeQuery(s);
00263     while (db.res->next()) {rc = (db.res->getString("CNT") == "1") ? true : false;}
00264     return rc;
00265 }
00266
00267 bool asset::asset_delete()
00268 {
00269     string s;
00270     s = "DELETE FROM " + oConfig.tbl_prefix + "_assets WHERE asset_id=" + std::to_string(
asset_id) + ";";
00271     if (db.runSQL(s))
00272     {return true;}
00273     else
00274     {
00275 #ifdef DEBUG
00276     cout << s;
00277 #endif
00278         error_message = error_message + "Sorry, I failed to delete the asset :-/";

```

```

00279     return false;
00280   }
00281 }

```

8.5 asset_class.class.cpp File Reference

Classes

- class [asset_class](#)

8.6 asset_class.class.cpp

```

00001 /*
00002  *
00003  * class assets_class
00004  *
00005  * copyright: (c) Philippe De Brouwer 2014
00006  *
00007  * last modification:
00008  *
00009  */
00010
00011 class asset_class
00012 {
00013 public:
00014     int     asset_class_id;
00015     string  asset_class_name;
00016     float   E_R;
00017
00018     string  dropDown_assetClass(int defaultAC = 0);
00019     bool    exists(int acID);
00020     std::map<int, string> get_assetList();
00021
00022 private:
00023 };
00024
00025
00026 /* *****
00027  *
00028  * exists
00029  * *****/
00030 /* returns "" if investment is valid */
00031 bool asset_class::exists(int acID)
00032 {
00033     // if exists asset_class
00034
00035     // correct currency
00036     return true;
00037 }
00038
00039 /******
00040  * dropDown_assetClass
00041  *
00042  * returns a string that is a dropdown box of all asset classes
00043  * *****/
00044 string asset_class::dropDown_assetClass(int defaultAC)
00045 {
00046     string s;
00047
00048     s = s + "<select required name=\"asset_class\">";
00049     db.res = db.stmt->executeQuery("SELECT asset_class_id, asset_class_name FROM " +
oConfig.tbl_prefix + "_asset_classes;");
00050     while (db.res->next())
00051     {
00052         s = s + "<option value=\"" + db.res->getString("asset_class_id") + "\"";
00053         if (db.res->getString("asset_class_id") == std::to_string(defaultAC)) {s = s + " selected "; }
00054         s = s + ">" + db.res->getString("asset_class_name") + "</option>";
00055     }
00056     s = s + "</select>";
00057     return s;
00058 }
00059
00060
00064 std::map<int, string> asset_class::get_assetList()
00065 {
00066     int i;
00067     std::map<int, string> theList;

```

```

00068     db.res = db.stmt->executeQuery("SELECT * FROM " + oConfig.
tbl_prefix + "_asset_classes ORDER BY asset_class_id;");
00069     while (db.res->next())
00070     {
00071         i = atoi(db.res->getString("asset_class_id").c_str());
00072         theList[i] = db.res->getString("asset_class_name");
00073     }
00074     return theList;
00075 }

```

8.7 bona.class.cpp File Reference

Classes

- class [bona](#)

8.8 bona.class.cpp

```

00001 /*
00002  *
00003  * class bona
00004  *
00005  * copyright: (c) Philippe De Brouwer 2014
00006  *
00007  * last modification:
00008  *
00009  */
00010
00011 class bona
00012 {
00013 public:
00014     std::map<int, column> tbl;
00015     string tbl_name = oConfig.tbl_prefix + "_assets";
00016     string class_name = ""; // eg. goal, cash flow, investment
00017     string action_ref; // eg. g, c, i
00018     void list (int iid, bool editable = true);
00019     bool save();
00020     void show(bool editable = true);
00021     void show_edit_form(int iid, bool showExisting = false);
00022     bool delete_bona();
00023     bool exists ();
00024     bool load_fromEnv();
00025     bool load_fromDB();
00026     std::map<string, int> defList;
00027 protected:
00028     string order_by = "";
00029 private:
00030     bool verify();
00031     void load_fromRecordSet();
00032     void show_th(bool editable);
00033     void show_tc();
00034     string s_get(string lbl);
00035     float f_get(string lbl);
00036     int i_get(string lbl);
00037     struct tm d_get(string lbl);
00038 };
00039
00040 /*****
00041  * load
00042  * loads the bona from the database if the key is set
00043  * *****/
00044 bool bona::load_fromDB()
00045 {
00046     bool rc = false;
00047     string s = "";
00048     s = "SELECT * FROM `" + tbl_name + "` WHERE " + tbl[1].column_name + " = " +
tbl[1].quote() + tbl[1].get_sValue() + tbl[1].quote() + ";";
00049     db.res = db.stmt->executeQuery(s);
00050     while (db.res->next())
00051     {
00052         load_fromRecordSet();
00053     }
00054     return rc;
00055 }
00056
00057 /*****
00058  * load_fromRecordSet

```

```

00059 * *****/
00060 void bona::load_fromRecordSet ()
00061 {
00062     for (std::map<int, column>::iterator it=tbl.begin(); it!=tbl.end(); ++it)
00063     {
00064         it->second.set_value(db.res->getString(it->second.column_name));
00065     }
00066 }
00067
00068 /*****
00069 *   show_th == table header
00070 * *****/
00071 void bona::show_th(bool editable)
00072 {
00073     cout << "<table class=\"table table-hover \" + class_name + \">\n";
00074     // the table header
00075     cout << "<thead><tr>";
00076     for (std::map<int, column>::iterator it=tbl.begin(); it!=tbl.end(); ++it)
00077     {
00078         if (it->second.input_type != "hidden")
00079         {
00080             if (it->second.show_with_previous)
00081             { cout << "<br>";}
00082             else
00083             { cout << "<th>";}
00084             cout << it->second.label;
00085             std::map<int, column>::iterator it_2 = it;
00086             if (it!=tbl.end()) it_2++;
00087             if (it==tbl.end() || !(it_2->second.show_with_previous))
00088             {
00089                 cout << "</th>";
00090             }
00091             // cout << "<th>" << it->second.label << "</th>";
00092         }
00093     }
00094     if (editable) cout << "<th>Actions</th>";
00095     cout << "</tr></thead><tbody>\n";
00096 }
00097 /*****
00098 *   show_tc == table close
00099 * *****/
00100 void bona::show_tc()
00101 {
00102     cout << "</tbody></table>\n";
00103 }
00104
00105 /*****
00106 *   show_edit
00107 * *****/
00108 void bona::show_edit_form(int iid, bool showExisting)
00109 {
00110     std::map<int, column>::iterator it;
00111     cout << "<span id=\"is\" << this->class_name << \"></span>";
00112     cout << "<form method=\"post\" action=\"\" + oConfig.soft_url + "?a=" +
00113     action_ref + "s\">";
00114     show_th(true); //of course editable ;- )
00115     // the input form
00116     cout << "<tr>";
00117     for (it=tbl.begin(); it!=tbl.end(); ++it)
00118     {
00119         if (it->second.show_with_previous)
00120         { cout << "<br>";}
00121         else
00122         { if (it->second.input_type != "hidden") cout << "<td>";}
00123         it->second.show_inputField(iid, showExisting);
00124         std::map<int, column>::iterator it_2 = it;
00125         if (it!=tbl.end()) it_2++;
00126         if (it==tbl.end() || !(it_2->second.show_with_previous))
00127         {
00128             if (it->second.input_type != "hidden") cout << "</td>";
00129         }
00130     }
00131     switch (oConfig.layout)
00132     {
00133         case '1' :
00134             cout << "<td><button type=\"submit\" value=\"Submit\"><img src=\"\" <<
00135             oConfig.css_dir << "check.svg\" alt=\"add\" width=\"16\" alt=\"Save\"></button>";
00136             cout << "<a href=\"\" + oConfig.soft_url + "?a=" <<
00137             action_ref << "l&id=" << iid << "\" class=\"button\"><img src=\"\" <<
00138             oConfig.css_dir << "arrow_left.svg\" alt=\"Go back without saving the changes!\" width=\"16\"
00139             height=\"16\"></a></td>";
00140             break;
00141         case '2' :
00142             cout << "<td><button type=\"submit\" value=\"Submit\" data-toggle=\"tooltip\" title=\"\" <<
00143             t_save << "\"><span class=\"glyphicon glyphicon-ok\" style=\"color:green\"></span></button>";
00144             cout << "<a data-toggle=\"tooltip\" title=\"\" << t_cancel << "\" href=\"\" +
00145             oConfig.soft_url + "?a=" << action_ref << "l&id=" << iid << "\"><span

```

```

class=\<span></span></a></td>" ;
00139     break;
00140     default :
00141         cout << "<td><button type=\"submit\" value=\"Submit\"><img src=\"\" <<
oConfig.css_dir << "check.svg\" alt=\"add\" width=\"16\" alt=\"Save\"></button>";
00142         cout << "<a href=\"\" + oConfig.soft_url + "?a=" <<
action_ref << "l&id=" << iid << "\" class=\"button\"><img src=\"\" <<
oConfig.css_dir << "arrow_left.svg\" alt=\"Go back without saving the changes!\" width=\"16\"
height=\"16\"></a></td>" ;
00143     break;
00144     }
00145
00146     cout << "</tr>\n";
00147     show_tc();
00148     cout << "</form>";
00149 }
00150
00151
00152
00157 void bona::list(int iid, bool editable)
00158 {
00159     std::map<int, column>::iterator it;
00160
00161     cout << "<span id=\"is\" << this->class_name << "\"></span>";
00162
00163     if (class_name != "")
00164     {
00165         cout << "<div class=\"alert alert-info\">";
00166         cout << "<p>" << t_before_list[this->class_name] << "</p>";
00167         cout << "</div>\n";
00168     }
00169
00170
00171     if (editable) cout << "<form method=\"post\" action=\"\" + oConfig.
soft_url + "?a=" + action_ref + "a\">";
00172     show_th(editable);
00173
00174     // the list of existing bonas
00175     string s = "";
00176     s = "SELECT * FROM " + tbl_name + " WHERE investor = " + to_string(iid);
00177     if (this->order_by != "") s = s + " ORDER BY " + this->order_by;
00178     s = s + ";";
00179     db.res = db.stmt->executeQuery(s);
00180     while (db.res->next())
00181     {
00182         load_fromRecordSet();
00183         show(true); // includes <tr> tags
00184     }
00185
00186     // the input form
00187     if (editable)
00188     {
00189         cout << "<tr>";
00190         for (it=tbl.begin(); it!=tbl.end(); ++it)
00191         {
00192             if (it->second.input_type != "hidden")
00193             {
00194
00195                 if (it->second.show_with_previous)
00196                 { cout << "<br>";}
00197                 else
00198                 { cout << "<td>";}
00199                 it->second.show_inputField(iid, false);
00200                 std::map<int, column>::iterator it_2 = it;
00201                 if (it!=tbl.end()) it_2++;
00202                 if (it==tbl.end() || !(it_2->second.show_with_previous))
00203                 {
00204                     cout << "</td>";
00205                 }
00206             }
00207         }
00208         cout << "<td>";
00209         switch (oConfig.layout)
00210         {
00211             case '1' :
00212                 cout << "<button type=\"submit\" value=\"Submit\"><img src=\"\" <<
oConfig.css_dir << "plus.svg\" alt=\"add\" width=\"16\" height=\"16\" alt=\"Save\"></button>";
00213                 cout << "<button type=\"reset\"><img src=\"\" << oConfig.css_dir << "undo.svg\" alt=\"
Reset!\" width=\"16\" height=\"16\"></button></td>" ;
00214                 break;
00215             case '2' :
00216                 cout << "<button type=\"submit\" value=\"Submit\" data-toggle=\"tooltip\" title=\"\" <<
t_save << "\"><span class=\"glyphicon glyphicon-ok\" style=\"color:green\"></span></button>";
00217                 cout << "<button type=\"reset\" data-toggle=\"tooltip\" title=\"\" <<
t_reset << "\"><span class=\"glyphicon glyphicon-chevron-left\" style=\"color:blue\"
></span></button></td>" ;
00218                 break;

```

```

00219     default :
00220         cout << "<button type=\"submit\" value=\"Submit\"><img src=\"\" <<
oConfig.css_dir << "plus.svg\" alt=\"add\" width=\"16\" height=\"16\" alt=\"Save\"></button>"
;
00221         cout << "<button type=\"reset\"><img src=\"\" << oConfig.css_dir << "undo.svg\" alt=\"
Reset\" width=\"16\" height=\"16\"></button></td>" ;
00222     break;
00223     }
00224     cout << "</tr>\n";
00225     }
00226     show_tc();
00227     cout << "<input name=\"iid\" type=\"hidden\" value=\"\" << to_string(iid) << "\">";
00228     cout << "</form>";
00229 }
00230
00231 /*****
00232 * show
00233 * *****/
00234 void bona::show(bool editable)
00235 {
00236     cout << "<tr>";
00237     for (std::map<int, column>::iterator it=tbl.begin(); it!=tbl.end(); ++it)
00238     {
00239         if (it->second.input_type != "hidden")
00240         {
00241             if (it->second.show_with_previous)
00242             { cout << "<br>";}
00243             else
00244             { cout << "<td>";}
00245             it->second.show();
00246             std::map<int, column>::iterator it_2 = it;
00247             if (it!=tbl.end()) it_2++;
00248             if (it==tbl.end() || !(it_2->second.show_with_previous))
00249             {
00250                 cout << "</td>";
00251             }
00252         }
00253     }
00254     if (editable)
00255     {
00256         switch (oConfig.layout)
00257         {
00258             case '1' :
00259                 cout << "<td><a href=\"\" + oConfig.soft_url + "?a=" +
action_ref + "e&amp;" + action_ref + "id=" + tbl[1].get_sValue() + "\"><img src=\"\"
<< oConfig.css_dir << "pen.svg\" width=\"16\" height=\"16\" ></a>&nbsp;<a href=\"\" +
oConfig.soft_url + "?a=" + action_ref + "d&amp;" +
action_ref + "id=" + tbl[1].get_sValue() + "\"><img src=\"\" <<
oConfig.css_dir << "x.svg\" width=\"16\" height=\"16\" ></a></td>";
00260             break;
00261             case '2' :
00262                 cout << "<td><a class=\"btn btn-default\" href=\"\" + oConfig.
soft_url + "?a=" + action_ref + "e&amp;" + action_ref + "id=" +
tbl[1].get_sValue() + "\" data-toggle=\"tooltip\" title=\"\" << t_edit << "\"><span class=\"
glyphicon glyphicon-pencil\" style=\"color:blue\"></span></a>&nbsp;<a href=\"\" +
00263                 cout << "<a href=\"\" + oConfig.soft_url + "?a=" + action_ref + "d&amp;" +
action_ref + "id=" + tbl[1].get_sValue() + "\" data-toggle=\"tooltip\" title=\"\" <<
t_delete << "\"><span class=\"glyphicon glyphicon-remove\" style=\"color:red\"></span></a></td>";
00264             break;
00265             default :
00266                 cout << "<td><a href=\"\" + oConfig.soft_url + "?a=" +
action_ref + "e&amp;" + action_ref + "id=" + tbl[1].get_sValue() + "\"><img src=\"\"
<< oConfig.css_dir << "pen.svg\" width=\"16\" height=\"16\" ></a>&nbsp;<a href=\"\" +
oConfig.soft_url + "?a=" + action_ref + "d&amp;" +
action_ref + "id=" + tbl[1].get_sValue() + "\"><img src=\"\" <<
oConfig.css_dir << "x.svg\" width=\"16\" height=\"16\" ></a></td>";
00267             break;
00268         }
00269     }
00270     cout << "</tr>\n";
00271 }
00272
00273 /*****
00274 * save
00275 * *****/
00276 bool bona::save()
00277 {
00278     std::map<int, column>::iterator it;
00279     string s;
00280     if (exists())
00281     {
00282         s = "UPDATE `" + tbl_name + "` SET ";
00283         for (it=tbl.begin(); it!=tbl.end(); ++it)
00284         {
00285             s = s + it->second.column_name + " = " + it->second.quote() + it->second.get_sValue() + it->second.
quote();
00286             if (++it!=tbl.end()) {s = s + ",";}

```

```

00287     --it;
00288     }
00289     s = s + " WHERE " + tbl[1].column_name + "=" + tbl[1].quote() + tbl[1].get_sValue() +
tbl[1].quote() + ";";
00290     }
00291     else
00292     {
00293     s = "INSERT INTO `" + tbl_name + "` (";
00294     for (it=tbl.begin(); it!=tbl.end(); ++it)
00295     {
00296     if (it->second.get_sValue().length() > 0)
00297     {
00298     s = s + it->second.column_name;
00299     if (++it!=tbl.end()) {s = s + ","; }
00300     it--;
00301     }
00302     }
00303     s = s + ") value (" ;
00304     for (it=tbl.begin(); it!=tbl.end(); ++it)
00305     {
00306     if (it->second.get_sValue().length() > 0)
00307     {
00308     s = s + it->second.quote() + it->second.get_sValue() + it->second.quote();
00309     if (++it!=tbl.end()) {s = s + ","; }
00310     it--;
00311     }
00312     }
00313     s = s + ");";
00314     }
00315     #ifdef DEBUG
00316     debug_info = debug_info + "<br>goal.save: " + s;
00317     #endif
00318
00319     if (db.runSQL(s))
00320     {return true;}
00321     else
00322     {
00323     #ifdef DEBUG
00324     debug_info = debug_info + "<br>" + s;
00325     #endif
00326     error_message = error_message + "Sorry, I failed to add the " +
class_name + "to the database.";
00327     return false;
00328     }
00329     return true;
00330
00331 }
00332
00333 /*****
00334 * load_fromEnv
00335 * *****/
00336 bool bona::load_fromEnv()
00337 {
00338     for (std::map<int, column>::iterator it=tbl.begin(); it!=tbl.end(); ++it)
00339     {
00340     // cout << it->second.post_code << "<br>";
00341     it->second.set_value(cgi(it->second.post_code));
00342     // debug_info = debug_info + "<br>" + it->second.get_sValue();
00343     }
00344     if (verify()) {return true;} else {return false;}
00345     }
00346
00347 /*****
00348 * delete_bona
00349 * *****/
00350 bool bona::delete_bona()
00351 {
00352     string s = "";
00353     s = "DELETE FROM `" + tbl_name + "` WHERE " + tbl[1].column_name + " = " +
tbl[1].quote() + tbl[1].get_sValue() + tbl[1].quote() + ";";
00354     if (db.runSQL(s))
00355     {return true;}
00356     else
00357     {
00358     #ifdef DEBUG
00359     cout << s;
00360     #endif
00361     error_message = error_message + "Sorry, I failed to delete the goal :-/";
00362     return false;
00363     }
00364 }
00365
00366 /*****
00367 * verify
00368 * *****/
00369 bool bona::verify()
00370 {

```

```

00371  bool rc = true;
00372  for (std::map<int, column>::iterator it=tbl.begin(); it!=tbl.end(); ++it)
00373  {
00374  //  if (tbl[l].get_iValue() > 0)  {rc = true;} else {rc = false;}
00375  }
00376  return rc;
00377 }
00378
00379 /*****
00380  * exits
00381  *****/
00382 bool bona::exists()
00383 {
00384     string s, q;
00385     q = tbl[l].quote();
00386     bool rc = false;
00387     s = "SELECT COUNT(*) AS CNT FROM `" + tbl_name + "` WHERE " + tbl[l].column_name + " = " +
tbl[l].quote() + tbl[l].get_sValue() + tbl[l].quote() + ";";
00388 #ifdef DEBUG
00389 debug_info = debug_info + "<br>bona.exists(): " + s;
00390 debug_info = debug_info + "<BR>to_string(tbl[l].get_iValue()): " + to_string(tbl[l].get_iValue()) + "
<br>";
00391 #endif
00392     if (tbl[l].get_iValue() != 0)
00393     {
00394         db.res = db.stmt->executeQuery(s);
00395         while (db.res->next()) {rc = (db.res->getString("CNT") == "1") ? true : false;}
00396         return rc;
00397     }
00398     else
00399     {return false;}
00400 } //exists
00401
00402
00403 /* ****
00404  * functions to access members via their label (such as description)
00405  * ****
00406  * usage: descr = theBona.s_get('description');
00407  */
00408 string bona::s_get(string lbl)  {return tbl[defList[lbl]].get_sValue();}
00409 float  bona::f_get(string lbl)  {return tbl[defList[lbl]].get_fValue();}
00410 int    bona::i_get(string lbl)  {return tbl[defList[lbl]].get_iValue();}
00411 struct tm  bona::d_get(string lbl)  {return tbl[defList[lbl]].get_dValue();}

```

8.9 bona_cf.class.cpp File Reference

Classes

- class [cash_flow](#)

8.10 bona_cf.class.cpp

```

00001 /*
00002  *
00003  * class cash_flow
00004  *
00005  * copyright: (c) Philippe De Brouwer 2014
00006  *
00007  * last modification:
00008  *
00009  */
00010
00011 class cash_flow: public bona
00012 {
00013 public:
00014 // -- the specific fields ready to use. ... unless we can make the tbl map map::<string, column> so that
we can reference
00015 // the tbl by tbl['currency'] ... but can we still use the indexing then? XXX
00016
00017 /*  string get_description();
00018     float get_amount();
00019     string get_currency();
00020     tm     get_from_date();
00021     tm     get_till_date();
00022     string get_frequency();*/
00023
00024     cash_flow(); // the CONSTRUCTOR

```

```

00025 };
00026
00027
00028 /*****
00029  * cash_flow
00030  * CONSTRUCTOR
00031  * *****/
00032 cash_flow::cash_flow()
00033 {
00034     int i;
00035     tbl_name = oConfig.tbl_prefix + "_cash_flows";
00036     class_name = "cash_flow";
00037     action_ref = "c";
00038     i=1;
00039     tbl[i].column_name = "cash_flow_id";
00040     tbl[i].post_code = "cid";
00041     tbl[i].placeholder = "";
00042     tbl[i].col_type = 'i';
00043     tbl[i].input_type = "hidden";
00044     defList[tbl[i].column_name] = i;
00045     i++;
00046     tbl[i].column_name = "investor";
00047     tbl[i].post_code = "iid";
00048     tbl[i].placeholder = "";
00049     tbl[i].col_type = 'i';
00050     tbl[i].input_type = "hidden";
00051     defList[tbl[i].column_name] = i;
00052     i++;
00053     tbl[i].column_name = "description";
00054     tbl[i].post_code = "desc";
00055     tbl[i].label = "Description";
00056     tbl[i].placeholder = "describe the cash_flow in your own words";
00057     tbl[i].col_type = 's';
00058     tbl[i].input_type = "text";
00059     defList[tbl[i].column_name] = i;
00060     i++;
00061     tbl[i].column_name = "amount";
00062     tbl[i].post_code = "amt";
00063     tbl[i].label = "Amount";
00064     tbl[i].placeholder = "10000";
00065     tbl[i].col_type = 'f';
00066     tbl[i].input_type = "number";
00067     defList[tbl[i].column_name] = i;
00068     i++;
00069     tbl[i].column_name = "currency";
00070     tbl[i].post_code = "curr";
00071     tbl[i].label = "Currency";
00072     tbl[i].placeholder = "5000";
00073     tbl[i].col_type = 's';
00074     tbl[i].input_type = "dropDown";
00075     tbl[i].drop_down_table = oConfig.tbl_prefix + "_currencies";
00076     tbl[i].drop_down_keyField = "currency_id";
00077     tbl[i].drop_down_keyFieldQuote = "'";
00078     tbl[i].drop_down_showField = "currency_id";
00079     tbl[i].drop_down_default = "EUR";
00080     defList[tbl[i].column_name] = i;
00081     i++;
00082     tbl[i].column_name = "from_date";
00083     tbl[i].post_code = "fdt";
00084     tbl[i].label = "From Date";
00085     tbl[i].placeholder = "2025-12-25";
00086     tbl[i].col_type = 'd';
00087     tbl[i].input_type = "date";
00088     defList[tbl[i].column_name] = i;
00089     i++;
00090     tbl[i].column_name = "till_date";
00091     tbl[i].post_code = "tdt";
00092     tbl[i].label = "Till Date";
00093     tbl[i].placeholder = "2030-12-25";
00094     tbl[i].col_type = 'd';
00095     tbl[i].input_type = "date";
00096     defList[tbl[i].column_name] = i;
00097     i++;
00098     tbl[i].column_name = "frequency";
00099     tbl[i].post_code = "freq";
00100     tbl[i].label = "Frequency";
00101     tbl[i].placeholder = "";
00102     tbl[i].col_type = 's';
00103     tbl[i].input_type = "dropDown";
00104     tbl[i].drop_down_table = oConfig.tbl_prefix + "_frequencies";
00105     tbl[i].drop_down_keyField = "frequency_id";
00106     tbl[i].drop_down_keyFieldQuote = "'";
00107     tbl[i].drop_down_showField = "frequency_name";
00108     tbl[i].drop_down_default = "M";
00109     defList[tbl[i].column_name] = i;
00110
00111     tbl[1].set_value(0); // to initialize the cash_flow_id to zero (avoid NULLs)

```

```
00112 }
```

8.11 bona_goal.class.cpp File Reference

Classes

- class [goal](#)

8.12 bona_goal.class.cpp

```
00001 /*
00002 *
00003 * class goal_di == database interface for goal
00004 *
00005 * copyright: (c) Philippe De Brouwer 2014
00006 *
00007 * last modification:
00008 *
00009 */
00010
00011 class goal: public bona
00012 {
00013 public:
00014     goal();
00015 };
00016
00017
00018 /*****
00019 * goal
00020 * CONSTRUCTOR
00021 * *****/
00022 goal::goal()
00023 {
00024     int i;
00025     tbl_name = oConfig.tbl_prefix + "_goals";
00026     class_name = "goal";
00027     action_ref = "g";
00028     order_by = "priority ASC";
00029     i=1;
00030     tbl[i].column_name = "goal_id";
00031     tbl[i].post_code = "gid";
00032     tbl[i].placeholder = "";
00033     tbl[i].col_type = 'i';
00034     tbl[i].input_type = "hidden";
00035     i++;
00036     tbl[i].column_name = "investor";
00037     tbl[i].post_code = "iid"; //this will be overwritten by Apache with the hidden iid in the form!
00038     tbl[i].placeholder = "";
00039     tbl[i].col_type = 'i';
00040     tbl[i].input_type = "hidden";
00041     i++;
00042     tbl[i].column_name = "goal_type";
00043     tbl[i].post_code = "gtype";
00044     tbl[i].label = "Goal Type";
00045     tbl[i].placeholder = "1";
00046     tbl[i].col_type = 's';
00047     tbl[i].input_type = "dropDown";
00048     tbl[i].drop_down_table = oConfig.tbl_prefix + "_goal_types";
00049     tbl[i].drop_down_keyField = "goal_type_id";
00050     tbl[i].drop_down_keyFieldQuote = "'";
00051     tbl[i].drop_down_showField = "description";
00052     tbl[i].drop_down_default = "1";
00053     defList[tbl[i].column_name] = i;
00054     i++;
00055     tbl[i].column_name = "description";
00056     tbl[i].post_code = "desc";
00057     tbl[i].label = "Goal Description";
00058     tbl[i].placeholder = "describe the goal in your own words";
00059     tbl[i].col_type = 's';
00060     tbl[i].input_type = "text";
00061     defList[tbl[i].column_name] = i;
00062     i++;
00063     tbl[i].column_name = "realization_age";
00064     tbl[i].post_code = "when";
00065     tbl[i].label = "Age of<br>Realization";
00066     tbl[i].placeholder = "66";
00067     tbl[i].col_type = 'i';
```

```

00068     tbl[i].input_type = "number";
00069     i++;
00070     tbl[i].column_name = "from_date";
00071     tbl[i].post_code = "fdt";
00072     tbl[i].label = "From Date";
00073     tbl[i].placeholder = "2025-12-25";
00074     tbl[i].col_type = 'd';
00075     tbl[i].input_type = "date";
00076     deflList[tbl[i].column_name] = i;
00077     i++;
00078     tbl[i].column_name = "till_date";
00079     tbl[i].post_code = "tdt";
00080     tbl[i].label = "Till Date";
00081     tbl[i].placeholder = "2030-12-25";
00082     tbl[i].col_type = 'd';
00083     tbl[i].input_type = "date";
00084     tbl[i].show_with_previous = true;
00085     deflList[tbl[i].column_name] = i;
00086     i++;
00087     tbl[i].column_name = "frequency";
00088     tbl[i].post_code = "freq";
00089     tbl[i].label = "Frequency";
00090     tbl[i].placeholder = "";
00091     tbl[i].col_type = 's';
00092     tbl[i].input_type = "dropDown";
00093     tbl[i].drop_down_table = oConfig.tbl_prefix + "_frequencies";
00094     tbl[i].drop_down_keyField = "frequency_id";
00095     tbl[i].drop_down_keyFieldQuote = "'";
00096     tbl[i].drop_down_showField = "frequency_name";
00097     tbl[i].drop_down_default = "M";
00098     tbl[i].show_with_previous = true;
00099     deflList[tbl[i].column_name] = i;
00100     i++;
00101     tbl[i].column_name = "amount";
00102     tbl[i].post_code = "amt";
00103     tbl[i].label = "Amount";
00104     tbl[i].placeholder = "10000";
00105     tbl[i].col_type = 'f';
00106     tbl[i].input_type = "number";
00107     i++;
00108     tbl[i].column_name = "currency";
00109     tbl[i].post_code = "curr";
00110     tbl[i].label = "Currency";
00111     //?? tbl[i].placeholder = "5000";
00112     tbl[i].col_type = 's';
00113     tbl[i].input_type = "dropDown";
00114     tbl[i].drop_down_table = oConfig.tbl_prefix + "_currencies";
00115     tbl[i].drop_down_keyField = "currency_id";
00116     tbl[i].drop_down_keyFieldQuote = "'";
00117     tbl[i].drop_down_showField = "currency_id";
00118     tbl[i].drop_down_default = "EUR";
00119     i++;
00120     tbl[i].column_name = "max_shortfall";
00121     tbl[i].post_code = "msf";
00122     // tbl[i].label = "Max Shortfall";
00123     tbl[i].placeholder = "1000";
00124     tbl[i].col_type = 'f';
00125     tbl[i].input_type = "hidden";
00126     i++;
00127     tbl[i].column_name = "priority";
00128     tbl[i].post_code = "imp";
00129     tbl[i].label = "Priority";
00130     tbl[i].placeholder = "number from 1 (highest) to 10 (lowest)";
00131     tbl[i].col_type = 'i';
00132     tbl[i].input_type = "number";
00133     /*
00134     i++;
00135     tbl[i].column_name = "need_level";
00136     tbl[i].post_code = "nlev";
00137     tbl[i].label = "Need Level";
00138     tbl[i].placeholder = "";
00139     tbl[i].col_type = 'i';
00140     tbl[i].input_type = "dropDown";
00141     tbl[i].drop_down_table = oConfig.tbl_prefix + "_need_levels";
00142     tbl[i].drop_down_keyField = "need_level_id";
00143     tbl[i].drop_down_keyFieldQuote = "";
00144     tbl[i].drop_down_showField = "description";
00145     //tbl[i].drop_down_default = "1";
00146     */
00147     tbl[1].set_value(0); // to initialize the goal_id to zero (avoid NULLs)
00148 }

```

8.13 bootstrap.namespace.cpp File Reference

Namespaces

- [bootstrap](#)

Functions

- void [bootstrap::bs_modal](#) (string title, string content, string id="myModal", string b1_text="", string b1_url="")
- void [bootstrap::bs_navbar_start](#) ()
- void [bootstrap::bs_navbar_item](#) (string my_text="", string my_url="#", bool is_active=false)
- void [bootstrap::bs_navbar_end](#) ()
- void [bootstrap::bs_navbar_end1](#) ()
- void [bootstrap::bs_navbar_end2](#) ()

8.14 bootstrap.namespace.cpp

```

00001 namespace bootstrap
00002 {
00003     void bs_modal(string title, string content, string id = "myModal", string b1_text = "", string
        b1_url = "");
00004     void bs_navbar_start();
00005     void bs_navbar_item(string my_text = "", string my_url = "#", bool is_active = false);
00006     void bs_navbar_end();
00007     void bs_navbar_end1();
00008     void bs_navbar_end2();
00009 }
00010
00011
00017 inline void bootstrap::bs_modal(string title, string content, string id, string b1_text,
        string b1_url)
00018 { // btn-lg
00019     cout << "<button class=\"btn btn-default\" data-toggle=\"modal\" data-target=\"#\" << id << \">";
00020     cout << title << "</button>";
00021     cout << "<div class=\"modal fade\" id=\"\" << id << \"\" tabindex=\"-1\" role=\"dialog\" aria-labelledby=\"
        \" << id + \"label\" << \"\" aria-hidden=\"true\">";
00022     cout <<     "<div class=\"modal-dialog\">";
00023     cout <<         "<div class=\"modal-content\">";
00024     cout <<             "<div class=\"modal-header\">";
00025     cout <<                 "<button type=\"button\" class=\"close\" data-dismiss=\"modal\" aria-hidden=\"true\"
        >&times;</button>";
00026     cout <<                     "<h4 class=\"modal-title\" id=\"\" << id + \"label\" << \">\" << title << "</h4>";
00027     cout <<                         "</div>";
00028     cout <<                             "<div class=\"modal-body\">\" << content << "</div>";
00029     cout <<                                 "<div class=\"modal-footer\">";
00030     cout <<                                     "<button type=\"button\" class=\"btn btn-default\" data-dismiss=\"modal\">Close</button>
        ";
00031     if (b1_text != "") cout << "<a href=\"\" << b1_url << \"\" class=\"btn btn-primary\">\" << b1_text << "
        </button>";
00032     //<button type=\"button\" class=\"btn btn-primary\">\" << b1_text << "</button>";
00033     cout <<         "</div>";
00034     cout <<             "</div>\"; // <!-- /.modal-content -->
00035     cout <<                 "</div>\"; //<!-- /.modal -->";
00036     cout <<                     "</div>\"; //<!-- /.modal -->";
00037 }
00038
00044 inline void bootstrap::bs_navbar_start()
00045 {
00046     cout << "<nav class=\"navbar navbar-default\">";
00047     cout << "<div class=\"container-fluid\">";
00048     cout << "<div class=\"navbar-header\">";
00049
00050     cout << "<button type=\"button\" class=\"navbar-toggle collapsed\" data-toggle=\"collapse\" data-target=
        \"#pfp-menu\" aria-expanded=\"false\">";
00051     cout << "<span class=\"sr-only\">Toggle navigation</span>";
00052     cout << "<span class=\"icon-bar\"></span>";
00053     cout << "<span class=\"icon-bar\"></span>";
00054     cout << "<span class=\"icon-bar\"></span>";
00055     cout << "</button>";
00056
00057     cout << "<a class=\"navbar-brand\" href=\"#\"><img class=\"noPadding\" src=\"\" <<
        oConfig.img_dir << \"ginkgo.png\" width=\"50px\"></a>";
00058     cout << "</div>";
00059
00060     // here follows the nav-items
00061     cout << "<div class=\"collapse navbar-collapse\" id=\"pfp-menu\">";
00062     cout << "<ul class=\"nav nav-tabs\">";
00063 }

```

```

00069 inline void bootstrap::bs_navbar_end1()
00070 {
00071     cout << "</ul>";
00072 }
00078 inline void bootstrap::bs_navbar_end2()
00079 {
00080     cout << "</div>";
00081     cout << "</div>";
00082     cout << "</nav>";
00083 }
00089 inline void bootstrap::bs_navbar_end()
00090 {
00091     bs_navbar_end1();
00092     bs_navbar_end2();
00093 }
00099 inline void bootstrap::bs_navbar_item(string my_text, string my_url, bool
is_active)
00100 {
00101     cout << "<li role=\"presentation\" ";
00102     if (is_active) {cout << "class=\"active\"";}
00103     cout << "><a href=\"\" << my_url << \"\">\" << my_text << "</a></li>";
00104 }

```

8.15 column.class.cpp File Reference

Classes

- class [column](#)

8.16 column.class.cpp

```

00001
00002
00003 /*
00004 * convention: the first column object is supposed to be the PK
00005 */
00006 class column
00007 {
00008 public:
00009     string column_name;           // name used in the database, eg. asset_class, asset_id
00010     char col_type;                // i=int, f=float, s=string, d=date
00011     // bool show_it = true;       // true if it should be shown to the user
00012     // bool is_used = false;      // true if it is used by the relevant class
00013     string input_type;            // text, number, checkbox, etc.
00014                                     // ==> set to "hidden" in order not to show it on the screen
00015                                     // ==> set to "dropDown" for drop_down box
00016                                     // ==> set to "date" for date
00017     string drop_down_table;       // eg. " + oConfig.tbl_prefix + "_asset_classes
00018     string drop_down_keyField;    // eg. asset_class_id
00019     string drop_down_keyFieldQuote; // "" or ""
00020     string drop_down_showField;   // eg. asset_class_name
00021     string drop_down_default;     // eg. USD
00022     string label;                 // eg. description, amount, etc. (column header)
00023     string placeholder;           // eg. describe the goal, boss@universe.com
00024     string post_code;             // eg. desc, amnt, aid, iid, etc. NOTE: also used for the "id" (not
only for "name") attributes
00025     bool show_with_previous = false; // eg. from_Date, till_Date and frequency to be
stacked in one table cell -->
00026                                     // in that case show_with_previous should be true for the two last
00027     string data_align = "center"; // data-align = left | right | center
00028
00029     string quote();
00030     int get_iValue();
00031     string get_sValue();
00032     float get_fValue();
00033     struct tm get_dValue();
00034     void set_value(int value);
00035     void set_value(string value);
00036     void set_value(float value);
00037     void show_inputField(int iid, bool showExisting = false); // send the input field to
stdout
00038     void show(); // send the field value to stdout in a readable form (simply get_sValue
or the dop_down_showField of the referenced table)
00039 private:
00040     int iValue;
00041     string sValue;
00042     float fValue;

```

```

00043 struct tm dValue;
00044 };
00045
00046 int column::get_iValue() { return iValue;}
00047 string column::get_sValue() { return sValue;}
00048 float column::get_fValue() { return fValue;}
00049 struct tm column::get_dValue() { return dValue;}
00050
00051 void column::set_value(int value) {
00052     iValue = value;
00053     sValue = std::to_string(value);
00054     fValue = value + 0.0;
00055 }
00056 void column::set_value(string value) {
00057     sValue = value;
00058     if (col_type == 'd') dateStr2tm(value, &dValue);
00059     if (col_type != 'd' && col_type != 's')
00060     {
00061         iValue = atoi(value.c_str());
00062         fValue = atof(value.c_str());
00063     }
00064 }
00065 void column::set_value(float value) {
00066     iValue = round(value);
00067     sValue = to_string(value);
00068     fValue = value;
00069 }
00070
00071 string column::quote() {
00072     return (col_type == 's' || col_type == 'd') ? "\"" : "";
00073 }
00074
00075 /* *****
00076 * show
00077 * *****
00078 */
00079 void column::show()
00080 {
00081     if (input_type != "hidden"){
00082         if ((input_type != "dropDown") || (drop_down_keyField ==
drop_down_showField))
00083         {
00084             if (col_type == 'f') cout << curr_format(get_fValue());
00085             else cout << get_sValue();
00086         }
00087         else
00088         {
00089             db.res2 = db.stmt->executeQuery("SELECT " + drop_down_showField + " FROM
" + drop_down_table + " WHERE " + drop_down_keyField + " = " +
drop_down_keyFieldQuote + get_sValue() +
drop_down_keyFieldQuote + ";");
00090             while (db.res2->next()) cout << db.res2->getString(
drop_down_showField);
00091         }
00092     }
00093 }
00094
00095 /* *****
00096 *show_inputField
00097 * *****
00098 */
00099 void column::show_inputField(int iid, bool showExisting)
00100 {
00101     string input_type fld;
00102     if (input_type == "dropDown")
00103     {
00104         cout << "<select ";
00105         if (input_type != "hidden") cout << "required ";
00106         cout << "name=\"" + post_code + "\" id=\"" + post_code + "\">";
00107         db.res = db.stmt->executeQuery("SELECT " + drop_down_keyField + ", " +
drop_down_showField + " FROM " + drop_down_table + ";");
00108         while (db.res->next())
00109         {
00110             cout << "<option value=\"" + db.res->getString(drop_down_keyField) + "\"";
00111             if (showExisting)
00112             {
00113                 if (db.res->getString(drop_down_keyField) ==
get_sValue()) cout << " selected ";
00114             }
00115             else
00116             {
00117                 if (db.res->getString(drop_down_keyField) ==
drop_down_default) {cout << " selected "; }
00118             }
00119             cout << ">" << db.res->getString(drop_down_showField) + "</option>";
00120         }
00121         cout << "</select>";

```

```

00122     }
00123     else
00124     {
00125         input_type_fld = (input_type != "date")?input_type:"text\" data-date-format=\"
yyyy-mm-dd\";
00126         cout << "<input type=\"" << input_type_fld << "\" name=\"" << post_code << "\" id=\"" <<
post_code << "\" ";
00127         if (input_type != "hidden")
00128         {
00129             cout << " required ";
00130             if (col_type == 'i') cout << " size=\""1\" ";
00131             else if (col_type == 'f') cout << " size=\""5\" ";
00132             else if (col_type == 'd') cout << " size=\""6\" ";
00133             else cout << " size=\""12\" ";
00134         }
00135         if (post_code == "iid")
00136         {
00137             cout << " value=\"" << to_string(iid) << "\" ";
00138         }
00139         else
00140         {
00141             if (showExisting) cout << " value=\"" << get_sValue() << "\" ";
00142         }
00143         cout << " placeholder=\"" << placeholder << "\">\n\";
00144     }
00145 }
00146
00147 /* *****usage
00148 tbl[1].column_name = "asset_id";
00149 tbl[1].set_value(cgi("aid"));
00150 tbl[1].col_type = 'i';
00151 tbl[1].show_it = false;
00152 tbl[1].drop_down = false;
00153
00154 tbl[2].column_name = "description";
00155 tbl[2].set_value(cgi("description"));
00156 tbl[2].col_type = 's';
00157 tbl[2].show_it = true;
00158 tbl[2].drop_down = false;
00159
00160 for (tbl::iterator it=tbl.begin(); it!=tbl.end(); ++it)
00161     std::cout << it->column_name << " => " << it->get_value() << '\n';
00162
00163 for (int i , i < 10, i++)
00164 {cout << i << ": exists:" << exists(i) << "\n";}
00165
00166 return o;
00167 } //main
00168 */

```

8.17 config.class.cpp File Reference

Classes

- class [config](#)

Macros

- #define [LOCAL](#)

8.17.1 Macro Definition Documentation

8.17.1.1 #define LOCAL

Definition at line 43 of file [config.class.cpp](#).

8.18 config.class.cpp

```

00001
00010 class config

```

```

00011 {
00012 public:
00013
00015 char layout = '2';
00016 char language = '1';
00017 bool show_aside = false;
00018 char font_combination = '5';
00019 const bool parse_wrapper = false;
00020 const string wrapper_name = "eFinancialPlanner";
00021 const string wrapper_url = "http://eFinancialPlanner.eu";
00022 const string soft_name = "Goal Ranking 1";
00023 const string soft_version = "V0.4.0";
00024 std::vector<int> total_portf_monthNbrs = {1};
00025 const bool show_social = false;
00026 const bool merge_ER_vol = true;
00027 const bool simul_show_table = false;
00028
00030 string riskFunction = "VaR";
00031
00033 const string db_host = "tcp://127.0.0.1:3306";
00034 const string db_name = "efp";
00035 const string db_user = "efp";
00036 const string db_pwd = "efp.cger";
00037 const string tbl_prefix = "pfp3";
00038
00039
00041 const string soft_file = "pfp5.cgi";
00042
00043 #define LOCAL
00044 #ifdef LOCAL
00045 const string soft_server_url = "http://localhost";
00046 const string www_dir = "/efp/";
00047 const string img_dir = "/efp/img/";
00048 const string css_dir = "/efp/style/";
00049 const string soft_dir = "/cgi-bin/";
00050 const string scripts_dir = "/efp/scripts/";
00051 #else
00052 const string soft_server_url = "http://www.phdb.nazwa.pl";
00053 const string soft_dir = "/cgi-bin/";
00054 const string www_dir = "/";
00055 const string img_dir = "/img/";
00056 const string css_dir = "../style/";
00057 const string scripts_dir = "../scripts/";
00058 #endif
00059 string soft_url = this->soft_server_url + this->
soft_dir + this->soft_file;
00060 };

```

8.19 currency.class.cpp File Reference

Classes

- class `cls_currency`

8.20 currency.class.cpp

```

00001 class cls_currency
00002 {
00003 public:
00004
00005     string entity;
00006     string currency_name;
00007     string currency_id;
00008     //char symbol;
00009
00010     string dropDown_currency(string defaultCurr = "EUR");
00011     bool set_curr(string the_curr);
00012     float to_curr(float amnt, string the_curr);
00013     float conversion_factor(string from_curr, string to_curr);
00014 };
00015
00016 /*****
00017 * dropDown_currency
00018 *
00019 * *****/
00020 string cls_currency::dropDown_currency(string defaultCurr)
00021 {

```

```

00022     string s;
00023
00024     s = s + "<select required name=\"currency\">";
00025     db.res = db.stmt->executeQuery("SELECT currency_id FROM " + oConfig.
tbl_prefix + "_currencies;");
00026     while (db.res->next())
00027     {
00028         s = s + "<option value=\"" + db.res->getString("currency_id") + "\"";
00029         if (db.res->getString("currency_id") == defaultCurr) {s = s + " selected "; }
00030         s = s + ">" + db.res->getString("currency_id") + "</option>";
00031     }
00032     s = s + "</select>";
00033     return s;
00034 }
00035
00041 bool cls_currency::set_curr(string the_curr)
00042 {
00043     string s;
00044     db_helper this_db;
00045     //s = "SELECT COUNT(*) AS EXIST_NBR from " + oConfig.tbl_prefix + "_currencies WHERE currency_id = ' " +
the_curr + "'";
00046     s = "SELECT currency_id FROM " + oConfig.tbl_prefix + "_currencies WHERE currency_id = '
" + the_curr + "'";
00047     this_db.res = this_db.stmt->executeQuery(s);
00048     if(this_db.res->rowCount() == 1)
00049     {
00050         this->currency_id = the_curr;
00051         return true;
00052     }
00053     else
00054     {
00055         error_message = error_message + "<li>ERRNO3411</li>";
00056         return false; //table not found ;- )
00057     }
00058     error_message = error_message + "<li>ERRNO3412</li>";
00059     return false; //table not found ;- )
00060 }
00061
00067 inline float cls_currency::to_curr(float amnt, string to_curr)
00068 {
00069     if (to_curr == this->currency_id) { return amnt; }
00070     db_helper this_db;
00071     string s;
00072
00074     s = "SELECT * FROM " + oConfig.tbl_prefix + "_currencies WHERE currency_id = '" + this->
currency_id + "'";
00075     this_db.res = this_db.stmt->executeQuery(s);
00076     if (this_db.res->next())
00077     {
00078         amnt = amnt * atof(this_db.res->getString("eur_p_one").c_str());
00079     }
00080     else
00081     {
00082         error_message = error_message + "<li>ERRNO.C.1011: invalid start
currency</li>";
00083         return 0; //no currency
00084     }
00085
00087     if (to_curr != "EUR")
00088     {
00089         s = "SELECT * FROM " + oConfig.tbl_prefix + "_currencies WHERE currency_id = '" +
to_curr + "'";
00090         this_db.res = this_db.stmt->executeQuery(s);
00091         if (this_db.res->next())
00092         {
00093             amnt = amnt * atof(this_db.res->getString("one_p_eur").c_str());
00094         }
00095         else
00096         {
00097             error_message = error_message + "<li>ERRNO.C.1010: invalid destiny currency
in to_curr</li>";
00098             return 0;
00099         }
00100     }
00101
00103     return amnt;
00104 }
00105

```

8.21 db_helper.class.cpp File Reference

Classes

- class [db_helper](#)

8.22 db_helper.class.cpp

```

00001 /*
00002 *
00003 * class db_helper
00004 *
00005 * copyright: (c) Philippe De Brouwer 2014
00006 *
00007 * last modification:
00008 *
00009 */
00010
00011 class db_helper
00012 {
00013 public:
00014
00015     sql::Driver          *driver;
00016     sql::Connection     *con;
00017     sql::Statement      *stmt;
00018     sql::ResultSet      *res;
00019     sql::ResultSet      *res2; // spare one to use when eg. pulling the drop-down list while res
                                refers to the master object
00020     sql::PreparedStatement *pstmt;
00021
00022     //string host = "tcp://127.0.0.1:3306";
00023     //string user = "efp";
00024     //string pwd = "efp.cger";
00025     string db = "efp";
00026
00027     db_helper();
00028     ~db_helper();
00029     bool runSQL(string s);
00030
00031     float getFloat(string s);
00032     int getInt(string s);
00033
00034 private:
00035 };
00036
00037 /* *****
00038 * CONSTRUCTOR
00039 * ******/
00040 db_helper::db_helper()
00041 {
00042     try {
00043         /* Create a connection */
00044         driver = get_driver_instance();
00045         con = driver->connect(oConfig.db_host,oConfig.
db_user,oConfig.db_pwd);
00046         // con->setSchema("efp");
00047         con->setSchema(oConfig.db_name);
00048         stmt = con->createStatement();
00049
00050         /* res = stmt->executeQuery("SELECT 'Hello World!' AS _message");
00051         while (res->next()) {
00052             cout << "\t... MySQL replies: ";
00053             // Access column data by alias or column name
00054             cout << res->getString("_message") << endl;
00055             cout << "\t... MySQL says it again: ";
00056             // Access column data by numeric offset, 1 is the first column
00057             cout << res->getString(1) << endl;
00058         */
00059         /* '?' is the supported placeholder syntax */
00060         /*pstmt = con->prepareStatement("INSERT INTO test(id) VALUES (?)");
00061         for (int i = 1; i <= 10; i++) {
00062             pstmt->setInt(1, i);
00063             pstmt->executeUpdate();
00064         */
00065         delete pstmt;
00066     }
00067
00068     /* //Select in ascending order
00069     pstmt = con->prepareStatement("SELECT id FROM test ORDER BY id ASC");
00070     res = pstmt->executeQuery();
00071     //Fetch in reverse = descending order!
00072     res->afterLast();
00073     while (res->previous())
00074         cout << "\t... MySQL counts: " << res->getInt("id") << endl;

```

```

00075 delete res;
00076 delete pstmt;
00077 delete con
00078 */
00079 } catch (sql::SQLException &e) {
00080     cout << "# ERR: SQLException in " << __FILE__;
00081     cout << "(" << __FUNCTION__ << ") on line " << __LINE__ << endl;
00082     cout << "# ERR: " << e.what();
00083     cout << " (MySQL error code: " << e.getErrorCode();
00084     cout << ", SQLState: " << e.getSQLState() << ")" << endl;
00085 }
00086 }
00087
00088 /* *****
00089  * DESTRUCTOR
00090  * *****/
00091 db_helper::~db_helper()
00092 {
00093     delete res;
00094     delete stmt;
00095     delete con;
00096 }
00097
00098 /* *****
00099  * runSQL
00100  * *****/
00101
00102 bool db_helper::runSQL(string s)
00103 {
00104     bool bSuccess = true;
00105     try
00106     {
00107
00108         sql::Driver *driver;
00109         sql::Connection *con;
00110         sql::Statement *stmt;
00111         //sql::ResultSet *res;
00112         //sql::PreparedStatement *pstmt;
00113         // Create a connection
00114         driver = get_driver_instance();
00115         con = driver->connect(oConfig.db_host,oConfig.db_user,
oConfig.db_pwd);
00116         con->setSchema(oConfig.db_name);
00117
00118
00119         stmt = con->createStatement();
00120         //res = stmt->executeQuery(s); // ==> if it is supposed to return a resultset
00121         stmt->execute(s); // ==> if it is NOT supposed to return a resultset
00122     } //try
00123
00124     catch (sql::SQLException &e)
00125     {
00126         bSuccess = false;
00127 #ifdef DEBUG
00128     cout << "# ERR: SQLException in " << __FILE__;
00129     cout << "(" << __FUNCTION__ << ") on line " << __LINE__ << endl;
00130     cout << "# ERR: " << e.what();
00131     cout << " (MySQL error code: " << e.getErrorCode();
00132     cout << ", SQLState: " << e.getSQLState() << ")" << endl;
00133 #endif
00134     }
00135 // if (EXIT_SUCCESS == 0) {return true;} else {return false;} // return EXIT_SUCCESS;
00136 if (bSuccess) {return true;} else {return false;}
00137 }
00138
00142 inline float db_helper::getFloat(string s)
00143 {
00144     return atof(this->res->getString(s).c_str());
00145 }
00146
00150 inline int db_helper::getInt(string s)
00151 {
00152     return atoi(this->res->getString(s).c_str());
00153 }

```

8.23 efp.cpp File Reference

```
#include <iostream>
#include <string>
#include <ctime>
#include <vector>
#include <map>
#include <unordered_map>
#include <math.h>
#include <locale>
#include <iomanip>
#include <errno.h>
#include "normdist.h"
#include <cgicc/Cgicc.h>
#include <cgicc/HTTPHTMLHeader.h>
#include <cgicc/HTMLClasses.h>
#include <cppconn/prepared_statement.h>
#include <mysql_connection.h>
#include <driver.h>
#include <exception.h>
#include <resultset.h>
#include <statement.h>
#include <string.h>
#include <cmath>
#include "config.class.cpp"
#include "t_personalize_Eng_UK.h"
#include "texts_Eng_UK.h"
#include "t_disclaimer_Eng_UK.h"
#include "global_functions.h"
#include "bootstrap.namespace.cpp"
#include "html_helper.class.cpp"
#include "db_helper.class.cpp"
#include "frequency.class.cpp"
#include "currency.class.cpp"
#include "asset_class.class.cpp"
#include "asset.class.cpp"
#include "column.class.cpp"
#include "bona.class.cpp"
#include "bona_goal.class.cpp"
#include "bona_cf.class.cpp"
#include "portfolio.class.cpp"
#include "market.class.cpp"
#include "investor.class.cpp"
#include "aGOAL.struct.cpp"
#include "investment_problem.class.cpp"
#include "simulation.class.cpp"
#include "follow_up.class.cpp"
#include "investor_ui.class.cpp"
```

Macros

- #define [SECONDS_IN_MONTH](#) 2629800
Marco's:
- #define [MAX_AGE](#) 150
the maximum age allowed in the goals
- #define [MIN_ALPHA](#) 0.075

- = (1 - confidence_level) for the most important goal*
- #define `MAX_ALPHA` 0.25
 - = (1 - confidence_level) for the least important goal*
- #define `ALPHA_UPPER_LIMIT` 0.4
 - the upper limit for all alpha's to be used*
- #define `ALPHA_LOWER_LIMIT` 0.1
 - the lower limit for all alpha's to be used*
- #define `ALPHA_PLOT_LOW` 0.1
 - the low alpha value to plot*
- #define `ALPHA_PLOT_HIGH` 0.9
 - the high alpha to plot*
- #define `RAINY_DAY_MONTH_NBR` 3
 - the artificial horizon in months for the "rainy day goal"*
- #define `RETIREMENT_AGE` 65
 - the standard retirement age (applied to everyone!)*
- #define `SIMULATE_TILL_DEFAULT` 90
 - the standard age to stop the simulation*
- #define `MIN_YEARS_TO_SIMULATE` 2
 - the minimal number of years that can be simulated with this tool*
- #define `NBR_ITERATIONS` 20
 - number of iterations in goalseek procedures*
- #define `PRECISION` 0.000001
 - equals roundup (1 / 2^{NBR_ITERATIONS})*
- #define `CURR_LEN` 14
 - the maximum length of an amount in number of digits (eg. 14)*
- #define `NBR_TABS` 7
 - number of tabs in the main menu of the software*
- #define `NBR_SUB_TABS` 3
 - maximum number of tabs used in the sub-menu*
- #define `NBR_RIGHT_MENU` 3
 - the number of menu items floated to the right*
- #define `INPUT_WIDTH_TEXT` 15
 - the width of input-text field in all forms*
- #define `INPUT_WIDTH_FLOAT` 5
 - the width of an input field for a float*
- #define `INPUT_WIDTH_INT` 2
 - the width of an input field for an int*
- #define `MAX_MONTHS_TO_SAFEST` 3
 - the maximum number of months for realization_monthNbr to be forced to the safest portfolio*
- #define `NBR_SCALE` 5
 - the number of items in a ordinal scale (labels are in t_scale125i2s)*
- #define `DEFAULT_SCALE` 3
 - the default value to use for preferences (on a scale from 1 to 5)*
- #define `MAX_SCALE` 5
 - the maximum value for the scale (5) (on a scale from 1 to 5)*
- #define `MIN_SCALE` 1
 - the minimum of the scale (1) (on a scale from 1 to 5)*
- #define `MAX_MNTHS_2_SIMULATE` 600
 - the maximum number of months that we allow to visualize a simulation*
- #define `MIN_MNTHS_2_SIMULATE` 1
 - the maximum number of months that we allow to visualize a simulation*

- `#define NBR_ASSET_CLASSES 10`
the number of asset classes (not dynamically updated!)
- `#define INDEX(x, y) ((y-1)+((NBR_ASSET_CLASSES)*(x-1)))`
*this macro allows to address int cell = array[INDEX(2,3)]; in stead of array[2+NBR_ASSET_CLASSES*3]; (using a one dimensional array to store a two dimnesional one!)*

Functions

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

Variables

- Cgicc `cgi`
- string `error_message` = ""
global message variables
- string `std_message` = ""
- config `oConfig`
mysql connection mysql connector from dev.mysql.com
- unordered_map< string, int > `goal_type_s2i`
Global lists.
- unordered_map< int, string > `goal_type_i2s`
- `html_helper hh`
own c++ classes
- `db_helper db`
- `cls_currency oCurrency`
- `asset_class oAssetClass`
- `asset oAsset`
- `goal oGoal`
- `cash_flow oCF`
- `investor_ui oInvestor`

8.23.1 Macro Definition Documentation

8.23.1.1 #define ALPHA_LOWER_LIMIT 0.1

the lower limit for all alpha's to be used

Definition at line 70 of file `efp.cpp`.

8.23.1.2 #define ALPHA_PLOT_HIGH 0.9

the high alpha to plot

Definition at line 72 of file `efp.cpp`.

8.23.1.3 #define ALPHA_PLOT_LOW 0.1

the low alpha value to plot

Definition at line 71 of file `efp.cpp`.

8.23.1.4 #define ALPHA_UPPER_LIMIT 0.4

the upper limit for all alpha's to be used

Definition at line 69 of file [efp.cpp](#).

8.23.1.5 #define CURR_LEN 14

the maximum length of an amount in number of digits (eg. 14)

Definition at line 79 of file [efp.cpp](#).

8.23.1.6 #define DEFAULT_SCALE 3

the default value to use for preferences (on a scale from 1 to 5)

Definition at line 89 of file [efp.cpp](#).

8.23.1.7 #define INDEX(x, y) ((y-1)+((NBR_ASSET_CLASSES)*(x-1)))

this macro allows to address int cell = array[INDEX(2,3)]; in stead of array[2+NBR_ASSET_CLASSES*3]; (using a one dimensional array to store a two dimensional one!)

Definition at line 100 of file [efp.cpp](#).

8.23.1.8 #define INPUT_WIDTH_FLOAT 5

the width of an input field for a float

Definition at line 84 of file [efp.cpp](#).

8.23.1.9 #define INPUT_WIDTH_INT 2

the width of an input field for an int

Definition at line 85 of file [efp.cpp](#).

8.23.1.10 #define INPUT_WIDTH_TEXT 15

the width of input-text field in all forms

Definition at line 83 of file [efp.cpp](#).

8.23.1.11 #define MAX_AGE 150

the maximum age allowed in the goals

Definition at line 66 of file [efp.cpp](#).

8.23.1.12 #define MAX_ALPHA 0.25

= (1 - confidence_level) for the least important goal

Definition at line 68 of file [efp.cpp](#).

8.23.1.13 #define MAX_MNTHS_2_SIMULATE 600

the maximum number of months that we allow to visualize a simulation

Definition at line 92 of file [efp.cpp](#).

8.23.1.14 #define MAX_MONTHS_TO_SAFEST 3

the maximum number of months for realization_monthNbr to be forced to the safest portfolio

Definition at line 87 of file [efp.cpp](#).

8.23.1.15 #define MAX_SCALE 5

the maximum value for the scale (5) (on a scale from 1 to 5)

Definition at line 90 of file [efp.cpp](#).

8.23.1.16 #define MIN_ALPHA 0.075

= (1 - confidence_level) for the most important goal

Definition at line 67 of file [efp.cpp](#).

8.23.1.17 #define MIN_MNTHS_2_SIMULATE 1

the maximum number of months that we allow to visualize a simulation

Definition at line 93 of file [efp.cpp](#).

8.23.1.18 #define MIN_SCALE 1

the minimum of the scale (1) (on a scale from 1 to 5)

Definition at line 91 of file [efp.cpp](#).

8.23.1.19 #define MIN_YEARS_TO_SIMULATE 2

the minimal number of years that can be simulated with this tool

Definition at line 76 of file [efp.cpp](#).

8.23.1.20 #define NBR_ASSET_CLASSES 10

the number of asset classes (not dynamically updated!)

Definition at line 97 of file [efp.cpp](#).

8.23.1.21 #define NBR_ITERATIONS 20

number of iterations in goalseek procedures

Definition at line 77 of file [efp.cpp](#).

8.23.1.22 #define NBR_RIGHT_MENU 3

the number of menu items floated to the right

Definition at line 82 of file [efp.cpp](#).

8.23.1.23 #define NBR_SCALE 5

the number of items in a ordinal scale (labels are in t_scale125i2s)

Definition at line 88 of file [efp.cpp](#).

8.23.1.24 #define NBR_SUB_TABS 3

maximum number of tabs used in the sub-menu

Definition at line 81 of file [efp.cpp](#).

8.23.1.25 #define NBR_TABS 7

number of tabs in the main menu of the software

Definition at line 80 of file [efp.cpp](#).

8.23.1.26 #define PRECISION 0.000001

equals roundup ($1 / 2^{\text{NBR_ITERATIONS}}$)

Definition at line 78 of file [efp.cpp](#).

8.23.1.27 #define RAINY_DAY_MONTH_NBR 3

the artificial horizon in months for the "rainy day goal"

Definition at line 73 of file [efp.cpp](#).

8.23.1.28 #define RETIREMENT_AGE 65

the standard retirement age (applied to everyone!)

Definition at line 74 of file [efp.cpp](#).

8.23.1.29 #define SECONDS_IN_MONTH 2629800

Marco's:

the number of seconds in the average month = $60 * 60 * 24 * 365.25 / 12$

Definition at line 65 of file [efp.cpp](#).

8.23.1.30 #define SIMULATE_TILL_DEFAULT 90

the standard age to stop the simulation

Definition at line 75 of file [efp.cpp](#).

8.23.2 Function Documentation

8.23.2.1 `int main (int argc, char ** argv)`

main < the investor is load whenever it is available (not only for thre screens that really need it -> so full name will be available for example within the disclaimers

- < allows for non-logged in people to see the inventory
- < personalisation
- < personalisation ///< save personalisation
- < personalisation
- < show edit form
- < execute the edit (when save is pressed)
- < show the help screen to add goals in a more supported way
- < save the relevant goals and show the list of goals
- < show the list of investment goals
- < add one goal
- < edit one goal
- < save one goal
- < delete one goal
- < show the list of investment cash_flows
- < add one [cash_flow](#)
- < edit one [cash_flow](#)
- < save one [cash_flow](#)
- < delete one [cash_flow](#)
- < show feedback
- < simulation
- < [follow_up](#) dashboard
- < edit the account information
- < edit the account information
- < do this after the customer has the screen displayed

Definition at line [218](#) of file [efp.cpp](#).

8.23.3 Variable Documentation

8.23.3.1 `Cgicc cgi`

Definition at line [129](#) of file [efp.cpp](#).

8.23.3.2 `db_helper db`

Definition at line [183](#) of file [efp.cpp](#).

8.23.3.3 string error_message = ""

global message variables

Definition at line 132 of file [efp.cpp](#).

8.23.3.4 unordered_map<int, string> goal_type_i2s

Initial value:

```
=  
{  
  {0, "unallocated"},  
  {1, "amount@date"},  
  {2, "income from/to"},  
  {3, "rainy day savings"}  
}
```

Definition at line 168 of file [efp.cpp](#).

8.23.3.5 unordered_map<string, int> goal_type_s2i

Initial value:

```
=  
{  
  {"unallocated", 0},  
  {"amount@date", 1},  
  {"income from/to", 2},  
  {"rainy day savings", 3},  
  {"amount asap", 4}  
}
```

Global lists.

Definition at line 160 of file [efp.cpp](#).

8.23.3.6 html_helper hh

own c++ classes

Definition at line 180 of file [efp.cpp](#).

8.23.3.7 asset oAsset

Definition at line 193 of file [efp.cpp](#).

8.23.3.8 asset_class oAssetClass

Definition at line 190 of file [efp.cpp](#).

8.23.3.9 cash_flow oCF

Definition at line 200 of file [efp.cpp](#).

8.23.3.10 config oConfig

mysql connection mysql connector from dev.mysql.com

< used in global_functions.h::phi(x) own include c++ files

Definition at line 149 of file [efp.cpp](#).

8.23.3.11 cls_currency oCurrency

Definition at line 187 of file [efp.cpp](#).

8.23.3.12 goal oGoal

Definition at line 198 of file [efp.cpp](#).

8.23.3.13 investor_ui oInvestor

Definition at line 211 of file [efp.cpp](#).

8.23.3.14 string std_message = ""

Definition at line 133 of file [efp.cpp](#).

8.24 efp.cpp

```

00001
00063 //#define LEND_RATE 0.1           ///< the rate at which people can lend money expressed as a percentage
      per year (eg. 0.1 = 10%)
00065 #define SECONDS_IN_MONTH 2629800
00066 #define MAX_AGE 150
00067 #define MIN_ALPHA 0.075
00068 #define MAX_ALPHA 0.25
00069 #define ALPHA_UPPER_LIMIT 0.4
00070 #define ALPHA_LOWER_LIMIT 0.1
00071 #define ALPHA_PLOT_LOW 0.1
00072 #define ALPHA_PLOT_HIGH 0.9
00073 #define RAINY_DAY_MONTH_NBR 3
00074 #define RETIREMENT_AGE 65
00075 #define SIMULATE_TILL_DEFAULT 90
00076 #define MIN_YEARS_TO_SIMULATE 2
00077 #define NBR_ITERATIONS 20
00078 #define PRECISSION 0.000001
00079 #define CURR_LEN 14
00080 #define NBR_TABS 7
00081 #define NBR_SUB_TABS 3
00082 #define NBR_RIGHT_MENU 3
00083 #define INPUT_WIDTH_TEXT 15
00084 #define INPUT_WIDTH_FLOAT 5
00085 #define INPUT_WIDTH_INT 2
00086 //define NBR_COLS_SP
00087 #define MAX_MONTHS_TO_SAFEST 3
00088 #define NBR_SCALE 5
00089 #define DEFAULT_SCALE 3
00090 #define MAX_SCALE 5
00091 #define MIN_SCALE 1
00092 #define MAX_MNTHS_2_SIMULATE 600
00093 #define MIN_MNTHS_2_SIMULATE 1
00094 //#define SHOW_Vexp_not_Vmed
00095
00096 // TODO find automatically:
00097 #define NBR_ASSET_CLASSES 10
00098 // uses previous
00099 //#define INDEX(x,y) ((x-1)+((NBR_ASSET_CLASSES)*(y-1))) ///< this macro allows to address int cell =
      array[INDEX(2,3)]; in stead of array[2+NBR_ASSET_CLASSES*3]; (using a one dimensional array to store a two
      dimnesional one!)
00100 #define INDEX(x,y) ((y-1)+((NBR_ASSET_CLASSES)*(x-1)))
00101 /*#define INDEX(x,y) ((x)+((NBR_ASSET_CLASSES)*(y))) ///< this macro allows to address int cell =
      array[INDEX(2,3)]; in stead of array[2+NBR_ASSET_CLASSES*3]; (using a one dimensional array to store a two

```

```

        dimnesional one!)*/*
00102
00103
00104
00105 // #define DEBUG
00106
00107
00108 #include <iostream>
00109 #include <string>
00110 #include <ctime>
00111 #include <vector>
00112 #include <map>
00113 #include <unordered_map>
00114 #include <math.h>
00115 #include <locale>
00116 #include <iomanip>          // used in the curr_format() function
00117 #include <errno.h>
00118
00119 #include "normdist.h" // has to be loaded before cgicc, conflicts with typedef class
        cgicc::HTMLBooleanElement<cgicc::aTag> cgicc::a
00120
00121 #include <cgicc/Cgicc.h>
00122 #include <cgicc/HTTPHTMLHeader.h>
00123 #include <cgicc/HTMLClasses.h>
00124 #include <cppconn/prepared_statement.h>
00125
00126 using namespace std;
00127 using namespace cgicc;
00128
00129 Cgicc cgi;
00130
00132 string error_message = "";
00133 string std_message = "";
00134 #ifdef DEBUG
00135 std::string debug_info = "";
00136 #endif
00137
00139 #include <mysql_connection.h>
00140 #include <driver.h>
00141 #include <exception.h>
00142 #include <resultset.h>
00143 #include <statement.h>
00144 #include <string.h>
00145 #include <cmath>
00146
00148 #include "config.class.cpp"
00149 config oConfig;
00150
00151 #include "t_personalize_Eng_UK.h"
00152 #include "texts_Eng_UK.h"
00153 #include "t_disclaimer_Eng_UK.h"
00154 #include "global_functions.h"
00155
00156 #include "bootstrap.namespace.cpp"
00157 using namespace bootstrap;
00158
00160 unordered_map<string, int> goal_type_s2i =
00161 {
00162     {"unallocated", 0},
00163     {"amount@date", 1},
00164     {"income from/to", 2},
00165     {"rainy day savings", 3},
00166     {"amount asap", 4}
00167 };
00168 unordered_map<int, string> goal_type_i2s =
00169 {
00170     {0, "unallocated"},
00171     {1, "amount@date"},
00172     {2, "income from/to"},
00173     {3, "rainy day savings"}
00174 };
00175
00176
00178 // #include "breadcrumb.struct.cpp"
00179 #include "html_helper.class.cpp"
00180 html_helper hh;
00181
00182 #include "db_helper.class.cpp"
00183 db_helper db;
00184
00185 #include "frequency.class.cpp"
00186 #include "currency.class.cpp"
00187 cls_currency oCurrency;
00188
00189 #include "asset_class.class.cpp"
00190 asset_class oAssetClass;
00191

```

```

00192 #include "asset.class.cpp"
00193 asset oAsset;
00194
00195 #include "column.class.cpp"
00196 #include "bona.class.cpp"
00197 #include "bona_goal.class.cpp"
00198 goal oGoal;
00199 #include "bona_cf.class.cpp"
00200 cash_flow oCF;
00201
00202 #include "portfolio.class.cpp"
00203 #include "market.class.cpp"
00204 #include "investor.class.cpp" // requires market.class
00205 #include "aGOAL.struct.cpp"
00206 #include "investment_problem.class.cpp" // requires investor.class and
    portfolio.class
00207 #include "simulation.class.cpp" // requires investment_problem.class
00208 #include "follow_up.class.cpp" // requires simulation.class
00209 #include "investor_ui.class.cpp" // requires investor.class
00210
00211 investor_ui oInvestor;
00212
00218 int main (int argc, char **argv)
00219 {
00220     oInvestor.set_investor_id_fromEnv(); // prune is_logged_in, get cookie,
    then check
00221                                     // if is_logged_in then the investor_id is set (otherwise it is 0)
00222     if (oInvestor.investor_id != 0) oInvestor.
    load_from_db(to_string(oInvestor.investor_id));
00223
00224     string action = cgi("a");
00225
00226     /*
00227     // TEST
00228     action = "rERvol";
00229     oInvestor.investor_id = 1;
00230     // END TEST
00231     */
00232
00233     if (action=="h" || action=="") oInvestor.home_screen();
00234     else if (action=="rf") oInvestor.register_form();
00235     else if (action=="re") oInvestor.register_exec();
00236     else if (action=="lf") oInvestor.login_form();
00237     else if (action=="le") oInvestor.login_exec();
00238     else if (action=="io") oInvestor.logout();
00239     else if (action=="d") oInvestor.show_disclaimer();
00240     else if (action=="gh") oInvestor.goal_help_form();
00241     else // so for al other action codes check if the user is allowed to do it
00242     {
00243         if (oInvestor.investor_id == 0)
00244         {
00245             error_message = error_message + t_errMsg["loginFirst"];
00246             oInvestor.home_screen();
00247         }
00248         else
00249         {
00250             if (action=="if") oInvestor.show_inventory_form();
00251             else if (action=="p") oInvestor.personalize_screen();
00252             else if (action=="ps") oInvestor.personalize_save();
00253             else if ((action=="rERvol" || (action=="rER") || (action=="rvol" || (action=="rcorr"))
    oInvestor.personalize_reload(action);
00254             else if (action=="aa") oInvestor.asset_add();
00255             else if (action=="ae") oInvestor.asset_edit();
00256             else if (action=="as") oInvestor.asset_save();
00257             else if (action=="ad") oInvestor.asset_delete();
00258             else if (action=="gh") oInvestor.goal_help_form();
00259             else if (action=="ghs") oInvestor.goal_help_save();
00260             else if (action=="gl") oInvestor.goal_list();
00261             else if (action=="ga") oInvestor.goal_add();
00262             else if (action=="ge") oInvestor.goal_edit();
00263             else if (action=="gs") oInvestor.goal_save();
00264             else if (action=="gd") oInvestor.goal_delete();
00265             else if (action=="cl") oInvestor.cash_flow_list();
00266             else if (action=="ca") oInvestor.cash_flow_add();
00267             else if (action=="ce") oInvestor.cash_flow_edit();
00268             else if (action=="cs") oInvestor.cash_flow_save();
00269             else if (action=="cd") oInvestor.cash_flow_delete();
00270             else if (action=="f") oInvestor.feedback();
00271             else if (action=="s") oInvestor.show_simulation();
00272             else if (action=="fu") oInvestor.show_follow_up();
00273             else if (action=="ie") oInvestor.account_form();
00274             else if (action=="is") oInvestor.account_save();
00275             else {oInvestor.home_screen();}
00276             oInvestor.update_last_activity_at();
00277         }
00278     }
00279     return 0; // to avoid Apache errors

```

```
00280 }
```

8.25 follow_up.class.cpp File Reference

Classes

- class [follow_up](#)

8.26 follow_up.class.cpp

```
00001
00007 class follow_up : public simulation
00008 {
00009 public:
00010     follow_up(int investor);
00011     void parse_dashboard();
00012 };
00013
00017 follow_up::follow_up(int investor)
00018     : simulation(investor)
00019 {
00020     //cout << ""; ///< this constructor does nothing more than launching the previous class' constructor
00021 }
00022
00026 void follow_up::parse_dashboard()
00027 {
00028     int g;
00029     cout << "<div class=\"alert alert-info\">";
00030     cout << "<h3>" << t_dashboard << " at " << mNbr2dateStr(
00031         get_nbrMonths2simulate()) << "</h3>";
00032     cout << "<table border=\"1\" class=\"followupDash\">";
00033     cout << "<tr><th>Nr.</th><th>Priority</th><th>Description</th><th>Previous color at " <<
00034         mNbr2dateStr(0) << "</th>";
00035     cout << "<tr><th>New Color at " << mNbr2dateStr(get_nbrMonths2simulate()) <
00036         < " or at goal-realization</th></tr>";
00037     for (g = 1; g <= get_nbr_goals(); g++)
00038     {
00039         cout << "<tr>";
00040         cout << " <td>" << g << "</td>";
00041         cout << " <td>" << this->goalZ[g].priority << "</td>";
00042         cout << " <td>" << this->goalZ[g].description << "</td>";
00043         cout << " <td class=\"" << hh.get_box_color_class(this->
00044             goalZ[g].color) << "\">" << hh.get_glyphicon(this->goalZ[g].color) << "</td>";
00045         cout << " <td class=\"" << hh.get_box_color_class(this->
00046             goalZ[g].color_followup) << "\">" << hh.get_glyphicon(this->
00047             goalZ[g].color_followup) << "</td>";
00048         cout << "</tr>";
00049     }
00050     cout << "</table>" << "</div>" << endl;
00051 }
00052
00056
00057
```

8.27 frequency.class.cpp File Reference

Classes

- class [cls_frequency](#)

8.28 frequency.class.cpp

```
00001 class cls_frequency
00002 {
00003     public:
00004
00005     string frequency_id;
00006     float multiplier;
00007
```

```

00008     bool set_freq(string the_freq);
00009     string get_freq_label();
00010 };
00011
00012
00018 bool cls_frequency::set_freq(string the_freq)
00019 {
00020     string s;
00021     s = "SELECT COUNT(*) AS EXIST_NBR from " + oConfig.tbl_prefix + "_frequencies WHERE
frequency_id = '" + the_freq + "'";
00022     db.res = db.stmt->executeQuery(s);
00023     while (db.res->next())
00024     {
00025         if (db.res->getString("EXIST_NBR") == "1")
00026         {
00027             this->frequency_id = the_freq;
00028             return true;
00029         }
00030         else
00031         {
00032             return false;
00033         }
00034     }
00035     return false; // table not found ;- )
00036 }
00037
00038
00044 string cls_frequency::get_freq_label()
00045 {
00046     string s;
00047     s = "SELECT * FROM " + oConfig.tbl_prefix + "_frequencies WHERE frequency_id = '" +
this->frequency_id + "'";
00048     db.res = db.stmt->executeQuery(s);
00049     while (db.res->next())
00050     {
00051         return db.res->getString("frequency_name");
00052     }
00053     return "";
00054 }

```

8.29 global_functions.h File Reference

Functions

- tm [get_tm](#) (int monthNbr=0)
 - global functions*
- int [tm2Mnbr](#) (tm tmDate)
- string [tm2DateStr](#) (tm tmDate)
- void [dateStr2tm](#) (string dStr, tm *target_tm)
- int [dateStr2Mnbr](#) (string dStr)
- string [mNbr2dateStr](#) (int monthNbr=0)
- bool [is_valid_dateStr](#) (string theDate)
- float [flSum](#) (std::map< int, float > *theMap, int N)
- void [thousand_separator](#) (float x, char *s)
- string [currFormat](#) (float x, string theCurr)
- string [curr_format](#) (float dv, string the_curr="")
- string [date_format_tm](#) (tm *the_tm)
- string [date_format_dateStr](#) (string s)
- string [round2str](#) (float x, int nbr_digits=0)
- string [months2yms](#) (int m)
- string [add0](#) (int x, int n=1)

Variables

- const float [Pi](#) = 3.141593
 - global variables*

8.29.1 Function Documentation

8.29.1.1 string add0 (int x, int n = 1) [inline]

add0

adds a zero before a number if it is smaller than the desired minimum length eg

x = the number to be formatted n = the minimal number of digits to make

Definition at line 311 of file [global_functions.h](#).

8.29.1.2 string curr_format (float dv, string the_curr = " ")

curr_format

inspired on <http://www.arachnoid.com/cpptutor/student3.html>

needs: <iostream>, <iomanip>, <string>

Definition at line 190 of file [global_functions.h](#).

8.29.1.3 string currFormat (float x, string theCurr) [inline]

[currFormat\(\)](#)

this function converts a given number into currency format followed by EUR

Definition at line 171 of file [global_functions.h](#).

8.29.1.4 string date_format_dateStr (string s) [inline]

date_format_dateStr

output the date in a standard format

Definition at line 246 of file [global_functions.h](#).

8.29.1.5 string date_format_tm (tm * the_tm) [inline]

date_format_tm

output the date in a standard format

Definition at line 236 of file [global_functions.h](#).

8.29.1.6 int dateStr2Mnbr (string dStr) [inline]

dateStr2Mnbr

converts a date string of the form YYYY-MM-DD to a month number

Definition at line 73 of file [global_functions.h](#).

8.29.1.7 void dateStr2tm (string dStr, tm * target_tm) [inline]

dateStr2tm

converts a date string of the form YYYY-MM-DD into a tm

Definition at line 60 of file [global_functions.h](#).

8.29.1.8 `float flSum (std::map< int, float > * theMap, int N) [inline]`

returns the sum of element of a map which elements are numbered from 1 to N

Definition at line 107 of file [global_functions.h](#).

8.29.1.9 `tm get_tm (int monthNbr = 0) [inline]`

global functions

tm_now returns the date in tm-format for a given number of months in the future the default is 0 months in the future, hence "now"

Definition at line 11 of file [global_functions.h](#).

8.29.1.10 `bool is_valid_dateStr (string theDate) [inline]`

is_valid_dateStr checks is the string is a valid ISO date of the form YYYY-MM-DD

Definition at line 96 of file [global_functions.h](#).

8.29.1.11 `string mNbr2dateStr (int monthNbr = 0) [inline]`

mNbr2dateStr

converts a month number to a date-string

Definition at line 85 of file [global_functions.h](#).

8.29.1.12 `string months2yms (int m) [inline]`

parse months2yms

parses a number of months into a year-month string which is something like "1 year and 3 months"

Definition at line 282 of file [global_functions.h](#).

8.29.1.13 `string round2str (float x, int nbr_digits = 0) [inline]`

round2str

rounds a float to a certain number of digits after the decimal and returns it as a string

Definition at line 258 of file [global_functions.h](#).

8.29.1.14 `void thousand_separator (float x, char * s)`

thousand_separator

returns a string with thousand separator added for the integer part of the amount

Definition at line 119 of file [global_functions.h](#).

8.29.1.15 `string tm2DateStr (tm tmDate) [inline]`

Definition at line 51 of file [global_functions.h](#).

8.29.1.16 int tm2Mnbr (tm tmDate) [inline]

tm2Mnbr converts a date into a number of months between now and the date

Definition at line 40 of file [global_functions.h](#).

8.29.2 Variable Documentation

8.29.2.1 const float Pi = 3.141593

global variables

Definition at line 2 of file [global_functions.h](#).

8.30 global_functions.h

```

00001 const float Pi = 3.141593;
00003
00005
00011 inline tm get_tm(int monthNbr = 0)
00012 {
00013     time_t t = time(0);    // get time now
00014     struct tm *tm_date;
00015     tm_date = localtime(&t);
00016     tm_date->tm_year += floor(monthNbr / 12);
00017     monthNbr = monthNbr % 12;
00018     while (monthNbr > 0)
00019     {
00020         if (tm_date->tm_mon == 11)
00021         {
00022             tm_date->tm_mon = 0;
00023             tm_date->tm_year++;
00024         }
00025         else
00026         {
00027             tm_date->tm_mon++;
00028         }
00029         monthNbr--;
00030     }
00031     mktime(tm_date);    //normalize the result
00032
00033     return *tm_date;
00034 }
00035
00040 inline int tm2Mnbr(tm tmDate) {
00041     double x;
00042     time_t now;
00043     time(&now);
00044     x = difftime(mktime(&tmDate), now);
00045     x = x / SECONDS_IN_MONTH + 0.5;    //x = x / 60 / 60 / 24 / 30.4375 + 0.5;
00046     return (int) x;
00047 }
00048
00049 /***** tm2DateStr *****/
00050 // returns a string YYYY-MM-DD
00051 inline string tm2DateStr(tm tmDate)
00052 {
00053     return (to_string(tmDate.tm_year + 1900) + "-" + to_string(tmDate.tm_mon + 1) + "-" + to_string(tmDate.
tm_mday)); }
00054
00060 inline void dateStr2tm(string dStr, tm * target_tm)
00061 {
00062     struct tm *xDate;
00063     xDate = target_tm;
00064     memset(xDate, 0, sizeof(struct tm));
00065     strptime(dStr.c_str(), "%Y-%m-%d", xDate);
00066 }
00067
00073 inline int dateStr2Mnbr(string dStr)
00074 {
00075     struct tm xDate;
00076     dateStr2tm(dStr, &xDate);
00077     return tm2Mnbr(xDate);
00078 }
00079
00085 inline string mNbr2dateStr(int monthNbr = 0)
00086 {

```

```

00087 struct tm tm_date;
00088 tm_date = get_tm(monthNbr);
00089 return tm2DateStr(tm_date);
00090 }
00091
00096 inline bool is_valid_dateStr(string theDate) {
00097     bool rc = true;
00098     struct tm tmDate;
00099     // if (theDate.length() != 10) rc = false;
00100     if (!strptime(theDate.c_str(), "%Y-%m-%d", &tmDate)) rc = false;
00101     return rc;
00102 }
00103
00107 inline float flSum(std::map<int, float>* theMap, int N) {
00108     int k;
00109     float theSum = 0;
00110     for (k = 0; k <= N; k++) {theSum += (*theMap)[k];}
00111     return theSum;
00112 }
00113
00119 void thousand_separator(float x, char* s) {
00120     int j, i = 0, k = 0, ix = (x >= 0) ? (int)floor(x) : (int)ceil(x);
00121     //char s[50];
00122     if( ix < 0)//test if the number is negative
00123     {
00124         strcpy(s, "-");
00125         ix *= -1;
00126     }
00127     else s[0] = '0'; //strcpy(s, "");
00128     int temp = ix;
00129     int p = 1;
00130     while( temp > 0) //counting number of digits
00131     {
00132         temp /= 10;
00133         p *= 10;
00134         i++;
00135     }
00136     j = i % 3;
00137     p /= 10;
00138
00139     while( i > 0)//display integer number with 1000 separator
00140     {
00141         s[k] = char ((ix/p) +'0');
00142         ix %= p;
00143         p /= 10;
00144         i--;
00145         k++;
00146         j--;
00147         if ((k % 3 == 0 && i > 0) || (j == 0 && i > 2) )
00148         {
00149             strcat(s, ",");
00150             k = 0;
00151         }
00152     }
00153 }
00154
00160 /*inline string currFormat(float x, string theCurr) {
00161     char s[50], s2[30];
00162     thousand_separator(x,s2);
00163     strcpy(s, s2);
00164     sprintf(s2,"%%.02f", x - floor(x));
00165     strcat(s,s2);
00166     // sprintf(s,"%%.02f", x);
00167     if (theCurr == "EUR") strcat(s, " &#8364;");
00168     else strcat(s,theCurr.c_str());
00169     return s;
00170 }*/
00171 inline string currFormat(float x, string theCurr) {
00172     char s[50];
00173     sprintf(s,"%%.02f", x);
00174     if (theCurr == "EUR") strcat(s, " &#8364;");
00175     else strcat(s,theCurr.c_str());
00176     return s;
00177 }
00178
00179
00188 //using namespace std;
00189
00190 string curr_format(float dv, string the_curr = "")
00191 {
00192     // const int width = CURR_LEN;
00193     const string radix = ".";
00194     const string thousands = ",";
00195     string the_sign, unit;
00196     if (the_curr == "EUR") unit = " &#8364;";
00197     else if (the_curr == "USD") unit = " &#36;";
00198     else if (the_curr == "GBP") unit = " &#163;";

```

```

00199     else if (the_curr == "JPY") unit = " ¥165;";
00200     else                               unit = the_curr;
00201 if (dv < 0)
00202 {
00203     dv = - dv;
00204     the_sign = " -";
00205 }
00206 else
00207 {
00208     the_sign = "";
00209 }
00210 unsigned long v = (unsigned long) ((dv * 100.0) + .5);
00211 string fmt,digit;
00212 int i = -2;
00213 do {
00214     if(i == 0) {
00215         fmt = radix + fmt;
00216     }
00217     if((i > 0) && !(i % 3)) {
00218         fmt = thousands + fmt;
00219     }
00220     digit = (v % 10) + '0';
00221     fmt = digit + fmt;
00222     v /= 10;
00223     i++;
00224 }
00225 while((v) || (i < 1));
00226 //cout << unit << setw(width) << fmt.c_str() << endl;
00227 return unit + the_sign + fmt;
00228 }
00229
00230
00236 inline string date_format_tm(tm * the_tm)
00237 {
00238     return to_string(the_tm->tm_year + 1900) + "-" + to_string(the_tm->tm_mon + 1) + "-" + to_string(the_tm->
tm_mday);
00239 }
00240
00246 inline string date_format_dateStr(string s)
00247 {
00248     struct tm the_tm;
00249     dateStr2tm(s, &the_tm);
00250     return to_string(the_tm.tm_year + 1900) + "-" + to_string(the_tm.tm_mon + 1) + "-" + to_string(the_tm.
tm_mday);
00251 }
00252
00258 inline string round2str(float x, int nbr_digits = 0)
00259 {
00260     string s = "";
00261     if (x < 0) s = "-" + s;
00262     x = abs(x);
00263     if (nbr_digits > 0)
00264     {
00265         s = s + to_string((int) x);
00266         int x_dec = round((x - (int)x) * pow(10, nbr_digits));
00267         s = s + "." + to_string(x_dec);
00268     }
00269     else
00270     {
00271         s = s + to_string((int)(x + 0.5));
00272     }
00273     return s;
00274 }
00275
00282 inline string months2yms(int m)
00283 {
00284     string s;
00285     int yrs = (int)(m /12);
00286     if (yrs >= 2)         s = to_string(yrs) + " " + t_years;
00287     else if (yrs >= 1)   s = to_string(yrs) + " " + t_year;
00288     else                   s = "";
00289
00290     m = m - yrs * 12;
00291
00292     if ((yrs != 0) && (m > 1)) s = s + " " + t_and + " ";
00293
00294     if (m >= 2)         s = s + to_string(m) + " " + t_months;
00295     else if (m >= 1)   s = s + to_string(m) + " " + t_month;
00296
00297     if ((yrs == 0) && (m == 0)) s = "0 " + t_months;
00298
00299     return s;
00300 }
00301
00311 inline string add0(int x, int n = 1)
00312 {
00313     string s = "";

```

```

00314     int    digits;
00315     for (digits = 1; digits <= n; digits++)
00316     {
00317         if (x < pow(10,digits)) s = s + "0";
00318     }
00319     s = s + to_string(x);
00320     return s;
00321 }

```

8.31 html_helper.class.cpp File Reference

Classes

- class [html_helper](#)

8.32 html_helper.class.cpp

```

00001 class html_helper
00002 {
00003 public:
00004     html_helper();
00005     void footer(string tab_name, string extraInfo = "");
00006     void show_home();
00007     void header(int iid=0, bool addSavingsPlan = false, bool addPlots = false);
00008     string get_string(const std::string& varName);
00009     const string c_salt = "k35f9vfgDtjTel";
00010     void show_toolBar(string tab_name, string usrName = "");
00011     void next_step(int stepNbr, int subStepNbr);
00012     // string dropDown_currency();
00013     void aside(const string& textLine);
00014     void cout_box_color(string the_color);
00015     string get_box_color_class(string the_color);
00016     string get_glyphicon(string the_color);
00017
00018     //-- the colors used in the graphs: red,green black, etc.:
00019     const string seriesColorsGoalPlot = "[ 'rgba(160, 0, 0, 0.6)', 'rgba(0, 0, 160,
0.6)', 'rgba(0, 160, 0, 0.6)', \"#000\", \"#663\", \"#958c12\", \"#953579\", \"#4b5de4\", \"#d8b83f\", \"#ff5800\", \"#0085cc\" ]";
00020     // const string seriesColorsGoalPlot = "[ \"#ff3333\", \"#33f\", \"#3a3\", \"#000\", \"#663\", \"#958c12\", \"#953579\", \"#4b5de4\", \"#d8b83f\", \"#ff5800\", \"#0085cc\" ]";
00021
00022     //-- the asset-class-colors, optimized for 1 cash, 3 bonds, 3 equity and 3 others:
00023     const string assetClassColors = "[ '#999', '#0c3', '#396', '#0c9', '#3cf', '#69f', '#96f', '#f93', '#fc3', '#f63', '#33f' ]";
00024
00025     string terms_of_use();
00026     string privacy_policy();
00027     string cookie_policy();
00028     string collapse_open(string parent_id, string title, int nbr, bool add_triangle = true, bool
is_open = true, string xtra_body_class = "");
00029     string collapse_close(string parent_id = "", int nbr = 0, bool add_triangle = true);
00030
00031 private:
00032     void header_menu_bar();
00033     void parse_date_time();
00034     void show_args(int argc, char **argv);
00035     void parse_breadcrumb(int tab, int sub_tab);
00036     void parse_title(int tab, int sub_tab);
00037
00038
00039     const std::map<int, string> c_photos_phdb = {{1, "phdb/ZM5K1325_01.jpg"}, {2, "
phdb/ZM5K1312_01b.jpg"},
00040         {3, "phdb/ZM5K1302_01c.jpg"}, {4, "phdb/ZM5K1313_01c.jpg"}, {5, "fvdspieg1.jpg"}, {6, "fvdspieg2.jpg"}};
00041     const string c_photos_phdb_dir = "/img/";
00042     const string c_content_header = "Content-type:text/html\r\n\r\n";
00043     const string c_doc_type = "<!DOCTYPE html>";
00044
00045     std::map<int, string> scale125i2s;
00046     typedef std::map<string, string> STRMAP;
00047     typedef std::map<int, STRMAP > STEPDEF;
00048     STEPDEF steps;
00049     std::map<int, STEPDEF> sub_steps;
00050     int get_prev_step(int theStep, int theSubStep);
00051     int get_prev_sub_step(int theStep, int theSubStep);
00052     int get_next_step(int theStep, int theSubStep);
00053     int get_next_sub_step(int theStep, int theSubStep);
00054     bool sub_step_exists(int theStep, int theSubStep);

```

```

00055     std::map<string, int> step_nbrs;
00056     std::map<string, int> sub_step_nbrs;
00057     int get_step_nbr(string step_name);
00058     int get_sub_step_nbr(string sub_step_name);
00059     /*
00060     const string c_tab_names[NBR_TABS] = {"Home","Login", "Goals", "Inventory", "Feedback", "Simulation",
"Follow Up"};
00061     const string c_tab_titles[NBR_TABS] = {
00062         "Welcome to " + oConfig.soft_name, // home
00063         "", //login
00064         "State <strong>YOUR</strong> personal life goals as clear as possible", //goals
00065         "Take your time to tell the software about your savings and future savings capacity", //inventory
00066         "Taking into account Your goals and Your savings capacity, this seems a reasonable plan", //feedback
00067         "Please see here what <strong>could</strong> happen: one of the millions of possible scenarios", //
simulation
00068         "Here you see one of the possible scenarios if you would not do anything between now and the given
moment" //followup
00069     };
00070     const string c_sub_tab_titles[NBR_TABS][NBR_SUB_TABS] = {
00071         {"", "", ""},
00072         {"Please log in if you registered before", "Register if you have no account yet", ""},
00073         {"", "", ""},
00074         {"List here Your EXISTING savings", "List here Your FUTURE savings capacity", ""},
00075         {"", "", ""},
00076         {"", "", ""},
00077         {"", "", ""}
00078     };
00079     const string c_tab_urls[NBR_TABS] = {
00080         oConfig.soft_url + "?a=h",
00081         oConfig.soft_url + "?a=rf",
00082         oConfig.soft_url + "?a=gl",
00083         oConfig.soft_url + "?a=if",
00084         oConfig.soft_url + "?a=f",
00085         oConfig.soft_url + "?a=s",
00086         oConfig.soft_url + "?a=fu"};
00087     const string c_sub_tab_names[NBR_TABS][NBR_SUB_TABS] = {
00088         {"", "", ""},
00089         {"Login", "Register", ""},
00090         {"", "", ""},
00091         {"Assets", "Cash Flows", ""},
00092         {"", "", ""},
00093         {"", "", ""},
00094         {"", "", ""}
00095     };
00096     const string c_sub_tab_urls[NBR_TABS][NBR_SUB_TABS] = {
00097         {"", "", ""},
00098         {oConfig.soft_url + "?a=lf", oConfig.soft_url + "?a=rf", ""},
00099         {"", "", ""},
00100         {oConfig.soft_url + "?a=if", oConfig.soft_url + "?a=cl", ""},
00101         {"", "", ""},
00102         {"", "", ""},
00103         {"", "", ""}};
00104
00105     */
00106     const string c_right_menu_names[NBR_RIGHT_MENU] = {"My Account", "
Disclaimers", "Log Out"};
00107     const string c_right_menu_urls[NBR_RIGHT_MENU] = {
00108         oConfig.soft_url + "?a=ie",
00109         oConfig.soft_url + "?a=d",
00110         oConfig.soft_url + "?a=io"
00111     };
00112     const string c_arrow = "&nbsp;<img src=\"" + oConfig.css_dir + "next.gif\" border=\""0
\" width=\""5\" height=\""9\" alt=\""\" class=\""noPadding\">&nbsp;";
00113     void show_social();
00114
00115     void navbar_start();
00116     void navbar_item(int k, string usr_name = "", string tab_name = "");
00117     void navbar_end();
00118     void navbar_box(int k, string usr_name = "");
00119
00120 };
00121
00125 html_helper::html_helper()
00126 {
00128
00130     int step = 1, sub_step = 1;
00131     steps[step]["name"] = "Home";
00132     steps[step]["url"] = oConfig.soft_url + "?a=h";
00133     steps[step]["title"] = t_titles["welcome"];
00134     steps[step]["must_login"] = "N";
00135     step_nbrs["home"] = step; sub_step_nbrs["home"] = 0;
00136
00138     /* step++;
00139     sub_step = 1;
00140     step_nbrs["login"] = step; sub_step_nbrs["login"] = sub_step;
00141     steps[step]["name"] = "Identification";
00142     steps[step]["url"] = oConfig.soft_url + "?a=rf";

```

```

00143     steps[step]["title"] = "";
00144     sub_steps[step][sub_step]["name"] = "Login";
00145     sub_steps[step][sub_step]["url"] = oConfig.soft_url + "?a=lf";
00146     sub_steps[step][sub_step]["title"] = ";//Log In";
00147     sub_step++;
00148     step_nbrs["register"] = step;     sub_step_nbrs["register"] = sub_step;
00149     sub_steps[step][sub_step]["name"] = "Register";
00150     sub_steps[step][sub_step]["url"] = oConfig.soft_url + "?a=rf";
00151     sub_steps[step][sub_step]["title"] = ";//Register";
00152     steps[step]["must_login"] = "N";*/
00153
00155     step++;
00156     step_nbrs["personalize"] = step;     sub_step_nbrs["personalize"] = 0;
00157     steps[step]["name"] = "Personalize";
00158     steps[step]["url"] = oConfig.soft_url + "?a=p";
00159     steps[step]["title"] = t_titles["personalize"];
00160     steps[step]["must_login"] = "Y";
00161
00163     step++;
00164     step_nbrs["goals"] = step;     sub_step_nbrs["goals"] = 0;
00165     steps[step]["name"] = "Goals";
00166     steps[step]["url"] = oConfig.soft_url + "?a=gl";
00167     steps[step]["title"] = t_titles["goal"];
00168     steps[step]["must_login"] = "Y";
00169
00171     step++;
00172     sub_step = 1;
00173     step_nbrs["assets"] = step;     sub_step_nbrs["assets"] = sub_step;
00174     steps[step]["name"] = "Inventory";
00175     steps[step]["url"] = oConfig.soft_url + "?a=if";
00176     steps[step]["title"] = t_titles["inventory"];
00177     sub_steps[step][sub_step]["name"] = "Assets";
00178     sub_steps[step][sub_step]["url"] = oConfig.soft_url + "?a=if";
00179     sub_steps[step][sub_step]["title"] = t_titles["asset"];
00180     sub_step++;
00181     step_nbrs["cfs"] = step;     sub_step_nbrs["cfs"] = sub_step;
00182     sub_steps[step][sub_step]["name"] = "Savings and expenses";
00183     sub_steps[step][sub_step]["url"] = oConfig.soft_url + "?a=cl";
00184     sub_steps[step][sub_step]["title"] = t_titles["cash_flow"];
00185     steps[step]["must_login"] = "Y";
00186
00188     step++;
00189     step_nbrs["feedback"] = step;     sub_step_nbrs["feedback"] = 0;
00190     steps[step]["name"] = "Feedback";
00191     steps[step]["url"] = oConfig.soft_url + "?a=f";
00192     steps[step]["title"] = t_titles["feedback"];
00193     steps[step]["must_login"] = "Y";
00194
00196     step++;
00197     step_nbrs["simulation"] = step;     sub_step_nbrs["simulation"] = 0;
00198     steps[step]["name"] = "Simulation";
00199     steps[step]["url"] = oConfig.soft_url + "?a=s";
00200     steps[step]["title"] = t_titles["simulation"];
00201     steps[step]["must_login"] = "Y";
00202
00204     step++;
00205     step_nbrs["followup"] = step;     sub_step_nbrs["followup"] = 0;
00206     steps[step]["name"] = "Follow-Up";
00207     steps[step]["url"] = oConfig.soft_url + "?a=fu";
00208     steps[step]["title"] = t_titles["follow_up"];
00209     steps[step]["must_login"] = "Y";
00210 }
00211
00215 inline int html_helper::get_prev_step(int theStep, int theSubStep)
00216 {
00217     if (theStep <= 1) return 0;
00218     if (theSubStep > 1)
00219     {
00220         return theStep;
00221     }
00222     else
00223     {
00224         return theStep - 1;
00225     }
00226 }
00230 inline int html_helper::get_next_step(int theStep, int theSubStep)
00231 {
00232     if (theStep <= 0) return 1; //erroneous input, so back to the safe home-screen
00233     if (theStep == 1) return 0; //at login, we do not show the next step (would not work)
00234     // if there is a next substep, then stay in this step
00235     auto outerIt = sub_steps.find(theStep);
00236     if (outerIt != sub_steps.end())
00237     {
00238         auto innerMapPtr = outerIt->second;
00239         const int KeyInner = theSubStep + 1;
00240         auto innerIt = innerMapPtr.find(KeyInner);
00241         if (innerIt != innerMapPtr.end()) return theStep;

```

```
00242     }
00243     if (steps.find(theStep+1) != steps.end()) return theStep + 1; //if there is a next step, then
return it
00244     return 0;
00245 }
00246
00250 inline int html_helper::get_next_sub_step(int theStep, int theSubStep)
00251 {
00252     auto outerIt = sub_steps.find(theStep);
00253     //if there are substeps
00254     if (outerIt != sub_steps.end())
00255     {
00256         auto innerMapPtr = outerIt->second;
00257         const int KeyInner = theSubStep+1;
00258         auto innerIt = innerMapPtr.find(KeyInner);
00259         if (innerIt != innerMapPtr.end())
00260         {
00261             return theSubStep+1;
00262         }
00263     }
00264     //so, now there are no substeps or there is no next one, so
00265     outerIt = sub_steps.find(theStep + 1); // go look in the next step
00266     if (outerIt != sub_steps.end())
00267     {
00268         auto innerMapPtr = outerIt->second;
00269         const int KeyInner = 1;
00270         auto innerIt = innerMapPtr.find(KeyInner);
00271         if (innerIt != innerMapPtr.end())
00272         {
00273             return 1;
00274         }
00275     }
00276     return 0;
00277 }
00284 inline int html_helper::get_prev_sub_step(int theStep, int theSubStep)
00285 {
00286     if (theSubStep > 1) return theSubStep - 1;
00287     if (theStep <= 1) return 0;
00288     // now we have to find the highest substep of the previous step
00289
00290
00291     auto outerIt = sub_steps.find(theStep - 1);
00292     if (outerIt != sub_steps.end())
00293     {
00294         auto innerMapPtr = outerIt->second;
00295         const int KeyInner = 1;
00296         auto innerIt = innerMapPtr.find(KeyInner);
00297         if (innerIt != innerMapPtr.end())
00298         {
00299             return KeyInner;
00300         }
00301     }
00302     return 0;
00303 }
00304
00308 inline bool html_helper::sub_step_exists(int theStep, int theSubStep)
00309 {
00310     auto outerIt = sub_steps.find(theStep);
00311     if (outerIt != sub_steps.end())
00312     {
00313         auto innerMapPtr = outerIt->second;
00314         const int KeyInner = theSubStep;
00315         auto innerIt = innerMapPtr.find(KeyInner);
00316         if (innerIt == innerMapPtr.end())
00317         {
00318             return false;
00319         }
00320         else
00321         {
00322             return true;
00323         }
00324     }
00325     else
00326     {
00327         return false;
00328     }
00329 }
00330
00334 int html_helper::get_step_nbr(string step_name)
00335 {
00336     if (step_name == "") return 0;
00337     try
00338     {
00339         return step_nbrs.at(step_name);
00340     }
00341     catch (std::out_of_range& e)
00342     {
```

```

00343 #ifdef DEBUG
00344     std::cerr << e.what() << std::endl;
00345 #endif
00346     cout << "error no. HH001";
00347 }
00348 return -1;
00349 }
00350
00354 int html_helper::get_sub_step_nbr(string sub_step_name)
00355 {
00356     if (sub_step_name == "") return 0;
00357     try
00358     {
00359         return sub_step_nbrs.at(sub_step_name);
00360     }
00361     catch (std::out_of_range& e)
00362     {
00363         #ifdef DEBUG
00364             std::cerr << e.what() << std::endl;
00365         #endif
00366         cout << "error no. HH002";
00367     }
00368     return -1;
00369 }
00370
00371
00375 inline string html_helper::collapse_open(string parent_id, string title, int nbr,
    bool add_triangle, bool is_open, string xtra_body_class)
00376 {
00377     string s, s_nbr = to_string(nbr), s_in = (is_open?"in":"";
00378     switch (oConfig.layout)
00379     {
00380         case '2':
00381             s = "\n<div class=\"panel panel-default\">";
00382             s = s + "\n<div class=\"brightgreenBox panel-heading\">";
00383             s = s + "<a data-toggle=\"collapse\" data-parent=\"#" + parent_id + "\" data-target=\""
#collapse" + s_nbr + "\">";
00384             s = s + title;
00385             if (add_triangle) s = s + "&nbsp;<span class=\"glyphicon glyphicon-triangle-bottom\"></span>";
00386             s = s + "</a>";
00387             s = s + "</div>";
00388             s = s + "\n<div id=\"collapse" + s_nbr + "\" class=\"panel-collapse collapse " + s_in + "\">";
00389             s = s + "\n<div class=\"panel-body " + xtra_body_class + "\">\n";
00390             break;
00391         case '1':
00392             default:
00393                 s = "<div clas=\"alert-info\">\n";
00394     }
00395     return s;
00396 }
00397
00401 inline string html_helper::collapse_close(string parent_id, int nbr, bool
    add_triangle)
00402 {
00403     string s = "";
00404     switch (oConfig.layout)
00405     {
00406         case '2' :
00407             if (parent_id != "")
00408             {
00409                 string s_nbr = to_string(nbr);
00410                 s = "<p><a data-toggle=\"collapse\" data-parent=\"#" + parent_id + "\" data-target=\"#"collapse" +
s_nbr + "\">";
00411                 s = s + t_collapse;
00412                 if (add_triangle) s = s + "&nbsp;<span class=\"glyphicon glyphicon-triangle-top\"></span>";
00413                 s = s + "</a></p>";
00414             }
00415             s = s + "</div></div></div>\n";
00416             break;
00417         case '1':
00418             default:
00419                 s = s + "</div>\n";
00420     }
00421     return s;
00422 }
00423
00424
00428 void html_helper::footer(string tab_name, string extraInfo)
00429 {
00430     int step = get_step_nbr(tab_name);
00431     int sub_step = get_sub_step_nbr(tab_name);
00432
00433     // cout << "<hr/>";
00434     if (extraInfo != "")
00435     {
00436         cout << endl;
00437         switch (oConfig.layout)

```

```

00438     {
00439         case '1' :
00440             cout << "<button class=\"btn btn-primary\" id=\"btnShow\">" <<
t_show_extra_info << "</button>&nbsp;";
00441             cout << "<button class=\"btn btn-primary\" id=\"btnHide\">" <<
t_hide_extra_info << "</button>";
00442             cout << "<div id=\"extraInfo\">" << t_extraInfo[extraInfo] << "</div>";
00443             break;
00444             case '2' :
00445                 bs_modal (t_show_extra_info + "&nbsp;<span class=\"glyphicon glyphicon-info-sign\"></span>"
, t_extraInfo[extraInfo]);
00446                 break;
00447             default :
00448                 cout << "<button class=\"btn btn-primary\" id=\"btnShow\">" << t_show_extra_info << "</button>&nbsp;";
";
00449                 cout << "<button class=\"btn btn-primary\" id=\"btnHide\">" << t_hide_extra_info << "</button>";
00450                 cout << "<div id=\"extraInfo\">" << t_extraInfo[extraInfo] << "</div>";
00451                 break;
00452             }
00453             //cout << "<br style=\"clear:left;\"/>";
00454         }
00455         next_step(step, sub_step);
00456         parse_breadcrumb(step, sub_step);
00457         cout << "</div><div class=\"alert alert-warning\">" << t_warning << "</div>\n";
00458         cout << "</div>\n"; // close the mainBody div
00459         cout << "<div class=\"footer\">";
00460         cout << "<p>";
00461         if (oConfig.layout == '2')
00462         {
00463             cout << "<span class=\"glyphicon glyphicon-copyright-mark\"></span>";
00464         }
00465         else
00466         {
00467             cout << "(C)";
00468         }
00469         cout << " Dr. Philippe De Brouwer (2015)</p>";
00470         cout << "</div>\n";
00471         /*
00472         #ifdef DEBUG
00473         int k, l;
00474         for (k = 1; k < NBR_TABS; k++){
00475             for (l = 1; l < NBR_SUB_TABS; l++){
00476                 cout << "[k,l]:[" << to_string(k) << to_string(l) << "] = " << c_sub_tab_names[k-1][l-1] << "<br>\n";
00477             }
00478         }
00479         #endif
00480         */
00481
00482         cout << "</div>";
00483
00484
00485         // placed at the end of the document in order to allow the page to load faster:
00486
00487         /*cout << "<script src=\"" << oConfig.css_dir << "bootstrap/js/jquery.js\"></script>";
00488         cout << "<script src=\"" << oConfig.css_dir << "bootstrap/js/transition.js\"></script>";
00489         cout << "<script src=\"" << oConfig.css_dir << "bootstrap/js/alert.js\"></script>";
00490         cout << "<script src=\"" << oConfig.css_dir << "bootstrap/js/modal.js\"></script>";
00491         cout << "<script src=\"" << oConfig.css_dir << "bootstrap/js/dropdown.js\"></script>";
00492         cout << "<script src=\"" << oConfig.css_dir << "bootstrap/js/scrollspy.js\"></script>";
00493         cout << "<script src=\"" << oConfig.css_dir << "bootstrap/js/tab.js\"></script>";
00494         cout << "<script src=\"" << oConfig.css_dir << "bootstrap/js/tooltip.js\"></script>";
00495         cout << "<script src=\"" << oConfig.css_dir << "bootstrap/js/popover.js\"></script>";
00496         cout << "<script src=\"" << oConfig.css_dir << "bootstrap/js/button.js\"></script>";
00497         cout << "<script src=\"" << oConfig.css_dir << "bootstrap/js/collapse.js\"></script>";
00498         cout << "<script src=\"" << oConfig.css_dir << "bootstrap/js/carousel.js\"></script>";
00499         cout << "<script src=\"" << oConfig.css_dir << "bootstrap/js/typeahead.js\"></script>";
00500         cout << "<script src=\"" << oConfig.css_dir << "bootstrap/js/affix.js\"></script>";
00501         cout << "<script src=\"" << oConfig.css_dir << "bootstrap/js/holder/holder.js\"></script>";
00502         cout << "<script src=\"" << oConfig.css_dir << "bootstrap/js/google-code-prettify/prettify.js\"></
script>";
00503         cout << "<script src=\"" << oConfig.css_dir << "bootstrap/js/application.js\"></script>";*/
00504
00505
00506         if (oConfig.show_social) show_social();
00507
00508         #ifdef DEBUG
00509         cout << "<br>-----[DEBUG INFO]-----\n<br>" + debug_info << "<br>\n-----[END DEBUG INFO]-----
\n<br>";
00510         #endif
00511
00512         /* int k;
00513         for (k = 1; k < 10; k++)
00514         {
00515             cout << "<br><br><b> " << steps[k][ "name" ] << "</b>";
00516             cout << "<BR>next_step (" << k << ",0) = " << get_next_step(k,0) << " -- get_next_sub_step = " <<
get_next_sub_step(k,0);
00517             cout << "<BR>next_step (" << k << ",1) = " << get_next_step(k,1) << " -- get_next_sub_step = " <<

```

```

00518     get_next_sub_step(k,1);
        cout << "<BR>next_step (" << k << ",2) = " << get_next_step(k,2) << " -- get_next_sub_step = " <<
        get_next_sub_step(k,2);
    }*/
00519
00520
00521
00522     cout << "</body>\n"; //close div-id = wrapper
00523     cout << "</html>";
00524 }
00525
00529 void html_helper::parse_date_time()
00530 {
00531     time_t t = time(0); // get time now
00532     struct tm * now = localtime( & t );
00533     cout << (now->tm_year + 1900) << '-' << (now->tm_mon + 1) << '-' << now->tm_mday;
00534 }
00535
00539 void html_helper::header_menu_bar()
00540 {
00541     cout << "<nav id=\"nav\"><ul>";
00542     cout << "<li><a href=\"" << oConfig.wrapper_url << "\"><b>HOME</b></a></li>";
00543     cout << "<li><a href=\"" << oConfig.wrapper_url << "/solutions.html\"
00544 ><b>SOLUTIONS</b></a></li>";
00545     cout << "<li><a href=\"" << oConfig.wrapper_url << "/further_info.html\"
00546 ><b>BACKGROUND INFORMATION</b></a></li>";
00547     cout << "<li><a href=\"" << oConfig.wrapper_url << "/about.html\"><b>ABOUT
00548 US</b></a></li>";
00549     cout << "<li><a href=\"" << oConfig.wrapper_url << "/contact.html\"
00550 ><b>CONTACT</b></a></li>";
00551     cout << "</ul></nav>\n";
00552
00553     cout << "<div class=\"history\"><a HREF=\"" << oConfig.wrapper_url << "\">eFinancialPlanner</a>";
00554     cout << "&nbsp;<img src=\"" << oConfig.css_dir << "next.gif" border="0" width="5"
00555 height="9" alt="" class="noPadding">&nbsp; ";
00556     cout << "<a HREF=\"" << oConfig.soft_url << "\">" << t_title << "</a></div>";
00557     cout << "\n";
00558     cout << "<div class=\"mainBody\" id=\"mainBodyAlone\"><div id=\"appWrapper\">";
00559 }
00560
00562 void html_helper::header(int iid, bool addSavingsPlan, bool addPlots)
00563 {
00564     if (iid != 0) cout << "Set-Cookie:iid=" << iid << ";";
00565     cout << c_content_header;
00566     cout << c_doc_type;
00567     switch(oConfig.language)
00568     {
00569     case '1':
00570         cout << "\n<html lang=\"en-UK\">\n"; break;
00571     case '2':
00572         cout << "\n<html lang=\"en-US\">\n"; break;
00573     default:
00574         cout << "\n<html lang=\"en-UK\">\n";
00575     }
00576     cout << "<head>\n";
00577     cout << "<title>" << t_title << "</title>\n";
00578     cout << "<meta charset=\"utf-8\">\n";
00579     cout << "<meta name=\"Description\" content=\"Personal Financial Planning\">\n";
00580     cout << "<meta name=\"Keywords\" content=\"financial planning, Philippe, De Brouwer, Philippe De
00581 Brouwer, PFP, Personal Financial Plan, Behavioral Finance, Behavioural Finance, Dr. Philippe J.S. De Brouwer\">\n";
00582     cout << "<meta name=\"author\" content=\"Philippe De Brouwer\">\n";
00583     cout << "<LINK REL=\"icon\" HREF=\"favicon.ico\" TYPE=\"image/icon\">\n";
00584     cout << "<LINK REL=\"SHORTCUT ICON\" HREF=\"favicon.ico\">\n";
00585     cout << "<script src=\"" << oConfig.scripts_dir << "dist/jquery.min.js\"></script>\n";
00586
00587     //cout << "<script class=\"include\" type=\"text/javascript\"
00588     //src=\"http://ajax.googleapis.com/ajax/libs/jquery/1.9.1/jquery.min.js\"></script>";
00589
00590     if (addPlots)
00591     {
00592     cout << "<script src=\"" << oConfig.scripts_dir << "dist/jquery.jqplot.min.js\"
00593 ></script>\n";
00594     cout << "<script src=\"" << oConfig.scripts_dir << "jqplot.pieRenderer.min.js\"
00595 ></script>\n";
00596     cout << "<link rel=\"stylesheet\" type=\"text/css\" href=\"" << oConfig.
00597 css_dir << "jquery.jqplot.min.css\">\n";
00598     cout << "<style type=\"text/css\">.jqplot-data-label {color: #444;font-size: 1.1em;}</style>\n";
00599     cout << "<script src=\"" << oConfig.scripts_dir << "jqplot.customMarkerRenderer.js
00600 \"></script>\n";
00601     }
00602
00603     //cout << "<script type=\"text/javascript\" src=\"" << oConfig.scripts_dir <<
00604     // "calendarDateInput.js\"></script>\n";
00605     cout << "<script src=\"" << oConfig.scripts_dir << "goal_type_sh.js\"></script>\n";

```

```

00604     cout << "<script src=\"" << oConfig.scripts_dir << "extraInfo.js\"></script>\n";
00605     if (addSavingsPlan) cout << "<script src=\"" << oConfig.scripts_dir << "
savings_plan_sh.js\"></script>\n";
00606
00608     cout << "<script src=\"" << oConfig.scripts_dir << "
datepicker/js/bootstrap-datepicker.js\"></script>\n";
00609     cout << "<link rel=\"stylesheet\" type=\"text/css\" href=\"" << oConfig.
scripts_dir << "datepicker/css/datepicker.css\">\n";
00610
00612     switch(oConfig.layout)
00613     {
00614     case '1':
00615     cout << "<link rel=\"stylesheet\" type=\"text/css\" href=\"" << oConfig.
css_dir << "base2.css\">\n";
00616     break;
00617     case '2':
00618     cout << "<link rel=\"stylesheet\" type=\"text/css\" href=\"" << oConfig.
css_dir << "base1.css\">\n";
00619     cout << "<meta name=\"viewport\" content=\"width=device-width, initial-scale=1.0\">\n";
00620     cout << "<link href=\"" << oConfig.css_dir << "bootstrap/css/bootstrap.css\" rel=\"
stylesheet\" media=\"screen\">\n";
00621     cout << "<link href=\"" << oConfig.css_dir << "bootstrap/css/bootstrap-responsive.css
\" rel=\"stylesheet\" media=\"screen\">\n";
00622     cout << "<link href=\"" << oConfig.css_dir << "bootstrap/css/bootstrap-theme.css\"
rel=\"stylesheet\">\n";
00623     cout << "<script src=\"" << oConfig.css_dir << "bootstrap/js/bootstrap.min.js\"></script>
";
00624     //>>> cout << "<link rel=\"stylesheet\" type=\"text/css\" href=\"" << oConfig.css_dir <<
"base2_post.css\">\n";
00625     break;
00626     default:
00627     cout << "<link rel=\"stylesheet\" type=\"text/css\" href=\"" << oConfig.
css_dir << "base2.css\">\n";
00628     }
00629
00631     switch (oConfig.font_combination)
00632     {
00633     case '1' : //corben&Nobile
00634     cout << "<link href=\"http://fonts.googleapis.com/css?family=Corben:bold\" rel=\"stylesheet\" type=
\"text/css\">";
00635     cout << "<style>h1, h2, h3, h4, h5, h6 {font-family: 'Corben', Georgia, Times, serif;}</style>";
00636     cout << "<link href=\"http://fonts.googleapis.com/css?family=Corben:bold\" rel=\"stylesheet\" type=
\"text/css\">";
00637     cout << "<style>div, body, p {font-family: 'Nobile', Helvetica, Arial, sans-serif;}</style>";
00638     break;
00639     case '2' : //Droid Serif+Sans
00640     cout << "<link href=\"http://fonts.googleapis.com/css?family=Droid+Serif\" rel=\"stylesheet\" type=
\"text/css\">";
00641     cout << "<style>h1, h2, h3, h4, h5, h6 {font-family: 'Droid Serif', Georgia, Times, serif;}</style>";
00642     cout << "<link href=\"http://fonts.googleapis.com/css?family=Droid+Sans\" rel=\"stylesheet\" type=
\"text/css\">";
00643     cout << "<style>div, body, p {font-family: 'Droid Sans', Helvetica, Arial, sans-serif;}</style>";
00644     break;
00645     case '3' : //Ubuntu + default
00646     cout << "<link href=\"http://fonts.googleapis.com/css?family=Ubuntu:bold\" rel=\"stylesheet\" type=
\"text/css\">";
00647     cout << "<style>h1, h2, h3, h4, h5, h6 {font-family: 'Ubuntu', Georgia, Times, serif;}</style>";
00648     cout << "<style>div, body, p {font-family: sans-serif;}</style>";
00649     break;
00650     case '4' : //Ubuntu+Vollkorn
00651     cout << "<link href=\"http://fonts.googleapis.com/css?family=Ubuntu:bold\" rel=\"stylesheet\" type=
\"text/css\">";
00652     cout << "<style>h1, h2, h3, h4, h5, h6 {font-family: 'Ubuntu', Georgia, Times, serif;}</style>";
00653     cout << "<link href=\"http://fonts.googleapis.com/css?family=Vollkorn\" rel=\"stylesheet\" type=\"
text/css\">";
00654     cout << "<style>div, body, p {font-family: 'Vollkorn', Georgia, Times, Serif;}</style>";
00655     break;
00656     case '5' : //|Journal + default
00657     cout << "<link href=\"" << oConfig.css_dir << "fonts/journal/font.css\" rel=\"
stylesheet\" type=\"text/css\">";
00658     cout << "<style>h1, h2, h3, h4, h5, h6 {font-weight:bold;font-family: 'JOURNAL', Georgia, Times,
serif;}</style>";
00659     cout << "
<style>h1{font-size:4.5em};h2{font-size:3.5em};h3{font-size:2.5em};h4{font-size:2em};h5{font-size:1em};</style>";
00660
00661     // cout << "<link href=\"http://fonts.googleapis.com/css?family=Vollkorn\" rel=\"stylesheet\"
type=\"text/css\">";
00662     cout << "<style>div, body, p {font-family: sans-serif;}</style>";
00663     break;
00664     case '6' : //|Journal + Gentium
00665     cout << "<link href=\"" << oConfig.css_dir << "fonts/journal/font.css\" rel=\"
stylesheet\" type=\"text/css\">";
00666     cout << "<style>h1, h2, h3, h4, h5, h6 {font-weight:bold;font-family: 'JOURNAL', Georgia, Times,
serif;}</style>";
00667     cout << "
<style>h1{font-size:3.5em};h2{font-size:2.5em};h3{font-size:2em};h4{font-size:1.75em};h5{font-size:1.5em};</style>";

```

```

00668     cout << "<link href=\"" << oConfig.css_dir << "fonts/gentium/font.css\" rel=\""
stylesheet\" type=\"text/css\">";
00669     cout << "<style>body {font-size: 17px;}table.jqplot-table-legend {font-size:.75em;}</style>";
00670     cout << "<style>div, body, p {font-family: GentiumPlusW, sans-serif;}</style>";
00671     break;
00672
00673     case '7' : //|Gentium + Gentium
00674     cout << "<link href=\"" << oConfig.css_dir << "fonts/gentium/font.css\" rel=\""
stylesheet\" type=\"text/css\">";
00675     cout << "<style>h1, h2, h3, h4, h5, h6 {font-weight:bold;font-family: GentiumPlusW,
sans-serif;}</style>";
00676     cout << "<style>body {font-size: 17px;}table.jqplot-table-legend {font-size:.75em;}</style>";
00677     //cout <<
"<style>h1{font-size:4.5em};h2{font-size:3.5em};h3{font-size:2.5em};h4{font-size:2em};h5{font-size:1em;}</style>";
00678     cout << "<style>div, body, p {font-family: GentiumPlusW, sans-serif;}</style>";
00679     break;
00680
00681     default:
00682     cout << "<style>h1, h2, h3, h4, h5, h6 {font-family: Georgia, Times, serif;}</style>";
00683     cout << "<style>div, body, p {font-family: 'Droid Sans', Helvetica, Arial, sans-serif;}</style>";
00684 }
00685
00686 cout << "</head>\n";
00687
00689 cout << "<body>\n";
00690 switch (oConfig.layout)
00691 {
00692     case '2':
00693     cout << "<div class=\"container-fluid\">";
00694     case '1':
00695     default:
00696     cout << "<div id=\"wrapper\">\n";
00697 }
00698 if (oConfig.parse_wrapper) header_menu_bar();
00699 // cout << h1(t_title + "/" + title);
00700 // cout << "<hr/>";
00701
00702 if (error_message != "")
00703 {
00704     cout << "<div class=\"alert alert-warning\"><ul>" << error_message << "</ul>";
00705     cout << "</div>\n"; // close the alert alert-warning div
00706     error_message = "";
00707 }
00708 if (std_message != "")
00709 {
00710     cout << "<div class=\"alert alert-info\"><ul>" << std_message << "</ul>";
00711     cout << "</div>\n";
00712     std_message = "";
00713 }
00714
00715 }
00716
00722 void html_helper::show_home()
00723 {
00724 /* long k;
00725     long fib=1, fibprev=1;
00726     cout << "<h2>Fibonacci numbers: </h2>";
00727     cout << fibprev;
00728     for (k = 1; k <= 75; k++) {
00729         cout << "\t--\t";
00730         cout << fib;
00731         fib = fib + fibprev;
00732         fibprev = fib - fibprev;
00733     }
00734     cout << "\r\n";
00735 */
00736 using namespace cgicc;
00737 Cgicc cgi;
00738 int isLogin;
00739
00740 form_iterator uid = cgi.getElement("uid");
00741 if(uid != cgi.getElements().end()) { isLogin = 1;} else { isLogin = 0;}
00742
00743 cout << p("This simple process consists of three easy steps:");
00744 cout << "<img src=\"" << oConfig.www_dir << "/img/ranking.jpeg\" align=\"right\"/>";
00745 cout << ul();
00746 cout << li("<a href=\"" + oConfig.soft_url + "?a=rf\">Register</a> or <a href=\"" +
oConfig.soft_url + "?a=lf\">Login</a>");
00747 if (isLogin == 0)
00748 {
00749     cout << li("STEP 1: Make an inventory of your assets and main cash-flows (income and expenses)");
00750     cout << li("STEP 2: Set Your personal financial goals");
00751     cout << li("STEP 3: Contemplate on the Feedback");
00752     cout << p(" ");
00753     cout << "<p>Next step: <button type=\"btn btn-primary\" class=\"btn btn-primary\" onclick=\"
window.location.href=\"" + oConfig.soft_url + "?a=rf\">Register</button> or ";
00754     cout << "<button type=\"btn btn-primary\" class=\"btn btn-primary\" onclick=\"window.location.href=\"";

```

```

    " + oConfig.soft_url + "?a=lf\'>Login</button></p>";
00755     cout << p(" ");
00756 }
00757 else
00758 {
00759     cout << li("STEP 1: <a href=\'" + oConfig.soft_url + "?a=if\'>Make an inventory of
your assets and main cash-flows (income and expenses)</a>");
00760     cout << li("STEP 2: <a href=\'" + oConfig.soft_url + "?a=gf\'>Set Your personal
financial goals</a>");
00761     cout << li("STEP 3: <a href=\'" + oConfig.soft_url + "?a=f\'>Contemplate on the
Feedback</a>");
00762     cout << p(" ");
00763     cout << "<p>Next step: <button type=\'btn btn-primary\' onclick=\'window.location.href=\'" +
oConfig.soft_url + "?a=gl\'>Define Your Goals</button></p>";
00764     cout << p(" ");
00765 }
00766     cout << ul();
00767
00768     // test for normal distributicgion
00769 /*     cout << "<a href=\'" + oConfig.soft_url + "?1000+i.cf+a+20\'>do dol()</a>";
00770     double a[] = {0.001, 0.01, 0.1, 0.5, 0.65, 0.9, 0.95, 0.98, 0.99, 0.999};
00771     cout << "<table border=\'1\'><tr><th>x</th><th>norminv(x)</th></tr>";
00772     for (k = 0; k < (sizeof(a)/sizeof(*a)); k++) {cout << "<tr><td>" << a[k] << "</td><td>" << norminv(a[k])
<< "</td></tr>";}
00773     cout << "</table>";
00774     double b[] = {-5,-4,-3,-2,-1,0,1,2,3,4,5};
00775     cout << "<table border=\'1\'><tr><th>x</th><th>phi(x)</th></tr>";
00776     for (k = 0; k < (sizeof(b)/sizeof(*b)); k++) {cout << "<tr><td>" << b[k] << "</td><td>" << phi(b[k]) <<
"</td></tr>";}
00777     cout << "</table>";
00778 */
00779 }
00780
00781
00782 /* *****
00783 * get_string
00784 * *****/
00785 string html_helper::get_string(const std::string& varName)
00786 {
00787     try
00788     {
00789         Cgicc formData;
00790         form_iterator fvalue1 = formData.getElement(varName);
00791         if( !fvalue1->isEmpty() && fvalue1 != (*formData).end()) {return **fvalue1;} else {return "";}
00792     }
00793     catch(exception& e) { cout << "**ERROR!!**\n";     return "ERROR";}
00794 }
00795
00802 inline void html_helper::navbar_start()
00803 {
00804     switch (oConfig.layout)
00805     {
00806         case '2' :
00807             bs_navbar_start();
00808         break;
00809         case '1' :
00810         default :
00811             cout << "<ul class=\'nav-tabs\'>";
00812         break;
00813     }
00814 }
00815
00821 inline void html_helper::navbar_item(int k, string usr_name, string tab_name)
00822 {
00823     int l;
00824     int selected_sub = get_sub_step_nbr(tab_name);
00825     int active_tab_nbr = get_step_nbr(tab_name);
00826     bool produce_sublist, is_active, is_disabled;
00827     string the_url;
00828
00829     is_active = (k == active_tab_nbr)?true:false;
00830     is_disabled = ((usr_name == "") && (steps[k]["must_login"] == "Y"))?true:false;
00831     the_url = ((usr_name == "") && (steps[k]["must_login"] == "Y"))?"&#35":
steps[k]["url"];
00832
00833     switch (oConfig.layout)
00834     {
00835         case '2' : // bootstrap
00836         {
00837             cout << "<li role=\'presentation\' class=\'";
00838             if (is_active)
00839             {
00840                 cout << "active";
00841             }
00842             else if (is_disabled)
00843             {
00844                 cout << "disabled ";

```

```

00845     }
00846     if (sub_step_exists(k,1) && oConfig.layout == '2')
00847     {
00848         cout << " dropdown"><a class="dropdown-toggle" data-toggle="dropdown" href="#" role="button
" aria-expanded="false"> << steps[k]["name"] << "<span class="caret"></span></a>";
00849     }
00850     else
00851     {
00852         cout << "\"><a href=\"" << the_url << "\"> << steps[k]["name"] << "</a>";;
00853     }
00854     produce_sublist = (sub_step_exists(k,1))?true:false; //this must be here as it can
depend on the presentation
00855     if (produce_sublist)
00856     {
00857         if (sub_step_exists(k,1) && oConfig.layout == '2')
00858         {
00859             cout << "<ul class="dropdown-menu" role="menu">";
00860         }
00861         else
00862         {
00863             cout << "<ul>";
00864         }
00865     }
00866     for (l = 1; (l <= NBR_SUB_TABS) && (sub_step_exists(k,l)); l++)
00867     {
00868         cout << "<li><A href=\"" << sub_steps[k][l]["url"] << "\"";
00869         if (l == selected_sub) {cout << " class="active" ";}
00870         cout << ">" << sub_steps[k][l]["name"] << "</a></li>";
00871     }
00872     cout << "</ul>";
00873 }
00874 cout << "</li>\n";
00875 }
00876 break;
00877 case '1':
00878 default :
00879 {
00880     cout << "<li role="presentation" class="";
00881     if (is_active)
00882     {
00883         cout << "active";
00884     }
00885     else if (is_disabled)
00886     {
00887         cout << "disabled ";
00888     }
00889     if (sub_step_exists(k,1) && oConfig.layout == '2')
00890     {
00891         cout << " dropdown"><a class="dropdown-toggle" data-toggle="dropdown" href="#" role="button
" aria-expanded="false"> << steps[k]["name"] << "<span class="caret"></span></a>";
00892     }
00893     else
00894     {
00895         cout << "\"><a href=\"" << the_url << "\"> << steps[k]["name"] << "</a>";;
00896     }
00897     produce_sublist = (k == active_tab_nbr && sub_step_exists(k,1))?true:false;
00898     if (produce_sublist)
00899     {
00900         if (sub_step_exists(k,1) && oConfig.layout == '2')
00901         {
00902             cout << "<ul class="dropdown-menu" role="menu">";
00903         }
00904         else
00905         {
00906             cout << "<ul>";
00907         }
00908     }
00909     for (l = 1; (l <= NBR_SUB_TABS) && (sub_step_exists(k,l)); l++)
00910     {
00911         cout << "<li><A href=\"" << sub_steps[k][l]["url"] << "\"";
00912         if (l == selected_sub) {cout << " class="active" ";}
00913         cout << ">" << sub_steps[k][l]["name"] << "</a></li>";
00914     }
00915     cout << "</ul>";
00916 }
00917 cout << "</li>\n";
00918 }
00919 break;
00920 }
00921 }
00922 }
00923
00929 inline void html_helper::navbar_end()
00930 {
00931     switch (oConfig.layout)
00932     {
00933         case '2' :

```

```

00934         bs_navbar_end();
00935         break;
00936         case '1' :
00937         default :
00938             cout << "</ul>";
00939             break;
00940     }
00941 }
00942
00948 inline void html_helper::navbar_box(int k, string usr_name)
00949 {
00950     if (usr_name == "") return; // no box if no user logged in as this is mainly for account management
00951
00952     switch (oConfig.layout)
00953     {
00954         case '2' :
00955         {
00956             cout << "<ul class=\"nav navbar-nav navbar-right\">";
00957             cout << "<li class=\"dropdown\">";
00958             cout << " <a href=\"#\" class=\"dropdown-toggle\" data-toggle=\"dropdown\" role=\"button\"
00959             aria-haspopup=\"true\" aria-expanded=\"false\"><span class=\"caret\"></span></a>";
00960             cout << " <ul class=\"dropdown-menu\">";
00961             for (k=0; k < NBR_RIGHT_MENU;k++)
00962             {
00963                 cout << "<li><a href=\"\" << c_right_menu_urls[k] << \"\>\" <<
00964                 c_right_menu_names[k] << "</a></li>";
00965             }
00966             cout << "</ul>";
00967             cout << "</li>"; //of the first menu item
00968             cout << "</ul>"; // close the right-menu
00969         }
00970         break;
00971         case '1' :
00972         default :
00973         {
00974             cout << "<span class=\"floatRight\">";
00975             cout << "logged in as: <i>" << usr_name << "</i>&nbsp;&nbsp;&nbsp;<br>";
00976             for (k=0; k < NBR_RIGHT_MENU;k++)
00977             {
00978                 cout << "<a href=\"\" << c_right_menu_urls[k] << \"\>\" <<
00979                 c_right_menu_names[k] << "</a> &nbsp;&nbsp;&nbsp;";
00980             }
00981             cout << "</span>";
00982         }
00983         break;
00984     }
00985 }
00988 void html_helper::show_toolBar(string tab_name , string usrName)
00989 {
00990     int k;
00991     int selected = get_step_nbr(tab_name);
00992     int selected_sub = get_sub_step_nbr(tab_name);
00993
00994     //open the nav-bar
00995     navbar_start();
00996     //place each item of the nav-bar
00997
00998     for (k = 1; k <= NBR_TABS; k++)
00999     {
01000         navbar_item(k, usrName, tab_name);
01001     }
01002
01003     // -- the box with account related links
01004     switch (oConfig.layout)
01005     {
01006         case '2' :
01007             bs_navbar_end1();
01008             navbar_box(k, usrName);
01009             navbar_end();
01010             bs_navbar_end2();
01011
01012             break;
01013         case '1' :
01014         default :
01015             navbar_end();
01016             navbar_box(k, usrName);
01017             break;
01018     }
01019
01020     cout << "<br style=\"clear:all;\>";
01021
01022     parse_title(selected, selected_sub);
01023     /*
01024     *
01025     cout << "<div class=\"nav\"><table class=\"toolBar\"><tr>";

```

```

01026     for (k = 1; k <= NBR_TABS; k++)
01027     {
01028         if (first) {first = false;} else {cout << "<td class=\"toolBarArrow\">&nbsp;</td>";}
01029         cout << "<TD class=";
01030         if (actives[k-1] == '0')
01031             {cout << "\"toolBarInactive\">" << c_tab_names[k-1] << "</TD>\n";}
01032         else
01033             {
01034                 if (k == selected) {cout << "\"toolBarSelected\"";} else {cout << "\"toolBar\"";}
01035                 cout << "><A class=\"toolBar\" href=\"" << c_tab_urls[k-1] << "\">" << c_tab_names[k-1] << "</A></
TD>\n";
01036             }
01037     }
01038     cout << "</tr></table></div>\n";*/
01039     // parse_breadcrumb(selected, selected_sub);
01040
01041 }
01042
01046 void html_helper::next_step(int stepNbr, int subStepNbr)
01047 {
01048     if (stepNbr != 0)
01049     {
01050         /*      cout << "<div class=\"nextStep\"><h2><img src=\"" << oConfig.css_dir << "arrow_right.svg\"
width=\"100px\" height=\"12px\" >NEXT STEP: <a href=\""
01051             if (subStepNbr == 0)
01052                 {cout << c_tab_urls[stepNbr] << "\">" << c_tab_names[stepNbr];}
01053             else
01054                 {cout << c_sub_tab_urls[stepNbr-1][subStepNbr] << "\">" << c_sub_tab_names[stepNbr-1][subStepNbr];}
01055             cout << "</a></h2></div>";
01056         */
01057
01058         int prevSubStepNbr = get_prev_sub_step(stepNbr, subStepNbr);
01059         int prevStepNbr     = get_prev_step(stepNbr, subStepNbr);
01060         int nextSubStepNbr = get_next_sub_step(stepNbr, subStepNbr);
01061         int nextStepNbr    = get_next_step(stepNbr, subStepNbr);
01062
01063         if (prevStepNbr != 0)
01064         {
01065             string back_symbol;
01066             switch (oConfig.layout)
01067             {
01068                 case '1' :
01069                     back_symbol = "<img src=\"" + oConfig.css_dir + "arrow_right.svg\" width=\"150px\"
height=\"12px\" >";
01070                     break;
01071                 case '2' :
01072                     back_symbol = "<span class=\"glyphicon glyphicon-step-backward\"></span>";
01073                     break;
01074                 default :
01075                     back_symbol = " <span class=\"glyphicon glyphicon-step-forward\"></span>";
01076                     break;
01077             }
01078
01079             cout << "\n<a class=\"btn btn-default\" href=\""
01080                 if (prevSubStepNbr == 0)
01081                 {
01082                     cout << steps[prevStepNbr]["url"] << "\">"<< back_symbol <<
t_prev_step << ": " << steps[prevStepNbr]["name"];
01084                 }
01085                 else
01086                 {
01087                     cout << sub_steps[prevStepNbr][prevSubStepNbr]["url"] << "\">" << back_symbol <<
t_prev_step << ": " << sub_steps[prevStepNbr][prevSubStepNbr]["name"];
01088                 }
01089                 cout << "</a>\n";
01090             }
01091
01092             if (nextStepNbr != 0)
01093             {
01094                 cout << "\n<a class=\"btn btn-primary\" href=\""
01095                     if (nextSubStepNbr == 0)
01096                     {
01097                         cout << steps[nextStepNbr]["url"] << "\">" << t_next_step << ": " <<
steps[nextStepNbr]["name"];
01099                     }
01100                     else
01101                     {
01102                         cout << sub_steps[nextStepNbr][nextSubStepNbr]["url"] << "\">" <<
t_next_step << ": " << sub_steps[nextStepNbr][nextSubStepNbr]["name"];
01103                     }
01104                     switch (oConfig.layout)
01105                     {
01106                         case '1' : cout << "<img src=\"" << oConfig.css_dir << "arrow_right.svg\" width=\"150px
\" height=\"12px\" >"; break;
01107                         case '2' : cout << "<span class=\"glyphicon glyphicon-step-forward\"></span>"; break;

```

```

01108     default : cout << " <span class=\"glyphicon glyphicon-step-forward\"></span>; break;
01109     }
01110     cout << "</a>\n";
01111     }
01112 }
01113 }
01114 }
01115 }
01119 void html_helper::aside(const string& textLine)
01120 {
01121     if (oConfig.show_aside)
01122     {
01123         srand (time(NULL)); //initialize random seed
01124         int iRand = rand() % c_photos_phdb.size() + 1; // generate random photo number
01125         cout << "<aside>";
01126         cout << "<img src=\"" << c_photos_phdb_dir << c_photos_phdb.find(iRand)->
second;
01127         cout << "\" alt=\"Philippe's picture\" align=\"center\" width=\"100px\">";
01128         cout << "<p>&quot;" << textLine << "&quot;</p>";
01129         show_social();
01130         cout << "</aside>\n";
01131     }
01132     else {}
01133 }
01134 }
01138 void html_helper::show_social()
01139 {
01140     cout << "<br><div id=\"share\">";
01141     cout << "<script src=\"//platform.linkedin.com/in.js\" type=\"text/javascript\">lang:
en_US</script><script type=\"IN/Share\" data-counter=\"right\"></script>";
01142     cout << "<div id=\"fb-root\"></div>";
01143     cout << "<script>(function(d, s, id) {";
01144     cout << "    var js, fjs = d.getElementsByTagName(s)[0];";
01145     cout << "    if (d.getElementById(id)) return;";
01146     cout << "    js = d.createElement(s); js.id = id;";
01147     cout << "    js.src = \"//connect.facebook.net/en_GB/all.js#xfbml=1\";";
01148     cout << "    fjs.parentNode.insertBefore(js, fjs);";
01149     cout << "    }(document, 'script', 'facebook-jssdk'))</script>";
01150     cout << "<div class=\"fb-like\" data-href=\"" << oConfig.soft_url << "\" data-layout=\"
button\" data-action=\"like\" data-show-faces=\"false\" data-share=\"true\"></div></div>";
01151 }
01152 }
01158 void html_helper::parse_breadcrumb(int tab, int sub_tab)
01159 {
01160     switch (oConfig.layout)
01161     {
01162         case '2' :
01163             cout << "<ol class =\"breadcrumb\">";
01164             cout << "<li>" << t_breadcrumb << "</li>";
01165             cout << "<li><a HREF=\"" << oConfig.wrapper_url << "\">" <<
oConfig.wrapper_name << "</a></li>";
01166             cout << "<li><a HREF=\"" << oConfig.soft_url << "\">" <<
oConfig.soft_name << "</a></li>";
01167             cout << "<li><a HREF=\"" << steps[tab]["url"] << "\">" << steps[tab]["name"] << "</a></li>";
01168             if (tab != 0 && sub_tab != 0)
01169             {
01170                 cout << "<li><a HREF=\"" << sub_steps[tab][sub_tab]["url"] << "\">" <<
sub_steps[tab][sub_tab]["name"] << "</a></li>";
01171             }
01172             cout << "</ol>";
01173             cout << "<br style=\"clear:left;\"/>";
01174             break;
01175         case '1' :
01176         default :
01177             cout << "<div class=\"breadcrumb\">" << t_breadcrumb;
01178             cout << "<a HREF=\"" << oConfig.wrapper_url << "\">" <<
oConfig.wrapper_name << "</a>" << c_arrow;
01179             cout << "<a HREF=\"" << oConfig.soft_url << "\">" << oConfig.
soft_name << "</a>";
01180             cout << "\" << c_arrow << "<a HREF=\"" << steps[tab]["url"] << "\">" << steps[tab]["name"] << "
</a>";
01181             if (tab != 0 && sub_tab != 0)
01182             {
01183                 cout << c_arrow;
01184                 cout << "<a HREF=\"" << sub_steps[tab][sub_tab]["url"] << "\">" <<
sub_steps[tab][sub_tab]["name"] << "</a>";
01185             }
01186             cout << "</div>";
01187             cout << "<br style=\"clear:left;\"/>";
01188             break;
01189     }
01190 }
01191 }
01192 }
01198 void html_helper::parse_title(int tab, int sub_tab)
01199 {
01200 // cout << "<BR>tab=" << tab << "    sub_tab=" << sub_tab;

```

```

01201 cout << "<br style=\"clear:all;\"/>\n";
01202 auto outerIt = steps.find(tab);
01203 if (outerIt != steps.end())
01204 {
01205     auto innerMapPtr = outerIt->second;
01206     const string KeyInner = "title";
01207     auto innerIt = innerMapPtr.find(KeyInner);
01208     if (innerIt != innerMapPtr.end()) cout << "<h1>" << steps[tab]["title"] << "</h1>";
01209 }
01210 cout << t_after_title[steps[tab]["name"]];
01211 auto outer2It = sub_steps.find(tab); // go look in the sub-step
01212 if (outer2It != sub_steps.end())
01213 {
01214     auto innerMapPtr = outer2It->second;
01215     const int KeyInner2 = sub_tab;
01216     auto innerIt = innerMapPtr.find(KeyInner2);
01217     if (innerIt != innerMapPtr.end())
01218     {
01219         auto inner2MapPtr = innerIt->second;
01220         const string KeyInner3 = "title";
01221         auto inner2It = inner2MapPtr.find(KeyInner3);
01222         if (inner2It != inner2MapPtr.end()) cout << "<h2>" << sub_steps[tab][sub_tab]["title"] << "
</h2>";
01223     }
01224 }
01225 cout << endl;
01226 }
01227
01233 void html_helper::cout_box_color(string the_color)
01234 {
01235     if (the_color == "brightgreen") cout << "brightgreenBox";
01236     else if (the_color == "green") cout << "greenBox";
01237     else if (the_color == "amber") cout << "amberBox";
01238     else if (the_color == "red") cout << "redBox";
01239     else if (the_color == "darkred") cout << "darkredBox";
01240     else {cout << "amberBox";}
01241 }
01242
01246 string html_helper::get_box_color_class(string the_color)
01247 {
01248     string s;
01249     if (the_color == "brightgreen") s = "brightgreenBox";
01250     else if (the_color == "green") s = "greenBox";
01251     else if (the_color == "amber") s = "amberBox";
01252     else if (the_color == "red") s = "redBox";
01253     else if (the_color == "darkred") s = "darkredBox";
01254     else {s = "amberBox";}
01255     return s;
01256 }
01257
01261 string html_helper::get_glyphicon(string the_color)
01262 {
01263     string s;
01264     switch (oConfig.layout)
01265     {
01266     case '2' :
01267         s = "<span class=\"";
01268         if (the_color == "brightgreen") s = s + "glyphicon glyphicon-thumbs-up";
01269         else if (the_color == "green") s = s + "glyphicon glyphicon-thumbs-up";
01270         else if (the_color == "amber") s = s + "glyphicon glyphicon-alert";
01271         else if (the_color == "red") s = s + "glyphicon glyphicon-thumbs-down";
01272         else if (the_color == "darkred") s = s + "glyphicon glyphicon-thumbs-down";
01273         else {s = s + "glyphicon";}
01274         s = s + "\"></span>";
01275         break;
01276     case '1' :
01277     default:
01278         if (the_color == "brightgreen") s = ":-D";
01279         else if (the_color == "green") s = ":-)";
01280         else if (the_color == "amber") s = ":-|";
01281         else if (the_color == "red") s = ":-(";
01282         else if (the_color == "darkred") s = ":-(-";
01283         else {s = ":-?";}
01284         break;
01285     }
01286     return s;
01287 }
01288
01289
01290
01294 inline string html_helper::terms_of_use()
01295 {
01296     string s = "<a href=\" + oConfig.soft_url + "?a=d#use\">Terms of Use</a>";
01297     return s;
01298 }
01299
01303 inline string html_helper::privacy_policy()

```

```

01304 {
01305     string s = "<a href=\" + oConfig.soft_url + "?a=d#privacy\">Privacy Policy</a>";
01306     return s;
01307 }
01308
01312 inline string html_helper::cookie_policy()
01313 {
01314     string s = "<a href=\" + oConfig.soft_url + "?a=d#cookie\">Cookie Policy</a>";
01315     return s;
01316 }

```

8.33 investment_problem.class.cpp File Reference

Classes

- class [investment_problem](#)

8.34 investment_problem.class.cpp

```

00001
00015 class investment_problem : public investor, public
portfolio
00016 {
00017 public:
00018 //inherited: int investor_id;
00019     investment_problem(int investor);
00020     void solve();
00021     int get_nbr_goals();
00022     std::map<int, aGOAL> goalZ;
00023     int get_max_month_for_lower_goals(int g);
00024     void javaGraph(int g, string xtra_var = "", string xtra_label = "", int followup_mnth = 0);
00025     void prepare_javaVars_colors(int g, int followup_mnth);
00026     void parse_javaGraph(int g, string xtra_var = "", string xtra_label = "");
00027
00028 protected:
00029     std::map<int, portfolio> portfolios;
00030     void set_followup_color_prae(float Vlow, float Vmed, float Vhigh, int g);
00031     void set_followup_color_post(float V, int g);
00032
00033 private:
00034
00035     std::map<int, float> means;
00036     std::map<int, float> means_left;
00037     std::map<int, float> means_used;
00038     std::map<int, float> means_block;
00039     std::map<int, float> means_tmp;
00040
00041     int nbr_portfolios;
00042     //inherited: age; // the age of the investor TODAY
00043     int nbr_goals;
00044     int nbr_assets = 0;
00045     int nbr_cfs;
00046
00047     void set_assets();
00048     void set_cfs();
00049     void set_goals();
00050     void set_means_goal(int g);
00051     void expand_cf(float amount, string freq = "M", string fromStr="", string tillStr="", int
goal_nbr = 0);
00052     void set_portfolios();
00053     float get_optimal_portfolio(int g);
00054     float calc_alpha(int priority, int monthNbr);
00055     void set_priorityLimits();
00056     float goal_seek_means(int g, int p);
00057     void goal_seek_tmp_means(float *perc, int g, int p);
00058     float calc_risk(int g, int p);
00059     float calc_V_high(int g, int p = 0);
00060     void set_goal_color(int g, float opt_perc);
00061     void set_goal_remarks(int g);
00062     void allocate_to_unallocated_goal(int unallocated_goal_nbr);
00063     int get_portfolio_safest();
00064     int get_portfolio_riskiest();
00065     bool set_unallocated_goal();
00066     //bool* portfolio_suitable = NULL;
00067     std::map<int, bool> portfolio_suitable;
00068
00069

```

```

00070 // float get_risk(const char* riskFunction, int theGoal, int thePortfolio);
00071 // float ExpectedShortfall_normDist(int theGoal, int thePortfolio);
00072 int priorityMax;
00073 int priorityMin;
00074 /*
00075 std::map<int, asset> assets; // set of asset_objects
00076 std::map<int, cf> cfs; // set of CF-objects
00077 std::map<int, goal> goals; // set of goal-objects
00078 */
00079
00080 };
00081
00085 investment_problem::investment_problem(int
investor)
00086 // : market(investor) ///< note: parent class constructor in the initializer list in order to pass the
parameter
00087 {
00088 if (load_from_db(to_string(investor)) <= 0 ) { /*TODO : errorhandling*/ }
00089 investor_id = investor;
00090 for (int i=0; i <= months2simulate(); i++)
00091 {
00092 means.insert (std::pair<int,float>(i, 0.0) );
00093 means_left.insert (std::pair<int,float>(i, 0.0) );
00094 means_used.insert (std::pair<int,float>(i, 0.0) );
00095 means_block.insert(std::pair<int,float>(i, 0.0) );
00096 means_tmp.insert (std::pair<int,float>(i, 0.0) );
00097 } // initialize means ...
00099 set_assets(); // adds assets to means[]
00100 set_cfs(); // adds cash flows to means[]
00101
00102 set_goals(); // adds goals to goalZ[] in the right order(!)
00103
00105 //market(investor_id); ///< re-execute the constructor with the right user-id now ... DOES NOT WORK
(changes investor_id into a pointer to market)
00106 load_ER(investor_id);
00107 load_covar(investor_id);
00108 set_portfolios(); // sets nbr_portfolios, the map portfolios and the portfolio_suitable map
(executes set_mu() and set_sigma, using the ER van covar)
00109
00111 if (nbr_goals == 0) {error_message = error_message +
t_errMsg["no_goals"];}
00112 if (nbr_assets + nbr_cfs == 0) {error_message =
error_message + t_errMsg["no_assets"];}
00113
00114 }
00115
00122 void investment_problem::set_assets()
00123 {
00124 string s, the_curr = "";
00125 float theSum = 0;
00126 cls_currency oCurr;
00127 s = "SELECT amount, currency FROM " + oConfig.tbl_prefix + "_assets WHERE investor = " +
std::to_string(investor_id) + ";";
00128
00129 db.res = db.stmt->executeQuery(s);
00130 while (db.res->next())
00131 {
00132 the_curr = db.res->getString("currency");
00133 if (oCurr.set_curr(the_curr))
00134 {
00135 theSum += oCurr.to_curr(db.getFloat("amount"), this->
currency);
00136 //theSum += db.getFloat("amount");
00137 nbr_assets++;
00138 }
00139 else
00140 {
00141 error_message = error_message + "<li> ERRN3400: invalid currency</li>";
00142 theSum += db.getFloat("amount");
00143 }
00144 // cout << " || the " << nbr_assets << " first assets are " << theSum << this->currency << " worth.
XDB<BR>";
00145 }
00146 // cout << "<BR>the number of assets=" << nbr_assets;
00147 this->means.at(0) = theSum;
00148 }
00149
00155 void investment_problem::expand_cf(float amount, string freq, string fromStr,
string tillStr, int goal_nbr)
00156 {
00157 int m, step;
00158 int startMnth;
00159 int endMnth = this->months2simulate();
00160 struct tm tml; struct tm tm2;
00161 if (freq == "M") step = 1;
00162 else if (freq == "A") step = 12;
00163 else if (freq == "Q") step = 3;

```

```

00164     else if (freq == "S") step = 6;
00165     else if (freq == "W")
00166     {
00167         amount *= 30.4375/7; // note: 30.4375 = 365.25/
00168         step = 1;
00169     }
00170     else if (freq == "D")
00171     {
00172         amount *= 30.4375; // note: 30.4375 = 365.25/
00173         step = 1;
00174     }
00175     else step = 1;
00176     tml = get_tm();
00177     dateStr2tm(fromStr, &tm2);
00178     if (is_valid_dateStr(fromStr) && (difftime(mktime(&tml), mktime(&tm2)) < 0)) {startMnth
= dateStr2Mnbr(fromStr);} else {startMnth = 0;}
00179     if (is_valid_dateStr(tillStr)) {endMnth = dateStr2Mnbr(tillStr);} else {
endMnth = this->months2simulate();}
00180     m = startMnth;
00181     while (m <= endMnth && m <= this->months2simulate()) //just in case endMnth >
months2simulate
00182     {
00183         if (goal_nbr == 0)
00184         {
00185             means.at(m) += amount;
00186         }
00187         else
00188         {
00189             goalZ[goal_nbr].means_goal.at(m) += amount;
00190         }
00191         m += step;
00192     }
00193 //      cout << "<BR>expand_cf: from=" << startMnth << " | to_m=" << m << " | add_to_means=" <<
add_to_means<< " | step=" << step;
00194 }
00195
00202 void investment_problem::set_cfs()
00203 {
00204     string s = "", d1, d2, the_curr;
00205     float amnt;
00206     cls_currency oCurr;
00207     s = "SELECT * FROM " + oConfig.tbl_prefix + "_cash_flows WHERE investor = " +
std::to_string(this->investor_id) + ";";
00208     db.res = db.stmt->executeQuery(s);
00209     while (db.res->next())
00210     {
00211         d1 = db.res->getString("from_date");
00212         d2 = db.res->getString("till_date");
00213         the_curr = db.res->getString("currency");
00214         oCurr.set_curr(the_curr);
00215         amnt = oCurr.to_curr(atof(db.res->getString("amount").c_str()), this->
currency);
00216
00217         expand_cf(amnt, db.res->getString("frequency").c_str(), d1, d2);
00218         nbr_cfs++; // increment the number of cash flows so that at the end this reflects the number of
cash flows
00219     }
00220 }
00221
00225 void investment_problem::set_goals()
00226 {
00227     string s = "";
00228     int nbr = 0;
00229     s = "SELECT * FROM " + oConfig.tbl_prefix + "_goals WHERE investor = " + to_string(
this->investor_id) + " ORDER BY priority ASC;";
00230     this->set_priorityLimits();
00231     db.res = db.stmt->executeQuery(s);
00232 #ifdef DEBUG
00233     debug_info = debug_info + "<BR>goal_type_s2i[\"amount@date\"] = " + to_string(
goal_type_s2i["amount@date"]);
00234     debug_info = debug_info + "<BR>goal_type_s2i[\"income from/to\"] = " + to_string(
goal_type_s2i["income from/to"]);
00235 #endif
00236     while (db.res->next())
00237     {
00238         nbr++;
00239         //      this->goalZ[nbr] = aGOAL(investor_id); //create new and run constructor
00240         this->goalZ.insert(pair<int, aGOAL>(nbr, aGOAL(investor_id)));
00241         this->goalZ[nbr].set_from_db();
00242 #ifdef DEBUG
00243         debug_info = debug_info + "<BR>---goal: " + to_string(nbr);
00244         debug_info = debug_info + "<BR>goal_type = " + to_string(goalZ[nbr].goal_type);
00245 #endif
00246
00247 //cout << "<br>g=" << to_string(nbr) << " amount=" << goalZ[nbr].amount;
00248 //GOAL_TYPES: (4, "amount asap"),
00250     if (goalZ[nbr].goal_type == goal_type_s2i["unallocated"]) //(0, "unallocated"),

```

```

00251     {
00252         goalZ[ nbr ].realization_monthNbr = months2simulate();
00253         goalZ[ nbr ].target_amount      = 0;
00254     }
00255     else if ( goalZ[ nbr ].goal_type == goal_type_s2i["amount@date"] ) //(1, "amount@date")
00256     {
00257         goalZ[ nbr ].realization_monthNbr = age2monthNbr( goalZ[ nbr ].realization_age );
00258         goalZ[ nbr ].target_amount      = goalZ[ nbr ].amount;
00259     }
00260     else if ( goalZ[ nbr ].goal_type == goal_type_s2i["income from/to"] ) // (2, "income
from/to")
00261     {
00262         goalZ[ nbr ].realization_monthNbr = dateStr2Mnbr( goalZ[ nbr ].till_date );
00263         goalZ[ nbr ].target_amount      = 0;
00264         // debug_info = debug_info + "<BR>goalZ[ nbr ].realization_monthNbr" +
to_string( goalZ[ nbr ].realization_monthNbr );
00265         // debug_info = debug_info + "<BR>goalZ[ nbr ].target_amount" + to_string( goalZ[ nbr ].target_amount );
00266     }
00267     else if ( goalZ[ nbr ].goal_type == goal_type_s2i["rainy day savings"] ) // (3, "rainy
day savings")
00268     {
00269         goalZ[ nbr ].realization_monthNbr = RAINY_DAY_MONTH_NBR;
00270         goalZ[ nbr ].target_amount      = goalZ[ nbr ].amount;
00271     }
00272     else if ( goalZ[ nbr ].goal_type == goal_type_s2i["amount asap"] ) // (4, "amount
asap")
00273     {
00274         goalZ[ nbr ].realization_monthNbr = months2simulate();
00275         goalZ[ nbr ].target_amount      = goalZ[ nbr ].amount;
00276     }
00277     else // assume that it is an amount@date type of goal
00278     {
00279         goalZ[ nbr ].realization_monthNbr = age2monthNbr( goalZ[ nbr ].realization_age );
00280         goalZ[ nbr ].target_amount      = goalZ[ nbr ].amount;
00281     }
00282     this->goalZ[ nbr ].alpha              = calc_alpha( this->goalZ[ nbr ].priority, this->
goalZ[ nbr ].realization_monthNbr );
00283     // cout << "<br>goalZ[" << nbr <<"].alpha = " << goalZ[ nbr ].alpha;
00284
00285     set_means_goal( nbr );
00286
00289     if ( goalZ[ nbr ].realization_monthNbr > months2simulate() )
goalZ[ nbr ].realization_monthNbr = months2simulate();
00290 }
00291 this->nbr_goals = nbr;
00292 }
00293 }
00294
00300 void investment_problem::set_priorityLimits()
00301 {
00302     string s = "";
00303     s = "SELECT MIN(priority) as theMin, MAX(priority) as theMax FROM " + oConfig.
tbl_prefix + "_goals WHERE investor = " + to_string( this->investor_id ) + " AND
goal_type != " + to_string( goal_type_s2i["unallocated"] ) + ";";
00304     db.res = db.stmt->executeQuery( s );
00305     while ( db.res->next() )
00306     {
00307         this->priorityMax = atoi( db.res->getString( "theMax" ).c_str() );
00308         this->priorityMin = atoi( db.res->getString( "theMin" ).c_str() );
00309     }
00310 }
00311
00312
00318 inline float investment_problem::calc_alpha( int priority, int monthNbr )
00319 {
00320     float x;
00321     if ( priorityMin != priorityMax )
00322     {
00323         x = MIN_ALPHA + ( MAX_ALPHA - MIN_ALPHA ) * ( priority -
priorityMin ) / ( priorityMax - priorityMin );
00324     }
00325     else
00326     {
00327         x = 0.01;
00328     }
00329     //xx cout << "[" << x;
00330     if ( x < ALPHA_UPPER_LIMIT ) x += ( ALPHA_UPPER_LIMIT - x ) * ( 1 - exp(-
monthNbr / 12 / 35) );
00331     //xx cout << " --> " << x ;
00332     if ( x > ALPHA_LOWER_LIMIT ) x -= ( x - ALPHA_LOWER_LIMIT ) * exp(-
monthNbr / 12 / 5 );
00333     //xx cout << " --> " << x << "]" ;
00334     return x;
00335 }
00336
00337

```

```

00341 void investment_problem::set_portfolios()
00342 {
00343     int nbr = 0, count = 0, p, ac;
00344     bool is_suitable;
00345     string s = "SELECT portfolio_id FROM " + oConfig.tbl_prefix + "_portfolios ORDER BY
portfolio_id";
00346     db.res = db.stmt->executeQuery(s);
00347     while (db.res->next())
00348     {
00349         nbr = db.res->getInt("portfolio_id");
00350         if (this->portfolios[nbr].get_portf_from_db(nbr)) //success retrieving info
00351         {
00352             this->portfolios[nbr].set_mu(this->assetClass_mu); //sets pMu
00353             this->portfolios[nbr].set_sigma(this->assetClass_covar); // sets pSigma
00354             //xxx cout << "<br>portf[" << nbr << "] ER=" << exp(12 * portfolios[nbr].pMu) - 1 << " S=" <<
sqrt(12) * portfolios[nbr].pSigma;
00355             count++;
00356         }
00357     }
00359     /* not needed any more because portfolio_suitable is a C++ map now and not a C array any longer
00360     bool* new_block = NULL;
00361     new_block = (bool*) realloc(portfolio_suitable, nbr_portfolios * sizeof(bool));
00362     if (new_block != NULL)
00363     {
00364         portfolio_suitable = new_block;
00365     }
00366     else
00367     {
00368         free(portfolio_suitable);
00369 #ifdef DEBUG
00370         debug_info = debug_info + "<br>realloc() failed in portfolio_suitable()";
00371 #endif
00372     }
00373     */
00374
00375     // set the number of portfolios (note: already used in the next lines!)
00376     this->nbr_portfolios = count;
00377
00378     // now set the portfolio_suitable map
00379     load_preferences(); // to set the map set_max_exposure (inherited from the investor
class)
00380     for (p = 1; p <= nbr_portfolios; p++)
00381     {
00382         is_suitable = true;
00383         for (ac = 1; ac <= NBR_ASSET_CLASSES; ac++)
00384         {
00385             //(portfolios[p].weights[ac-1] <= max_exposure[ac])?portfolio_suitable[p-1] = true :
portfolio_suitable[p-1] = false;
00386             if (portfolios[p].weights[ac-1] > max_exposure[ac]) is_suitable = false;
00387         }
00388         portfolio_suitable[p] = is_suitable;
00389     }
00390 }
00391 }
00392
00396 void investment_problem::solve()
00397 {
00398     int g = 1, m, k, freeFrom, unallocated_goal_nbr = 0;
00399     float opt_perc, amnt_used, currency_conversion;
00400     cls_currency g_currency;
00401
00403     //cout << " ### months2simulate=" << to_string(months2simulate());
00404     //cout << " ### goalZ[g].realization_monthNbr=" << to_string(goalZ[g].realization_monthNbr);
00405
00406
00407     for(m=0;m<=goalZ[g].realization_monthNbr;m++)
00408     {
00409         means_left.at(m) = means.at(m);
00410         //cout << " | means.at(" << m << ")=" << means.at(m);
00411     }
00412
00413     for (g = 1; g <= this->nbr_goals; g++)
00414     {
00415
00416         /*
00417         //TODO: prepare covar, E_R and eventually portfolios for the relevant goal-currency
00418         set_mu(goalZ[g].currency);
00419         set_covar(goalZ[g].currency);
00420         set_portfolios(goalZ[g].currency);
00421         */
00422
00423         // convert all relevant means_left to the goal-currency
00424         g_currency.set_curr(this->currency);
00425         currency_conversion = g_currency.to_curr(1.0, goalZ[g].
currency);
00426         if (currency_conversion != 0)
00427     {

```

```

00428     for(m=0;m<=goalZ[g].realization_monthNbr;m++)
00429     {
00430         means_left.at(m) = means_left.at(m) * currency_conversion;
00431     }
00432     }
00433     else
00434     { // if the currency_conversion equals zero, then we convert it to one (to avoid dividing by 0 later)
00435         currency_conversion = 1;
00436     }
00437
00438     if (goalZ[g].goal_type == goal_type_s2i["unallocated"])
00439     {
00440         unallocated_goal_nbr = g;
00441         continue;
00442     }
00443     freeFrom = get_max_month_for_lower_goals(g);
00444
00445
00446 /* TODO: refine the allocation for from/to goals
00447     if (goalZ[g].goal_type == goal_type_s2i["income from/to"])
00448     {
00449     }
00450     }
00451 */
00452     for(m=0; m<=goalZ[g].realization_monthNbr; m++)
00453     {
00454         means_used[m] = 0;
00455     }
00456
00457 //xx     set_means_goal(g); // calculate the means that are implied by the goal (eg as income "from/till"
00458     if ((goalZ[g].goal_type == goal_type_s2i["amount asap"]) || (
00459         goalZ[g].goal_type == goal_type_s2i["rainy day savings"]))
00460     {
00461         //for (k = 0; k < months2simulate(); k++) means_block[k] = 0;
00462         k = 0;
00463
00464         do
00465         {
00466             means_block[k] = means_left[k];
00467             goalZ[g].realization_monthNbr = k;
00468             if (k > 0)
00469             {
00470                 means_used[k-1] = means_block[k-1];
00471                 means_block[k-1] = 0;
00472             }
00473             opt_perc = get_optimal_portfolio(g);
00474         } while ((opt_perc > 1 - PRECISION) && (k < months2simulate()));
00475
00476 /*
00477     means_block[0] = means_left[0];
00478     goalZ[g].realization_monthNbr = 0;
00479     opt_perc = get_optimal_portfolio(g);
00480     for (k = 1; (k < months2simulate()) && (opt_perc > 1 - PRECISION); k++) //realization_monthNbr =
00481         months2simulate()-1
00482         {
00483             goalZ[g].realization_monthNbr = k;
00484             means_used[k-1] = means_block[k-1];
00485             means_block[k-1] = 0;
00486             means_block[k] = means_left[k];
00487             opt_perc = get_optimal_portfolio(g);
00488         }
00489 */
00490     }
00491     else
00492     {
00493         for (k = freeFrom; k <= goalZ[g].realization_monthNbr; k++)
00494             means_block[k] = means_left[k];
00495         opt_perc = get_optimal_portfolio(g);
00496
00497         if (opt_perc > 1 - PRECISION)
00498         {
00499             //means_used = means_block since we're only here if opt_perc == 1
00500             for (k = freeFrom; k <= goalZ[g].realization_monthNbr; k++)
00501             {
00502                 means_used[k] = means_block[k];
00503                 means_block[k] = 0;
00504             }
00505             for (k = 0; k < freeFrom;k++) means_block[k] = means_left[k];
00506 //xx     cout << "<br>====[GOAL " << g << "]"====<br>";
00507             opt_perc = get_optimal_portfolio(g);
00508 //xx     cout << "<br>====[end]====<br>";
00509         }
00510     }
00511
00512     for (k = 0; k <= goalZ[g].realization_monthNbr; k++)

```

```

00518     {
00519         amnt_used = means_used[k] + (means_block[k] * opt_perc);
00520         // if(amnt_used > 0)
00521         // {
00522         means_left[k] = (means_left[k] - amnt_used) / currency_conversion;
00523         goalZ[g].saving_plan[k] = amnt_used;
00524
00525         // }
00526     }
00528     for (k = 1; k <= NBR_ASSET_CLASSES; k++)
00529     {
00530         goalZ[g].benchmark[k] = portfolios[goalZ[g].optimal_portf].weights[k-1];
00531     }
00532     set_goal_color(g, opt_perc);
00533     set_goal_remarks(g);
00534
00535     //goalZ.save_results(); //TODO
00536     } //(g = 1; g <= this->nbr_goals; g++)
00537
00539     allocate_to_unallocated_goal(unallocated_goal_nbr);
00540 }
00541
00542
00546 void investment_problem::set_goal_color(int g, float opt_perc)
00547 {
00548     if (opt_perc < 1 - PRECISION)
00549     {
00550         goalZ[g].success = true;
00552         if (flSum(&means_left, goalZ[g].realization_monthNbr) > 0.1 *
00553             flSum(&goalZ[g].saving_plan, goalZ[g].realization_monthNbr))
00554         {
00555             goalZ[g].remarks = t_feedback["brightgreen"];
00556             goalZ[g].color = "brightgreen";
00557         }
00558         else
00559         {
00560             goalZ[g].remarks = t_feedback["green"];
00561             goalZ[g].color = "green";
00562         }
00563     } else // opt_perc = 1
00564     {
00565         goalZ[g].success = false;
00566         //cout << "<br>flSum(&goalZ["<<g<<"].saving_plan,
00567             goalZ[g].realization_monthNbr)="<<flSum(&goalZ[g].saving_plan, goalZ[g].realization_monthNbr);
00568         if (flSum(&goalZ[g].saving_plan, goalZ[g].realization_monthNbr) > 0)
00569         {
00570             goalZ[g].remarks = t_feedback["red"];
00571             goalZ[g].color = "red";
00572         }
00573         else
00574         {
00575             goalZ[g].remarks = t_feedback["darkred"];
00576             goalZ[g].color = "darkred";
00577         }
00578         if (calc_V_high(g) >= goalZ[g].amount - goalZ[g].max_shortfall)
00579         {
00580             goalZ[g].remarks = t_feedback["amber"];
00581             goalZ[g].color = "amber";
00582         }
00583     }
00584
00588 void investment_problem::set_goal_remarks(int g)
00589 {
00590     goalZ[g].remarks = "<u>" + goalZ[g].remarks + "</u>"
00591         + "<br>The portoflio corresponding to this benchmark is called " +
00592         portfolios[goalZ[g].optimal_portf].description + "."
00593         + "<br>The shortfall of this portfolio with the given savings plan is estimated to be "
00594         + curr_format(goalZ[g].realized_shortfall, currency) + " with a
00595         probability of "
00596         + to_string((int)((1-goalZ[g].alpha)*10000)/100) + "%."
00597         + "<br>The sum of all savings (not discounted) is " +
00598         curr_format(flSum(&goalZ[g].saving_plan, goalZ[g].realization_monthNbr),
00599             currency) + ".";
00600 }
00601
00604 void investment_problem::allocate_to_unallocated_goal(int
00605     unallocated_goal_nbr)
00606 {
00607     int k, optimal_portf;
00608     if (unallocated_goal_nbr == 0) // if it was not encountered above, then create it
00609     {
00610         unallocated_goal_nbr = ++nbr_goals;
00611         this->goalZ.insert(pair<int, aGOAL>(nbr_goals, aGOAL(
00612             investor_id)));
00613         goalZ[nbr_goals].description = "Unallocated Resources";

```

```

00612 }
00613
00614 if (set_unallocated_goal())
00615 {
00616     // set the saving_plan:
00617     for (k = 0; k <= months2simulate(); k++)
00618 //x.x.x     for (k = 1; k <= 10; k++)
00619     {
00620         goalZ[unallocated_goal_nbr].saving_plan[k] = means_left[k];
00621     }
00622     // set the optimal portfolio
00623     optimal_portf = get_portfolio_riskiest();
00624
00625     // set optimal_portf and alpha
00626     goalZ[unallocated_goal_nbr].optimal_portf = optimal_portf;
00627     goalZ[unallocated_goal_nbr].alpha = (MAX_ALPHA + MIN_ALPHA) / 2;
00628
00629     // set the benchmark
00630     for (k = 1; k <= NBR_ASSET_CLASSES; k++)
00631     {
00632         goalZ[unallocated_goal_nbr].benchmark[k] = portfolios[optimal_portf].weights[k-1];
00633     }
00634     //set color and comments
00635     goalZ[unallocated_goal_nbr].remarks = goalZ[unallocated_goal_nbr].remarks + "This is not
really a &quot;goal&quot;."
00636         + "This is simply all the assets that are not or cannot be used by other goals.<br>"
00637         + "The non-discounted sum of this savings is: "
00638         + curr_format(flSum(&goalZ[unallocated_goal_nbr].saving_plan,
months2simulate()), currency) ;
00639     goalZ[unallocated_goal_nbr].color = "brightgreen";
00640 }
00641 else
00642 {
00643     error_message = error_message + "<li>Failed to create the unallocated goal,
so after all the goals mentioned here, some means might be left</li>";
00644 }
00645 }
00646
00654 void investment_problem::set_means_goal(int g)
00655 {
00656     int m;
00657     // for ...:     goalZ[g].means_goal.insert (std::pair<int,float>(i, 0.0) );
00658     for (m = 0; m <= goalZ[g].realization_monthNbr; m++) goalZ[g].means_goal[m] = 0;
00659     if (goalZ[g].goal_type == goal_type_s2i["income from/to"]) // only for "income
from/till"
00660     {
00662         expand_cf(goalZ[g].amount, goalZ[g].frequency, goalZ[g].from_date,
goalZ[g].till_date, g);
00663     }
00668 }
00669
00670
00677 float investment_problem::get_optimal_portfolio(int g)
00678 {
00679     int p, optimal_portf = 0;
00680     float opt_perc = 1; // initialize to one to make sure it is replaced
00681     float* perc_bloc_used = NULL;
00682     float* new_block = NULL;
00683     new_block = (float*) realloc(perc_bloc_used, nbr_portfolios * sizeof(float));
00684     if (new_block != NULL)
00685     {
00686         perc_bloc_used = new_block;
00687     }
00688     else
00689     {
00690         free(perc_bloc_used);
00691 #ifdef DEBUG
00692         debug_info = debug_info + "<br/>realloc() failed in get_optimal_portfolio()";
00693 #endif
00694         return false;
00695     }
00696
00698     for (p = 1; p <= nbr_portfolios; p++)
00699     {
00700         // do this only for suitable portfolios
00701         if (portfolio_suitable[p])
00702         {
00703             perc_bloc_used[p-1] = goal_seek_means(g, p);
00704         }
00705         else
00706         {
00707             perc_bloc_used[p-1] = opt_perc + 1; // in order to exclude this portfolio automatically in the next
lines
00708         }
00709     }
00710
00712     for (p = 1; p <= nbr_portfolios; p++)

```

```

00713 {
00714 //xx cout << "<br>portf="<<p<<" , opt_perc: " << opt_perc << " | perc_bloc_used[" << p-1 << " ]:" <<
    perc_bloc_used[p-1];
00715     if (perc_bloc_used[p-1] <= opt_perc)
00716     {
00717         opt_perc = perc_bloc_used[p-1];
00718         optimal_portf = p;
00719     }
00720 }
00721 if (goalZ[g].realization_monthNbr <= MAX_MONTHS_TO_SAFEST)
00722 {
00723     goalZ[g].optimal_portf = get_portfolio_safest();
00724 }
00725 else
00726 {
00727     goalZ[g].optimal_portf = optimal_portf;
00728 }
00729
00730 //cout << "<br>g="<<to_string(g)<< " optimal_portf="<<to_string(goalZ[g].optimal_portf);
00731
00732 goalZ[g].realized_shortfall = calc_risk(g, optimal_portf);
00733 return opt_perc;
00734 }
00735
00745 float investment_problem::goal_seek_means(
00746     int g,
00747     int p
00748 )
00749 {
00750     int m;
00751     float perc = 0.5; // the perecentage used of this block
00752
00753     for(m = 0; m <= goalZ[g].realization_monthNbr; m++) means_tmp[m] = 0;
00754     if ((calc_risk(g, p) <= goalZ[g].max_shortfall) || (goalZ[g].realization_monthNbr < 0)
00755 )
00756     {
00757         return 0.0;
00758     }
00759
00760     for(m = 0; m <= goalZ[g].realization_monthNbr; m++) means_block[m] =
00761     means_block[m];
00762     if (calc_risk(g, p) >= goalZ[g].max_shortfall)
00763     {
00764         return 1.0;
00765     }
00766
00767
00768
00769
00770
00771 //xx for(m=0;m<=goalZ[g].realization_monthNbr;m++) means_tmp[m] = perc * means_block[m];
00772     goal_seek_tmp_means(&perc, g, p);
00773     return perc;
00774 }
00775
00779 inline void investment_problem::goal_seek_tmp_means(float *perc, int
    g, int p)
00780 {
00781     int k, m;
00782     for (k = 2; k <= NBR_ITERATIONS; k++)
00783     {
00784         if ((goalZ[g].goal_type == goal_type_s2i["amount asap"]) || (
00785             goalZ[g].goal_type == goal_type_s2i["rainy day savings"]))
00786         {
00787             means_tmp[goalZ[g].realization_monthNbr] = (*perc) *
00788             means_block[goalZ[g].realization_monthNbr];
00789         }
00790         else
00791         {
00792             for(m = 0; m <= goalZ[g].realization_monthNbr; m++) means_tmp[m] = (*perc) *
00793             means_block[m];
00794         }
00795         // for(m=1;m<=goalZ[g].realization_monthNbr;m++) cout << " ["<< m << "]" << means_tmp[m];
00796         // cout << "<br>perc=" << to_string(*perc) << " | calc_risk(g, p)=" << to_string(calc_risk(g, p)) << "
00797         | goalZ[g].max_shortfall" << to_string(goalZ[g].max_shortfall);
00798         if (calc_risk(g, p) > goalZ[g].max_shortfall)
00799         {
00800             *perc += pow(2,-k);
00801         }
00802         else
00803         {
00804             *perc -= pow(2,-k);
00805         }
00806     }
00807 }
00808
00809 float investment_problem::calc_risk(
00810     int g,
00811     int p
00812 )
00813 {
00814     float V_min,

```

```

00815     V,
00816     delta,
00817     s = 0,
00818     mnt_avg = 0;
00819     float const my_errf = erfinv(2 * goalZ[g].alpha - 1);
00820     int m;
00821
00822     V_min = V = delta = means_used[0] + means_tmp[0] - goalZ[g].means_goal[0];
00823     s = portfolios[p].pSigma;
00824     for (m = 1; m <= goalZ[g].realization_monthNbr; m++)
00825     {
00826         delta = means_used[m] + means_tmp[m] - goalZ[g].means_goal[m];
00827         mnt_avg++;
00828         if (abs(V) + abs(delta) != 0)
00829         {
00830             mnt_avg = (abs(V) * (mnt_avg) + abs(delta) * 1) / (abs(V) + abs(delta));
00831         }
00832         V = V * exp(portfolios[p].pLogR + pow(s, 2) / 2) + delta;
00833         V_min = V_min * exp(portfolios[p].pLogR + sqrt(2) * s * (sqrt(mnt_avg) - sqrt(mnt_avg-1
00834     )) * my_errf) + delta;
00835     }
00836     if (oConfig.riskFunction == "VaR") {return (goalZ[g].target_amount - V_min);}
00837     else {return (goalZ[g].target_amount - V_min);}
00838 }
00839
00840
00841 /* =====not used any more =====
00842 * turns the relevant risk measure for given goal and portfolio for means_block
00843 *
00844 * riskFunction = the risk function to use : ES_normdist
00845 */
00846 /*
00847 inline float investment_problem::get_risk(const char* riskFunction, int theGoal, int thePortfolio)
00848 {
00849     if (strcmp(riskFunction,"ES_normdist") == 0) return ExpectedShortfall_normDist(theGoal, thePortfolio);
00850     //else if (
00851     else return 0;
00852 }
00853 */
00854 /*float portfolio::get_risk(
00855     float alpha,                //!< the confidence level (small)
00856     float (*riskFunction)(float alpha, float (*AssetClass_mu), float (*AssetClass_varCov)),    //!<
00857     pointer to the risk function to use
00858     float (*AssetClass_mu),      //!<< pointer to the array of E[R] of the asset classes
00859     float (*AssetClass_varCov)  //!<< pointer to the two dimensional array of sigma[n][m]
00860 {
00861     return (*riskFunction) (alpha, AssetClass_mu, AssetClass_varCov);
00862 }
00863 */
00864 /* TODO: improve the previous as follows:
00865 * td::function<bool()> my_fun;
00866
00867 if (condition1)
00868     my_fun = std::bind(&MyClass::function_one, my_class);
00869 else if (condition2)
00870     my_fun = std::bind(&MyClass::function_two, my_class, a, b);
00871 else if (condition3)
00872     my_fun = std::bind(&MyClass::function_three, my_class, a, b, c);
00873 else if (condition4)
00874     my_fun = std::bind(&MyClass::function_four, my_class, a, b, c, d);
00875
00876 while (my_fun())
00877 { ... }
00878 */
00879
00880 /* OR:
00881 * or the sake of interest, there's also a simple lo-tech solution from the C world that as far as it goes,
00882 * works in C++. Instead of allowing arbitrary parameters, define the function as void (*func)(void*), and
00883 * make "params" void*. It's then the caller's job to define some struct that will contain the parameters, and
00884 * manage its lifecycle. Usually the caller would also write a simple wrapper to the function that's really needed
00885 * to be called:
00886
00887 void myfunc(int, float); // defined elsewhere
00888
00889 typedef struct {
00890     int foo;
00891     float bar;
00892 } myfunc_params;
00893
00894 void myfunc_wrapper(void *userdata) {
00895     myfunc_params *params = (myfunc_params *)p;
00896     myfunc(p->foo, p->bar);
00897 }
00898
00899 int main() {

```

```

00896     myfunc_params x = {1, 2};
00897     AddTimer(23, 5, myfunc_wrapper, &x);
00898     sleep(23*5 + 1);
00899 }*/
00900
00901 /* =====[not used any more]=====
00902 * returns the expected shortfall for the GAUSSIAN distribution for given alpha using
00903 *
00904 * ES = mu + a sigma
00905 *
00906 * with
00907 *
00908 * a = (1/alpha sqrt(2 Pi) exp {-[erffinv(2 alpha -1)]^2})
00909 */
00910 /*
00911 float investment_problem::ExpectedShortfall_normDist(int theGoal, int thePortfolio)
00912 {
00913     int k = 1;
00914     float EV, Sigma, a;
00915     EV = means_block[k] + means_used[k];
00916     Sigma = portfolios[thePortfolio].pSigma * EV;
00917     while (k <= this->goalZ[theGoal].realization_monthNbr)
00918     {
00919         EV = EV * (1 + portfolios[thePortfolio].pMu) + means_block[k] + means_used[k];
00920         Sigma = sqrt( 2 * Sigma * Sigma + pow((means_block[k] + means_used[k]) *
00921 portfolios[thePortfolio].pSigma, 2));
00922         k++;
00923     }
00924     // calculate a:
00925     a = fast_erffinv(2 * goalZ[theGoal].alpha - 1);
00926     a = exp(- pow(a,2));
00927     a = (1 / (2 * goalZ[theGoal].alpha * pow(Pi,0.5))) * a;
00928
00929     return (- EV + a * Sigma);
00930 }
00931 */
00932
00936 int investment_problem::get_nbr_goals()
00937 {
00938     return nbr_goals;
00939 }
00940
00945 int investment_problem::get_max_month_for_lower_goals(int
g)
00946 {
00947     int k, max_month = 0;
00948     if (g < nbr_goals)
00949     {
00950         for (k = g + 1; k <= this->nbr_goals; k++)
00951         {
00952             if (goalZ[k].goal_type != 0 && goalZ[k].realization_monthNbr > max_month) {max_month =
00953 goalZ[k].realization_monthNbr;}
00954         }
00955     }
00956     return max_month;
00957 }
00958
00970 void investment_problem::prepare_javaVars_colors(int g, int
followup_mnth)
00971 {
00972     int k;
00973     string s1, s2, s3, s4, comma;
00974     float Vlow, Vhigh, Vmed, Vexp, Vexp_evol,
00975         sigma, the_diff, age_now = age(0), age_at_cutoff, diff_cum, erf_low, erf_high, erf_med, age_at_k
,
00976         mnt_avg = 0,
00977         mnth_max = (goal_type_i2s[goalZ[g].goal_type] == "rainy day savings") ?
months2simulate() : goalZ[g].realization_monthNbr;
00978     int p = goalZ[g].optimal_portf;
00979     s1 = s2 = s3 = s4 = "";
00980     sigma = portfolios[p].pSigma ;
00981     erf_low = erffinv(2 * goalZ[g].alpha - 1);
00982     erf_high = erffinv(2 * (1-goalZ[g].alpha) - 1);
00983     erf_med = erffinv(0);
00984     set_means_goal(g);
00985
00986     if (followup_mnth == 0)
00987     { // in this case we are making a feedback or a Simulation, thus the tricolor graph starts from month 0,
00988         independent from the simulation
00989         Vlow = Vhigh = Vmed = diff_cum = Vexp = goalZ[g].saving_plan[0] - goalZ[g].means_goal[0];
00990     }
00991     else // in this case we are producing the Feedback screen and hence we start the tricolor plot at the end
00992     of the simulation
00993     {
00994         Vlow = Vhigh = Vmed = diff_cum = Vexp = goalZ[g].saving_plan[followup_mnth] - goalZ[g].means_goal[
00995 followup_mnth] + goalZ[g].simulation_endvalue;

```

```

00993     }
00994     age_at_cutoff = age_now + (float)followup_mnth / 12;
00995     // cout << "<br> goalZ[" << to_string(g) << "] saving_plan[0] " << to_string(goalZ[g].saving_plan[0]) << "
-- goalZ[g].means_goal[0] = " << to_string(goalZ[g].means_goal[0]);
00996     if ( followup_mnth < mnth_max)
00997     {
00998         s1 = "[" + to_string(age_at_cutoff) + ", " + to_string(Vexp) + "]";
00999         s2 = "[" + to_string(age_at_cutoff) + ", " + to_string(Vexp) + "]";
01000         s3 = "[" + to_string(age_at_cutoff) + ", " + to_string(Vexp) + "]";
01001     }
01002     else
01003     {
01004         s1 = s2 = s3 = "[]";
01005     }
01007
01008     for (k=followup_mnth + 1; k <= mnth_max; k++)
01009     {
01010         // Vexp_sign = (Vexp >= 0) ? 1.0 : -1.0;
01011         // Vexp = Vexp * Vexp_sign; // ==> is now always positive
01012         Vexp_evol = Vexp * exp(portfolios[p].pLogR + pow(portfolios[p].
pSigma, 2) / 2);
01013         // sigma = sqrt(pow(sigma, 2) + pow(portfolios[p].pSigma * Vexp,2));
01014         the_diff = goalZ[g].saving_plan[k] - goalZ[g].means_goal[k];
01015         diff_cum += the_diff;
01016         // Vexp = Vexp_sign * Vexp_evol + the_diff; //set back to positive or negative
01017         Vexp = Vexp_evol + the_diff; //set back to positive or negative
01018         mnt_avg++;
01019         if (abs(Vexp) + abs(the_diff) != 0)
01020         {
01021             mnt_avg = (abs(Vexp) * (mnt_avg) + abs(the_diff) * 1) / (abs(Vexp) + abs(the_diff));
01022         }
01023         // cout << "<br>VLow=" << to_string(Vlow) << " -- sigma=" << to_string(sigma) << "-- exp(portfolios["
<< to_string(p) << "].pLogR)= " << to_string(exp(portfolios[p].pLogR)) << " -- exp...=" <<
exp(portfolios[p].pLogR +sqrt(2) * (sqrt(mnt_avg) - sqrt(mnt_avg-1)) * sigma * erf_low);
01024
01025         Vlow = Vlow * exp(portfolios[p].pLogR +sqrt(2) * (sqrt(mnt_avg) - sqrt(mnt_avg-1)) *
sigma * erf_low) + the_diff;
01026         Vmed = Vmed * exp(portfolios[p].pLogR +sqrt(2) * (sqrt(mnt_avg) - sqrt(mnt_avg-1)) *
sigma * erf_med) + the_diff;
01027         Vhigh = Vhigh * exp(portfolios[p].pLogR +sqrt(2) * (sqrt(mnt_avg) - sqrt(mnt_avg-1)) *
sigma * erf_high) + the_diff;
01028         Vexp = Vexp_evol + the_diff; //set back to positive or negative
01029         age_at_k = age_now + (float)k / 12;
01030         s1 = s1 + ", [" + to_string(age_at_k) + ", " + to_string(Vlow) + "]";
01031         #ifdef SHOW_Vexp_not_Vmed
01032         s2 = s2 + ", [" + to_string(age_at_k) + ", " + to_string(Vexp) + "]";
01033         #else
01034         s2 = s2 + ", [" + to_string(age_at_k) + ", " + to_string(Vmed) + "]";
01035         #endif
01036         s3 = s3 + ", [" + to_string(age_at_k) + ", " + to_string(Vhigh) + "]";
01037     }
01038     //s4: for the rainy day savings a line, for all other goals a big dot
01039     if (goalZ[g].goal_type == goal_type_s2i["rainy day savings"])
01040     {
01041         s4 = s4 + "[" + to_string(age_now) + ", " + to_string(goalZ[g].target_amount) + "]";
01042         for (k=1; k <= mnth_max; k++)
01043         {
01044             age_at_k = age_now + (float)k / 12;
01045             s4 = s4 + ", [" + to_string(age_at_k) + ", " + to_string(goalZ[g].target_amount) + "]";
01046         }
01047     }
01048     else
01049     {
01050         s4 = "[" + to_string(age_now + ((float)goalZ[g].realization_monthNbr)/12) + ", " + to_string(goalZ[g].
target_amount) + "]";
01051     }
01052     // cout << "<div class=\"jqplot-target\" id=\"xchart\" << g << \"\" style=\"height: 300px; width: 500px;
position: relative;\">";
01053     // cout << "<script class=\"code\" type=\"text/javascript\">$(document).ready(function(){var plot1 =
$.jqplot ('xchart\" << g << \"\", [[3,7,9,1,5,3,8,2,5]]);});</script>";
01055
01057     goalZ[g].s_low = s1;
01058     goalZ[g].s_exp = s2;
01059     goalZ[g].s_high = s3;
01060     goalZ[g].s_goal = s4;
01061
01062     if (followup_mnth > 0)
01063     { // in this case we are in the "Follow-Up screen"
01064         if (followup_mnth + 1 > goalZ[g].realization_monthNbr)
01065         {
01066             // if the previous for-loop was not executed, so we are before
01067             set_followup_color_post(Vexp, g);
01068         }
01069         else
01070         {
01071             set_followup_color_prae(Vlow, Vexp, Vhigh, g);
01072         }

```

```

01073
01074 }
01075 }
01076
01077
01078
01089 void investment_problem::javaGraph(int g, string xtra_var, string xtra_label,
int followup_mnth)
01090 {
01091     prepare_javaVars_colors(g, followup_mnth);
01092
01093     parse_javaGraph( g, xtra_var, xtra_label);
01094 }
01095
01103 void investment_problem::parse_javaGraph(int g, string xtra_var, string
xtra_label)
01104 {
01105     string xtra_var_name;
01106     cout << "<div id=\"chartOverview\"<<g<<\" \" style=\"height:400px; width:700px;\"></div>";
01107     cout << "<script class=\"code\" type=\"text/javascript\">$(document).ready(function(){ \n";
01108     cout << "var l1 = [\" + this->goalZ[g].s_low + \"]; \n";
01109     cout << "var l2 = [\" + this->goalZ[g].s_exp + \"]; \n";
01110     cout << "var l3 = [\" + this->goalZ[g].s_high + \"]; \n";
01111     cout << "var l4 = [\" + this->goalZ[g].s_goal + \"]; \n";
01112     if (xtra_var != "")
01113     {
01114         cout << "var l5 = [\" + xtra_var + \"]; ";
01115         xtra_var_name = ", l5";
01116     }
01117
01118     /* cout << "var l1 = [[1,3],[2.5,4],[3,5],[5,7]]; \n";
01119     cout << "var l2 = [[1,3],[2.5,6],[3,6],[5,6]]; \n";
01120     cout << "var l3 = [[1,3],[2.5,3],[3,4],[5,6]]; \n";
01121     cout << "var l4 = [[1,3]]; \n";*/
01122     cout << "var overviewPlot" << g << " = var" << g << " = $.jqplot ('chartOverview" << g << "'", [l1, l2,
l3, l4" << xtra_var_name << "]" ";
01123     cout << ", {";
01124     cout <<     "seriesDefaults: {";
01125     cout <<         "showMarker: false ,";
01126     cout <<         "rendererOptions: { smooth: false },";
01127     cout <<         "markerOptions: { style:\"o\", size:5 }, ";
01128     cout <<         "lineWidth:7 ";
01129     cout <<     "},";
01130     cout <<     "legend: {show: true, location: 'e', fontSize: '2em', placement: 'outside', marginRight: \"
600px\", ";
01131     cout <<         "labels:['<<t_plotlabels_evolution[1]<<\", '<<
t_plotlabels_evolution[2]<<\", '<<t_plotlabels_evolution[3]<<\", '<<
t_plotlabels_evolution[4]<<g<<\"";
01132     if (xtra_var != "") cout << ", '<< xtra_label << \"";
01133     cout <<     "}],";
01134     // cout <<     "markerOptions: { style:\"o\", size:5 }, lineWidth:1}";
01135
01136     /* ////////////////
01137     if (goalZ[g].goal_type != goal_type_s2i["rainy day savings"])
01138     { // for the goals the special big dot
01139         cout << "{markerOptions: { show: true, style: 'circle', size: 12, shadow: false, shadowDepth: 6,
shadowAlpha: 0.07, shapeRenderer: new $.jqplot.customMarkerRenderer() }}"; // old markerOptions:
style:\"circle\", size:17
01140     }
01141     else
01142     { // except for the rainy day savings: the simple black line
01143         cout << "{showMarker: false}";
01144     }
01145     if (xtra_var != "") cout << ", {markerOptions: { style:\"o\", size:5 }, lineWidth:1}";
01146     cout << "],";
01147
01148     cout <<     "series: [{}, {}, {}, "; // for the goal: something special=
01149     if (goalZ[g].goal_type != goal_type_s2i["rainy day savings"])
01150     { // the typical presentation of the goal:
01151         cout <<         "{showMarker: true, markerOptions: { style:\"filledCircle\", size:17}, shadowDepth:
6, shadowAlpha: 0.07, shapeRenderer: new $.jqplot.customMarkerRenderer() }}";
01152     }
01153     else
01154     {
01155         cout <<         "{showMarker: false}";
01156     }
01157     if (xtra_var != "")
01158     { // if there is a simulation to be added, provide here the customization of the plot
01159         cout <<         ", {showMarker: true, markerOptions: {style:\"filledCircle\", size:15 },
lineWidth:3}";
01160     }
01161     cout <<     "], ";
01162
01163     cout <<     "seriesColors: " << hh.seriesColorsGoalPlot << ", ";
01164     cout <<     "axesDefaults: {labelRenderer: $.jqplot.CanvasAxisLabelRenderer}";
01165     //cout << "axes: {xaxis: {label: \"age\" }, yaxis: {label: \"\" << currency << \"\"}, {tickOptions:
{formatter:tickFormatter}}"; // , pad: 0

```

```

01167     cout <<      "axes: {xaxis: {label: \"age\" }, yaxis: {label: \"\" << goalZ[g].currency << \"\}}\"; // ,
pad: 0
01168
01169     cout << \"}\";
01170     cout << \"); });\";
01171     cout << \"</script>\";
01172     return;
01173
01174 //150215 cout << \"var plot\" << g << \" = $.jqplot('chartOverview\" << g << \"\", [11, 12, 13, 14\" <<
xtra_var_name << \"], {\";
01175
01176
01177     //      axesDefaults: {
01178     //          pad: 1.05
01179     //      },
01180     /*          //
01181     // Use the fillBetween option to control fill between two
01182     // lines on a plot.
01183     //
01184     fillBetween: {
01185         // series1: Required, if missing won't fill.
01186         series1: 1,
01187         // series2: Required, if missing won't fill.
01188         series2: 2,
01189         // color: Optional, defaults to fillColor of series1.
01190         color: \"rgba(227, 167, 111, 0.7)\",
01191         // baseSeries: Optional. Put fill on a layer below this series
01192         // index. Defaults to 0 (first series). If an index higher than 0 is
01193         // used, fill will hide series below it.
01194         baseSeries: 0,
01195         // fill: Optional, defaults to true. False to turn off fill.
01196         fill: true
01197     },*/
01198     cout << \"seriesDefaults: {rendererOptions: { smooth: false }},\";
01199     cout << \"legend: {show: true, location: 'e', placement: 'outside', marginRight: \\\"600px\\\",\";
01200     cout << \"labels:['negative scenario', 'expected evolution', 'good scenario', 'YOUR GOAL #\"<<g<<\"\";
01201     if (xtra_var != \"\") cout << \", ' \" << xtra_label << \"'\";
01202     cout << \"]},\";
01203 // cout << \"seriesDefaults: {showMarker: false}, \";
01204
01205
01206     cout << \"series:[{showMarker: false},{showMarker:false,lineWidth:5},{showMarker: false},\";
01207
01208
01209
01210     if (goalZ[g].goal_type != goal_type_s2i[\"rainy day savings\"]
01211     { // for the goals the special big dot
01212         cout << \"{markerOptions: { show: true, style: 'circle', size: 12, shadow: false, shadowDepth: 6,
shadowAlpha: 0.07, shapeRenderer: new $.jqplot.customMarkerRenderer() } }\"; // old markerOptions:
style:\\\"circle\\\", size:17
01213     }
01214     else
01215     { // except for the rainy day savings: the simple black line
01216         cout << \"{showMarker: false}\";
01217     }
01218     if (xtra_var != \"\") cout << \", {markerOptions: { style:\\\"o\\\", size:5 }, lineWidth:1}\";
01219     cout << \"]},\";
01220
01221
01222     cout << \"seriesColors: \" << hh.seriesColorsGoalPlot << \", \";
01223     cout << \"axesDefaults: {labelRenderer: $.jqplot.CanvasAxisLabelRenderer},\";
01224     //cout << \"axes: {xaxis: {label: \"age\" }, yaxis: {label: \"\" << currency << \"\"}, {tickOptions:
{formatter:tickFormatter}}\"; // , pad: 0
01225     cout << \"axes: {xaxis: {label: \"age\" }, yaxis: {label: \"\" << goalZ[g].currency << \"\}}\"; // ,
pad: 0
01226     cout << \" } }); }); </script>\";
01227 }
01228
01234 inline void investment_problem::set_followup_color_prae(float
Vlow, float Vmed, float Vhigh, int g)
01235 {
01236     //NOTE Vhigh is not used now ...
01237     if (Vlow > this->goalZ[g].target_amount) this->goalZ[g].color_followup = \"green\";
01238     else if (Vmed > this->goalZ[g].target_amount) this->goalZ[g].color_followup = \"amber\";
01239     else
01240         this->goalZ[g].color_followup = \"red\";
01241 }
01246 inline void investment_problem::set_followup_color_post(float V,
int g)
01247 {
01248     if (V > this->goalZ[g].target_amount) this->goalZ[g].color_followup = \"
green\";
01249     else if (V > this->goalZ[g].target_amount * (1 - MIN_ALPHA)) this->
goalZ[g].color_followup = \"amber\";
01250     else
01251         this->goalZ[g].color_followup = \"red\";
01252 }
01253
01260 float investment_problem::calc_V_high(int g, int p)

```

```

01261 {
01262     int k;
01263     float V0, Vhigh, sigma, logR, Vexp;
01264     if (p == 0) p = goalZ[g].optimal_portf;
01265     Vhigh = Vexp = V0 = goalZ[g].saving_plan[0] - goalZ[g].means_goal[0]; //rm 1;
01266     sigma = portfolios[p].pSigma * V0;
01267
01268     for (k=1; k <= goalZ[g].realization_monthNbr; k++)
01269     {
01270         sigma = sqrt(pow(sigma, 2) + pow(portfolios[p].pSigma * Vexp, 2));
01271         Vexp = Vexp * exp(portfolios[p].pLogR + pow(portfolios[p].
pSigma, 2) / 2) + goalZ[g].saving_plan[k] - goalZ[g].means_goal[k];
01272         logR = log(Vexp / V0);
01273         Vhigh = V0 * exp(logR + sigma / Vexp * norminv(ALPHA_PLOT_HIGH));
01274     }
01275     return Vhigh;
01276 }
01277
01278
01279
01285 bool investment_problem::set_unallocated_goal()
01286 {
01287     bool success = true;
01288     aGOAL unalloc_goal(investor_id);
01289     // check if an unallocated goal already exists
01290     string s = "SELECT * FROM " + oConfig.tbl_prefix + "_goals WHERE investor = " +
to_string(investor_id)
01291         + " AND goal_type = " + to_string(goal_type_s2i["unallocated"]) + ";"
01292     try
01293     {
01294         db.res = db.stmt->executeQuery(s);
01295         if (db.res->next())
01296         {
01297             // db.res->previous();
01298             // while (res->next())
01299             // {
01300                 unalloc_goal.set_from_db();
01301             // }
01302         }
01303         else
01304         {
01306             // this->goal_id = res->getInt("goal_id");
01307             //done in the constructor: this->investor = investor_id;
01308             unalloc_goal.amount = 0;
01309             unalloc_goal.goal_type = goal_type_s2i["unallocated"];
01310             unalloc_goal.description = "unallocated means";
01311             unalloc_goal.realization_age = simulate_till_age;
01312             unalloc_goal.currency = currency;
01313             unalloc_goal.priority = 99;
01314             unalloc_goal.from_date = "";
01315             unalloc_goal.till_date = "";
01316             unalloc_goal.frequency = "0";
01317             unalloc_goal.max_shortfall = 0;
01318             if (!unalloc_goal.add_to_db())
01319             {
01320                 error_message = error_message + "<li>Failed to save the new unallocated
goal.</li>";
01321                 success = false;
01322             }
01323         }
01324     }
01325     catch (sql::SQLException &e)
01326     {
01327         #ifdef DEBUG
01328             debug_info = "<br>" + s + "<br>";
01329             debug_info = debug_info + "# ERR: SQLException in " + __FILE__ + "(" + __FUNCTION__
+ ") on line " + to_string(__LINE__) + "<br># ERR: " + e.what()
01330                 + " (MySQL error code: " + to_string(e.getErrorCode())
01331                 + ", SQLState: " + e.getSQLState() + ")<br>";
01332         #endif
01333         error_message = error_message + "<li>Sorry, I failed to load a goal.</li>";
01334         success = false;
01335     }
01336     return success;
01337 }
01338 }
01339
01340
01341
01345 int investment_problem::get_portfolio_safest()
01346 {
01347     int p, the_portfolio = 1;
01348     float sigma = portfolios[1].pSigma;
01349     for (p = 1; p <= nbr_portfolios; p++)
01350     {
01351         if (portfolio_suitable[p] && portfolios[p].
pSigma <= sigma)
01352         {

```

```

01353     sigma = portfolios[p].pSigma;
01354     the_portfolio = p;
01355     }
01356 }
01357 return the_portfolio;
01358 }
01359
01363 int investment_problem::get_portfolio_riskiest()
01364 {
01365     int p, the_portfolio = 1;
01366     float sigma = portfolios[1].pSigma;
01367     for (p =1; p <= nbr_portfolios; p++)
01368     {
01369         if (portfolio_suitable[p] && portfolios[p].
pSigma >= sigma)
01370         {
01371             sigma = portfolios[p].pSigma;
01372             the_portfolio = p;
01373         }
01374     }
01375     return the_portfolio;
01376 }

```

8.35 investor.class.cpp File Reference

Classes

- class [investor](#)

8.36 investor.class.cpp

```

00001
00011 class investor : public market
00012 {
00013 public:
00014     int     investor_id;
00015     string  user_name;
00016     string  first_name;
00017     string  last_name;
00018     string  password;
00019     string  currency;
00020     struct  tm  birth_date;
00021     float  simulate_till_age;
00022
00023     string  get_full_name();
00024     float  age(int Mnbr = 0);
00025     int    months2simulate();
00026     int    load_from_db(string investorID); // NOTE: no password check and does NOT load password
00027
00028     //friend ostream& operator<<(ostream& os, const investor& the_investor);
00029
00030
00031 protected:
00032     int    age2monthNbr(float theAge);
00033     string  dateStr2Age(string the_date);
00034     bool   save();
00035     int    get_from_db(string investorID, string password);
00036     bool   add_to_db(); // for new investors
00037     bool   exists(int iid);
00038
00039     std::map<int, int>  experience; // number from 1 to 5 per asset class
00040     std::map<int, int>  knowledge;
00041     std::map<int, int>  desirability;
00042     std::map<int, float> max_exposure;
00043     void  load_preferences();
00044     bool  save_preferences();
00045     void  set_max_exposure();
00046     bool  set_scale(string scale_type, int ac, int val);
00047
00048
00049
00055 /*this would be used if the experience etc was a vector ... but then we cannot use the simple indices :-()
and have to resort to iterators
00056 * investor() : experience(NBR_ASSET_CLASSES, DEFAULT_SCALE),
00057               knowledge(NBR_ASSET_CLASSES, DEFAULT_SCALE),
00058               desirability(NBR_ASSET_CLASSES, DEFAULT_SCALE) {}
00059 */

```

```

00060 investor()
00061 {
00062     int i;
00063     for (i = 1; i <= NBR_ASSET_CLASSES; i++)
00064     {
00065         experience[i] = DEFAULT_SCALE;
00066         knowledge[i] = DEFAULT_SCALE;
00067         desirability[i] = MAX_SCALE;
00068         max_exposure[i] = 1;
00069     }
00070     experience[1] = MAX_SCALE;
00071     knowledge[1] = MAX_SCALE;
00072 }
00073 };
00074
00075
00082 /*
00083 ostream& operator<<(ostream& os, const investor& the_investor)
00084 {
00085
00086     os << "<h4>" << "the_investor.get_full_name()" << "</h4>"
00087     << "<ul>"
00088     << "<li>currency: " << the_investor.currency << "</li>"
00089     << "<li>birth date: " << tm2DateStr(the_investor.birth_date) << ", age: " <<
00090     "to_string(the_investor.age())" << "</li>"
00091     << "<li>simulate till age: " << to_string(the_investor.simulate_till_age) << "</li>"
00092     << "</ul>";
00093     return os;
00094 */
00095
00096
00102 // usage: const string& s = theInvestor.get_full_name()
00103 string investor::get_full_name() {string s; s =
00104     first_name + " " + last_name; return s; }
00105
00108 inline float investor::age(int Mnbr) {
00109     /* time_t age_date;
00110     time(&age_date); //set the age date to the current time as unix timestamp
00111     //TODO: this is potentially unsafe, if time is not implemented as unix timestamp
00112     age_date += Mnbr * 2629800; // 2629800 = 356.25 / 12 * 24 * 60 * 60
00113     return (difftime(age_date, mktime(&birth_date)) / 2629800);*/
00114     float the_age;
00115     tm tmNow = get_tm();
00116     the_age = tmNow.tm_year - birth_date.tm_year;
00117     the_age = the_age + (tmNow.tm_mon - birth_date.tm_mon) / 12 + Mnbr / 12;
00118     return the_age;
00119 }
00120
00124 inline int investor::months2simulate() {
00125     return (int) round(ceil(simulate_till_age - this->age(0)) * 12);
00126 }
00127
00134 int investor::load_from_db(string uid)
00135 {
00136     string sSQL;
00137     sSQL = "SELECT investor_id, user_name, first_name, last_name, birth_date, currency, simulate_till_age
00138     FROM " + oConfig.tbl_prefix + "_investors WHERE investor_id = " + uid + ";";
00139     try{
00140         sql::Driver *driver;
00141         sql::Connection *con;
00142         sql::Statement *stmt;
00143         sql::ResultSet *res;
00144         // sql::PreparedStatement *pstmt;
00145         // Create a connection
00146         driver = get_driver_instance();
00147         // con = driver->connect("tcp://127.0.0.1:3306","efp","efp.cger");
00148         con = driver->connect(oConfig.db_host, oConfig.db_user,
00149             oConfig.db_pwd);
00150         // con->setSchema("efp");
00151         con->setSchema(db.db);
00152         stmt = con->createStatement();
00153         res = stmt->executeQuery(sSQL);
00154         if (!(res->next()))
00155             {error_message = t_errMsg["user not found"] +
00156                 error_message;}
00157         else
00158         {
00159             // res = stmt->executeQuery(sSQL);
00160             res->previous();
00161             while (res->next())
00162             {
00163                 investor_id = res->getInt("investor_id");
00164                 user_name = res->getString("user_name");
00165                 first_name = res->getString("first_name");

```

```

00165         last_name           = res->getString("last_name");
00166         currency             = res->getString("currency");
00167         simulate_till_age    = atof((res->getString("simulate_till_age")).c_str());
00168         dateStr2tm(res->getString("birth_date"), &birth_date);
00169     }
00170 }
00171 delete res;
00172 delete stmt;
00173 delete con;
00174 } //try
00175
00176 catch (sql::SQLException &e) {
00177     error_message = error_message + t_errMsg["retrievingInvestor"];
00178 #ifdef DEBUG
00179     cout << "# ERR: SQLException in " << __FILE__;
00180     cout << "(" << __FUNCTION__ << ") on line " << __LINE__ << endl;
00181     cout << "# ERR: " << e.what();
00182     cout << " (MySQL error code: " << e.getErrorCode();
00183     cout << ", SQLState: " << e.getSQLState() << ") " << endl;
00184     cout << "[[" << sSQL << "]]*get*";
00185 #endif
00186 }
00187 // return EXIT_SUCCESS;
00188 if (EXIT_SUCCESS == 0)
00189     {return investor_id;}
00190     else
00191     {return 0;}
00192 }
00193
00200 inline int investor::age2monthNbr(float theAge)
00201 {
00202     return (int) round((theAge - this->age(0)) * 12);
00203 }
00204
00209 inline string investor::dateStr2Age(string the_date)
00210 {
00211     // struct tm tm_date;
00212     // dateStr2tm(the_date, &tm_date);
00213     int month_nbr = dateStr2Mnbr(the_date);
00214     return to_string(this->age() + month_nbr / 12);
00215 }
00216
00220 bool investor::save()
00221 {
00222     string s;
00223     s = "UPDATE " + oConfig.tbl_prefix + "_investors SET user_name = '" +
00224     user_name + "',";
00225     s += " first_name = '" + first_name + "',";
00226     s += " last_name = '" + last_name + "',";
00227     s += " simulate_till_age = '" + to_string(simulate_till_age) + "',";
00228     s += " birth_date = '" + tm2DateStr(birth_date) + "',";
00229     s += " currency = '" + currency + "'";
00230     s += " WHERE investor_id = " + to_string(investor_id) + ";";
00231 #ifdef DEBUG
00232     debug_info += "<br>" + s;
00233 #endif
00234     if (db.runSQL(s))
00235     {
00236         investor_id = load_from_db(to_string(investor_id)); // to load other
00237         infor than the investor_id
00238         return true;
00239     }
00240     else
00241     {
00242         error_message = error_message + t_errMsg["errorSavingUser"];
00243         return false;
00244     }
00245 }
00246
00249 int investor::get_from_db(string uid, string password) {
00250     string sSQL;
00251     sSQL = "SELECT investor_id, user_name, first_name, last_name, birth_date, simulate_till_age, currency
00252     FROM " + oConfig.tbl_prefix + "_investors WHERE user_name = '" + uid + "' &&
00253     AES_DECRYPT(password,'" + password + hh.c_salt + "') = '" + password + "'"; try{
00254     sql::Driver *driver;
00255     sql::Connection *con;
00256     sql::Statement *stmt;
00257     sql::ResultSet *res;
00258     // sql::PreparedStatement *pstmt;
00259     // Create a connection
00260     driver = get_driver_instance();
00261     // con = driver->connect("tcp://127.0.0.1:3306","efp","efp.cger");
00262     con = driver->connect(oConfig.db_host,oConfig.db_user,
00263     oConfig.db_pwd);
00264     // con->setSchema("efp");
00265     con->setSchema(db.db);
00266 }

```

```

00264 stmt = con->createStatement();
00265 res = stmt->executeQuery(sSQL);
00266 if (!(res->next()))
00267 {error_message = t_errMsg["pwdUserCombination"] +
error_message;}
00268 else
00269 {
00270 // res = stmt->executeQuery(sSQL);
00271 res->previous();
00272 while (res->next())
00273 {
00274 investor_id = res->getInt("investor_id");
00275 user_name = res->getInt("user_name");
00276 first_name = res->getString("first_name");
00277 last_name = res->getString("last_name");
00278 currency = res->getString("currency");
00279 simulate_till_age = atoi((res->getString("simulate_till_age")).c_str());
00280 dateStr2tm(res->getString("birth_date"), &birth_date);
00281 }
00282 }
00283 delete res;
00284 delete stmt;
00285 delete con;
00286 } //try
00287
00288 catch (sql::SQLException &e) {
00289 error_message = error_message + t_errMsg["retrievingInvestor"];
00290 #ifdef DEBUG
00291 cout << "# ERR: SQLException in " << __FILE__;
00292 cout << "(" << __FUNCTION__ << ")" on line " << __LINE__ << endl;
00293 cout << "# ERR: " << e.what();
00294 cout << " (MySQL error code: " << e.getErrorCode();
00295 cout << ", SQLState: " << e.getSQLState() << ")" << endl;
00296 #endif
00297 }
00298 // return EXIT_SUCCESS;
00299 if (EXIT_SUCCESS == 0)
00300 {
00301 return investor_id;}
00302 else
00303 {return 0;}
00304 }
00305
00313 bool investor::add_to_db()
00314 {
00315 string s;
00316 s = "INSERT INTO " + oConfig.tbl_prefix + "_investors (user_name, first_name, last_name,
password, birth_date, simulate_till_age, currency) ";
00317 s += "value ('" + user_name + "', '" + first_name + "', '" +
last_name + "', " +
00318 s += "AES_ENCRYPT('" + password + "',CONCAT('" + password + "','" +
hh.c_salt + "')),'" +
00319 s += "'" + tm2DateStr(birth_date) + "', " + to_string(
simulate_till_age) + "','" + currency + "'))";
00320 #ifdef DEBUG
00321 debug_info += "<br>" + s;
00322 #endif
00323 if (db.runSQL(s))
00324 {
00325 investor_id = get_from_db(user_name, password); //to set the
investor_id
00326 return true;
00327 }
00328 }
00329 else
00330 {
00331 error_message = error_message + t_errMsg["userNameTaken1"] +
user_name + t_errMsg["userNameTaken2"];
00332 return false;
00333 }
00334 }
00335
00336
00337
00343 bool investor::exists(int iid)
00344 {
00345 string s = "SELECT investor_id FROM " + oConfig.tbl_prefix + "_investors WHERE
investor_id = " + to_string(iid) + " ";
00346 db.res = db.stmt->executeQuery(s);
00347 if ((db.res->next()))
00348 {
00349 return true;
00350 }
00351 else
00352 {
00353 return false;
00354 }

```

```

00355
00356 }
00357
00363 void investor::set_max_exposure()
00364 {
00365     int i;
00366     float denom, sumExp = 0;
00367     denom = (MAX_SCALE + MAX_SCALE) * (MAX_SCALE - 1);
00368     for (i = 1; i <= NBR_ASSET_CLASSES; i++)
00369     {
00370         max_exposure[i] = (experience[i] + knowledge[i]) * (
desirability[i] - 1) / denom;;
00371         sumExp += max_exposure[i];
00372         //error_message = error_message + to_string(max_exposure[i]) + "<BR>";
00373     }
00374     if (max_exposure[1] < 0.6) max_exposure[1] = 0.6; // safest asset at least 60%
00375     if (sumExp <= 1) max_exposure[1] = 1; // too little room --> class
00376     if (sumExp <= 1.5) max_exposure[2] = max((float)0.5, (float)
max_exposure[2]); // if it was smaller than 1 then also the second class is upgraded
00377     if (sumExp <= 2) max_exposure[3] = max((float)0.2, (float)max_exposure[3]); // if it was smaller than 1
then also the
00378 }
00379
00383 void investor::load_preferences()
00384 {
00385     int ac;
00386     string s = "SELECT * FROM " + oConfig.tbl_prefix + "_preferences WHERE investor = " +
to_string(this->investor_id) + ";";
00387     db.res = db.stmt->executeQuery(s);
00388     while (db.res->next())
00389     {
00390         ac = db.getInt("asset_class");
00391         experience[ac] = db.getInt("experience");
00392         knowledge[ac] = db.getInt("knowledge");
00393         desirability[ac] = db.getInt("desirability");
00394     }
00395     set_max_exposure();
00396     // TODO this function will break if one would reduce the number of asset classes
00397 }
00398
00402 bool investor::set_scale(string scale_type, int ac, int val)
00403 {
00404     if (ac < 1 || ac > NBR_ASSET_CLASSES) return false;
00405     if (val < MIN_SCALE) val = MIN_SCALE;
00406     if (val > MAX_SCALE) val = MAX_SCALE;
00407     if (scale_type == "experience") experience[ac] = val;
00408     if (scale_type == "knowledge") knowledge[ac] = val;
00409     if (scale_type == "desirability") desirability[ac] = val;
00410     return true;
00411 }
00412
00416 bool investor::save_preferences()
00417 {
00418     int i;
00419     string s;
00420     bool success = true;
00421     s = "DELETE FROM " + oConfig.tbl_prefix + "_preferences WHERE investor = " + to_string(
investor_id) + ";";
00422     if (db.runSQL(s)) {} else {}//{error_message = error_message + "<li>Please contact us. #errno
E010</li>";}
00423     s = "INSERT INTO " + oConfig.tbl_prefix + "_preferences (investor, asset_class,
experience, knowledge, desirability, max_exposure) VALUES ";
00424     for(i=1; i <= NBR_ASSET_CLASSES; i++)
00425     {
00426         if (i > 1) s = s + ", ";
00427         s = s + "(" + to_string(investor_id) + ", " + to_string(i) + ", " + to_string(
experience[i]) + ", "
00428             + to_string(knowledge[i]) + ", " + to_string(desirability[i]) + ", " +
00429             to_string(max_exposure[i]) + ")";
00430     }
00431     s = s + ";";
00432     //error_message = error_message + "<br>" + s;
00433     if (db.runSQL(s)) {}
00434     else
00435     {
00436         error_message = error_message + "<li>Please contact us. #errno E011</li>";
00437         success = false;
00438     }
00439     //cout << s;
00440     return success;
00441 }

```

8.37 investor_ui.class.cpp File Reference

Classes

- class `investor_ui`

8.38 investor_ui.class.cpp

```

00001
00014 class investor_ui : public investor
00015 {
00016 public:
00017     investor_ui();
00018     // int investor_id;
00019     // string user_name;
00020     // string first_name;
00021     // string last_name;
00022     //
00023     // string currency;
00024     // struct tm birth_date;
00025
00026     // char active_actions[6] = "11000"; //e.g. 11000 for someone who has logged in but did nothing else
00027     int active_action = 1; // eg. 1 for the home screen
00028
00029     bool set_investor_id (const string& s);
00030     bool set_user_name (const string& s);
00031     bool set_first_name (const string& s);
00032     bool set_last_name (const string& s);
00033     bool set_password (const string& s);
00034     bool set_currency (const string& s);
00035     bool set_birth_date (const string& s);
00036     bool set_simulate_till_age(const string& s);
00037     bool is_logged_in ();
00038     void logout ();
00039     bool set_investor_id_fromEnv();
00040     void login ();
00041     void update_last_activity_at();
00042
00043     //inherited: string get_full_name();
00044     //inherited: float age(int Mnbr = 0);
00045     //inherited: int months2simulate();
00046
00047     string parse_user_info(bool parse_wrapper = true);
00048
00049     void home_screen();
00050     void show_home_screen_form();
00051     void register_form();
00052     void register_exec();
00053     void account_form();
00054     void account_save();
00055     void login_form();
00056     void login_exec();
00057     void show_disclaimer();
00058     void personalize_screen();
00059     void personalize_save();
00060     void personalize_reload(string action);
00061
00062     void show_inventory_form();
00063     void asset_add();
00064     void asset_edit();
00065     void asset_save();
00066     void asset_delete();
00067
00068     void goal_help_form();
00069     void goal_help_save();
00070
00071     void goal_list();
00072     void goal_add();
00073     void goal_edit();
00074     void goal_save();
00075     void goal_delete();
00076
00077     void cash_flow_list();
00078     void cash_flow_add();
00079     void cash_flow_edit();
00080     void cash_flow_save();
00081     void cash_flow_delete();
00082
00083     void feedback();
00084     void show_simulation();
00085     void show_follow_up();
00086     void show_total_portfolio(investment_problem * oInvProbl);
00087     void parse_savings_plan(investment_problem * oInvProbl, int g); //
        show the savings plan for goal g

```

```

00088
00089     html_helper hh;
00090
00091
00092 private:
00093     void show_register_form_body();
00094     void show_account_form_body();
00095     void show_goal_help_form_body();
00096     void show_login_form_body();
00097     void show_inventory_form_body();
00098     void parse_bm(std::map<int, float> benchmark, int ref_number);
00099     // personalize:
00100     void personalize_form();
00101     void personalize_pref_table();
00102     void personalize_ER_table();
00103     void personalize_vol_table();
00104     void personalize_ERvol_table();
00105     void personalize_covar_table();
00106     void parse_simulation_duration_form(int nbrMonths, string action = "s");
00107     void parse_simulation_insight(simulation *oSIm);
00108     void parse_form_open_personalize(string the_content);
00109     void parse_form_close_personalize(string the_content, bool show_reload = true
);
00110 );
00111     string button_to(string go_to, string description, char type_redir);
00112     void personalize_set_general();
00113     void personalize_set_prefs();
00114     void personalize_set_ER();
00115     void personalize_set_vol();
00116     void personalize_set_corr();
00117     void parse_goal_box(investment_problem * oInvProbl, int g, string s_var =
"", string s_desc = "");
00118     std::map<int, string> asset_class_names;
00119
00120 };
00121
00122
00126 investor_ui::investor_ui()
00127 {
00128 }
00129
00133 string investor_ui::parse_user_info(bool parse_wrapper)
00134 {
00135     string s;
00136     switch (oConfig.layout)
00137     {
00138     case '2' :
00139         s = "<div class=\"alert alert-info\">";
00140         s = s + "<h4>" + this->get_full_name() + "</h4>"
+ "<ul>"
00141         + "<li>" + t_b_currency + ": " + this->currency + "</li>"
00142         + "<li>" + t_b_date + ": " + tm2DateStr(this->
birth_date) + ", age: " + round2str(this->age(),1) + " years old</li>"
00143         + "<li>" + t_sim_till + ": " + to_string((int)round(this->
simulate_till_age)) + " years old</li>"
00144         + "</ul>"
+ "</div>";
00145     break;
00146     case '1' :
00147     default :
00148         s = "<div class=\"alert alert-info\">";
00149         s = s + "<h4>" + this->get_full_name() + "</h4>"
+ "<ul>"
00150         + "<li>" + t_b_currency + ": " + this->currency + "</li>"
00151         + "<li>" + t_b_date + ": " + tm2DateStr(this->
birth_date) + ", age: " + round2str(this->age(),1) + " years old</li>"
00152         + "<li>" + t_sim_till + ": " + to_string((int)round(this->
simulate_till_age)) + " years old</li>"
00153         + "</ul>"
+ "</div>";
00154     break;
00155     }
00156     return s;
00157 }
00158
00160 inline string investor_ui::button_to(string go_to, string description, char
type_redir)
00161 {
00162     string s;
00163     s = "<button class=\"btn btn-primary\" onclick=\"location.href='";
00164     switch (type_redir)
00165     {
00166     case 'a': s = s + oConfig.soft_url + "?a="; break;
00167     case 's': s = s + oConfig.soft_url + "?a2="; break;
00168     case 'u': {};
00169     default : {};

```

```

00180     }
00181     s = s + go_to + "';\"";
00182     s = s + description;
00183     s = s + "</button>";
00184     return s;
00185 }
00186
00187
00191 void investor_ui::show_home_screen_form()
00192 {
00193     switch (oConfig.layout)
00194     {
00195     case '2':
00196         cout << "<div class=\"row-fluid\">";
00197         cout << "<div class=\"col-md-3 alert alert-info\">\n";
00198             show_register_form_body();
00199         cout << "</div>\n";
00200         cout << "<div class=\"col-md-1\"><center>&nbsp;or&nbsp;</center></div>\n";
00201         cout << "<div class=\"col-md-2 alert alert-info\" >";
00202             show_login_form_body();
00203         cout << "</div>\n";
00204         cout << "</div>\n"; // row
00205         break;
00206     case '1':
00207     default:
00208         //cout << "<div class=\"container-fluid\">";
00209         cout << "<div class=\"row-fluid\">";
00210         cout << "<div class=\"span1\"></div>\n";
00211         cout << "<div class=\"span5 alert alert-info\">\n";
00212             show_register_form_body();
00213         cout << "</div>\n"; // span
00214         cout << "<div class=\"span1\"><p class=\"center\">...OR...</p></div>\n";
00215         cout << "<div class=\"span4 alert alert-info\" >";
00216             show_login_form_body();
00217         cout << "</div>\n"; // span
00218         cout << "</div>\n"; // row
00219         //cout << "</div>\n"; // container
00220     }
00221
00222 /*     cout << p("This simple process consists of three easy steps:");
00223     cout << "<img src=\"" << oConfig.www_dir << "/img/ranking.jpeg\" align=\"right\"/>";
00224     cout << ul();
00225     cout << li("<a href=\"" + oConfig.soft_url + "?a=rf\">Register</a> or <a href=\"" + oConfig.soft_url +
00226         "?a=lf\">Login</a>");
00227     if (investor_id == 0)
00228     {
00229         cout << li("STEP 1: Make an inventory of your assets and main cash-flows (income and expenses)");
00230         cout << li("STEP 2: Set Your personal financial goals");
00231         cout << li("STEP 3: Contemplate on the Feedback");
00232         cout << p(" ");
00233         cout << "<p>Next step: <button type=\"btn btn-primary\" class=\"btn btn-primary\"
00234             onclick=\"window.location.href=\'" + oConfig.soft_url + "?a=rf\'\">Register</button> or ";
00235         cout << "<button type=\"btn btn-primary\" class=\"btn btn-primary\"
00236             onclick=\"window.location.href=\'" + oConfig.soft_url + "?a=lf\'\">Login</button></p>";
00237         cout << p(" ");
00238     }
00239     else
00240     {
00241         cout << li("STEP 1: <a href=\"" + oConfig.soft_url + "?a=if\">Make an inventory of your assets and
00242             main cash-flows (income and expenses)</a>");
00243         cout << li("STEP 2: <a href=\"" + oConfig.soft_url + "?a=gl\">Set Your personal financial goals</
00244             a>");
00245         cout << li("STEP 3: <a href=\"" + oConfig.soft_url + "?a=f\">Contemplate on the Feedback</a>");
00246         cout << p(" ");
00247         cout << "<p>Next step: <button type=\"btn btn-primary\" onclick=\"window.location.href=\'" +
00248             oConfig.soft_url + "?a=gl\'\">Define Your Goals</button></p>";
00249         cout << p(" ");
00250     }
00251     cout << ul();*/
00252 }
00253
00254
00255 void investor_ui::home_screen()
00256 {
00257     hh.header(investor_id);
00258     hh.show_toolBar("home", is_logged_in().get_full_name());
00259     hh.aside(t_aside["welcome"]);
00260     //hh.show_home();
00261     show_home_screen_form();
00262     hh.footer("home", "welcome");
00263 }
00264
00270 void investor_ui::show_register_form_body()
00271 {
00272     cout << "<h2>" << t_Register << "</h2>";
00273     cout << "<n<form class=\"regform\" method=\"post\" action=\"" + oConfig.

```

```

soft_url + ">\n";
00274 cout << "<fieldset>\n";
00275 cout << "<ol>\n";
00276 cout << "<li>\n";
00277 cout << "<label for=\"email\">valid e-mail</label><input type=\"email\" name=\"email\" autofocus=\"true\"
required placeholder=\"boss@universe.org\">\n";
00278 cout << "</li>\n<li>";
00279 cout << "<label for=\"password\">Password</label><input type=\"password\" name=\"password\" required
value=\"\" autocomplete=\"off\" placeholder=\"min 8 characters, captial, special, etc.\" pattern=\"(?=.*\\
d)(?=.*[a-z])(?=.*[A-Z]).{8,}\" title=\"Password must be 8 or more characters with at least one number, an
upper-case letter, a lower-case letter, and one special character!\">\n";
00280 cout << "</li>\n<li>";
00281 cout << "<label for=\"password1\">Repeat Password</label><input type=\"password\" name=\"password1\"
required value=\"\" autocomplete=\"off\" placeholder=\"8-10 characters\" pattern=\"(?=.*\\
d)(?=.*[a-z])(?=.*[A-Z]).{8,}\" title=\"Password must be 8 or more characters with at least one number, an upper-case let
lower-case letter, and one special character!\">\n";
00282
00283 //was: pattern=\"^(?=.*{8,})(?=.*[a-z])(?=.*[A-Z])(?=.*[\\d])(?=.*[\\W]).$\"
00284
00285 cout << "</li>\n<li>";
00286 cout << "<label for=\"first_name\">First Name</label><input type=\"text\" name=\"first_name\"
placeholder=\"FirstName\">\n";
00287 cout << "</li>\n<li>";
00288 cout << "<label for=\"last_name\">Last Name</label><input type=\"text\" name=\"last_name\" placeholder=
\"LastName\">\n";
00289 cout << "</li>\n<li>";
00290 cout << "<label for=\"currency\">Base Currency</label> << oCurrency.
dropDown_currency(\"EUR\");
00291 //<input type=\"text\" name=\"currency\" value=\"EUR\" required placeholder=\"EUR\">\n";
00292 cout << "</li>\n<li>";
00293 cout << "<label for=\"bdate\">birth date</label><input type=\"text\" id=\"datepicker\" data-date-format=
\"yyyy-mm-dd\" data-date-viewmode=\"years\" required placeholder=\"yyyy-mm-dd\" title=\"Enter your birth date
\" name=\"bdate\">\n";
00294 // cout << "<script>$('#datepicker').datepicker('setEndDate', '2012-01-01')</script>";
00295
00296 cout << "<script type=\"text/javascript\">$('#datepicker').datepicker({format: \"yyyy-mm-dd\",
autoclose: true, todayBtn: false, setEndDate: \"2016-01-01\"});</script>";
00297
00298 //cout << "<script type=\"text/javascript\">var nowTemp = new Date(); var now = new
Date(nowTemp.getFullYear(), nowTemp.getMonth(), nowTemp.getDate(), 0, 0, 0, 0); $('#datepicker').datepicker({onRender:
function(date) { return date.valueOf() <= now.valueOf() ? 'disabled' : '';}).on('changeDate', function(ev) {
checkout.hide();}).data('datepicker');</script>";
00299
00300
00301 cout << "</li>\n<li>";
00302 cout << "<label for=\"simtill\">simulate till age</label><input type=\"number\" required placeholder=\"90
\" title=\"Fill what age is this simulation relevant\" name=\"simtill\">\n";
00303 cout << "</li>\n<li>";
00304 // cout << "<label for=\"user_type\">e-mail</label><input type=\"radio\" name=\"user_type\"
value=\"investor\" checked=\"checked\" />&nbsp;investor <input type=\"radio\" name=\"user_type\" value=\"advisor\"
/>&nbsp;advisor\n";
00305 cout << "</li>\n<li>";
00306 cout << "<input type=\"checkbox\" name=\"agree\" value=\"ragree\" required/>I hereby confirm that I agree
with the << hh.terms_of_use() << ", the << hh.privacy_policy() << " and the
" << hh.cookie_policy() << ".";
00307 cout << "</li>\n<li>";
00308 cout << "<button class=\"btn btn-primary btn-block\" type=\"submit\" value=\"Submit\"> << t_Register <<
\"</button>\n&nbsp;";
00309 // cout << "<button class=\"btn btn-primary\" type=\"reset\" value=\"reset\">Reset</button>\n&nbsp;";
00310 cout << "</li></ol>\n";
00311 cout << "</fieldset>";
00312 cout << "<input type=\"hidden\" name=\"a\" value=\"re\" />&nbsp;\n";
00313 cout << "</form>\n";
00314 // cout << t_regFrm["requiredFields"];
00315 cout << t_regFrm["policiesNote"];
00316 }
00317
00318 /* *****
00319 * register_form
00320 *
00321 * *****/
00322 void investor_ui::register_form()
00323 {
00324 hh.header(investor_id);
00325 hh.show_toolBar("register");
00326 hh.aside(t_aside["register"]);
00327 show_register_form_body();
00328 hh.footer("register", "register");
00329 }
00330
00331
00332 void investor_ui::show_account_form_body()
00333 {
00334 //cout << "<script>$('#sandbox-container input').datepicker({format: \"yyyy-mm-dd\"});</script>";
00335 cout << "<br>\n";
00336 cout << "<div class=\"well\">";
00337 cout << "\n<form class=\"\" << ((oConfig.layout == '2')?"regform":"") << "\" method=\"post\"
action=\"\" << oConfig.soft_url << ">\n";

```

```

00340 cout << "<fieldset>\n";
00341 cout << "<ol>\n";
00342 cout << "<li>\n";
00343 cout << "<label for=\"email\">e-mail</label><input type=\"email\" name=\"email\" autofocus=\"true\"
required placeholder=\"boss@universe.org\" value=\"\" << user_name << "\">\n";
00344 cout << "</li>\n<li>";
00345 cout << "<label for=\"password_old\">Old Password</label><input type=\"password\" name=\"password_old\"
value=\"\" autocomplete=\"off\" placeholder=\"8-10 characters\" pattern=\"
^(?=.*{8,})(?=.*[a-z])(?=.*[A-Z])(?=.*[\\d])(?=.*[\\W]).*$\" title=\"Password must be 8 or more characters with at least one
upper-case letter, a lower-case letter, and one special character!\">\n";
00346 cout << "</li>\n<li>";
00347 cout << "<label for=\"password\">New Password</label><input type=\"password\" name=\"password\" value=\"\"
\" autocomplete=\"off\" placeholder=\"8-10 characters\" pattern=\"^(?=.*{8,})(?=.*[a-z])(?=.*[A-Z])(?=.*[\\d])(?=.*[\\W]).*$\" title=\"Password must be 8 or more characters with at least one
number, an upper-case
letter, a lower-case letter, and one special character!\">\n";
00348 cout << "</li>\n<li>";
00349 cout << "<label for=\"password1\">Repeat New Password</label><input type=\"password\" name=\"password1\"
value=\"\" autocomplete=\"off\" placeholder=\"8-10 characters\" pattern=\"
^(?=.*{8,})(?=.*[a-z])(?=.*[A-Z])(?=.*[\\d])(?=.*[\\W]).*$\" title=\"Password must be 8 or more characters with at least one
upper-case letter, a lower-case letter, and one special character!\">\n";
00350 cout << "</li>\n<li>";
00351 cout << "<label for=\"first_name\">First Name</label><input type=\"text\" name=\"first_name\" required
placeholder=\"FirstName\" value=\"\" << first_name << "\">\n";
00352 cout << "</li>\n<li>";
00353 cout << "<label for=\"last_name\">Last Name</label><input type=\"text\" name=\"last_name\" required
placeholder=\"LastName\" value=\"\" << last_name << "\">\n";
00354 cout << "</li>\n<li>";
00355 cout << "<label for=\"currency\">Currency</label> << oCurrency.
dropDown_currency(currency) << "\n";
00356 cout << "</li>\n<li>";
00357 cout << "<label for=\"bdate\">birth date</label><input class=\"datepicker\" data-date-format=\"yyyy-mm-dd
\" data-date-viewmode=\"years\" required placeholder=\"yyyy-mm-dd\" title=\"Enter your birth date\" name=\"
bdate\" value=\"\" << tm2DateStr(birth_date) << "\">\n";
00358 cout << "<script>$(\".datepicker\").datepicker();</script>";
00359 cout << "</li>\n<li>";
00360
00361 // cout << "<label for=\"user_type\">e-mail</label><input type=\"radio\" name=\"user_type\"
value=\"investor\" checked=\"checked\" />&nbsp;investor <input type=\"radio\" name=\"user_type\" value=\"advisor\"
/>&nbsp;advisor *n";
00362 cout << "</li>\n<li>";
00363 // cout << "<input type=\"checkbox\" name=\"agree\" value=\"ragree\" required/>I hereby confirm that I
agree with Privacy and Cookie Policy.";
00364 // cout << "</li>\n<li>";
00365 cout << "<label for=\"simtill\">simulate till age</label><input type=\"number\" required placeholder=\"
110\" title=\"Till what age is this simulation relevant\" name=\"simtill\" value=\"\" <<
simulate_till_age << "\">\n";
00366 cout << "</li>\n<li>";
00367
00368 cout << "<button class=\"btn btn-primary\" type=\"submit\" value=\"Submit\">Save</button>\n&nbsp;";
00369 cout << "<button class=\"btn btn-primary\" type=\"reset\" value=\"reset\">Reset</button>\n&nbsp;";
00370 cout << "</li></ol>\n";
00371 cout << "</fieldset>";
00372 cout << "<input type=\"hidden\" name=\"a\" value=\"is\" />&nbsp;\n";
00373 cout << "</form>\n";
00374 cout << t_regFrm["requiredFields"];
00375 cout << t_regFrm["policiesNote"];
00376 cout << "<p>[*] only to be filled out IF you want to change the password, in that case all three fields
have to be filled out.</p>";
00377 cout << "</div>"; // well
00378 }
00379
00383 void investor_ui::account_form()
00384 {
00385 hh.header(investor_id);
00386 hh.show_toolBar("", get_full_name());
00387 // hh.aside(t_aside["register"]);
00388 show_account_form_body();
00389 hh.footer("", "account");
00390 }
00391
00395 void investor_ui::show_disclaimer()
00396 {
00397 hh.header(investor_id);
00398 hh.show_toolBar("", get_full_name());
00399 hh.aside(t_aside["disclaimer"]);
00400 cout << t_disclaimer;
00401 hh.footer("", "");
00402 }
00403
00409 void investor_ui::register_exec()
00410 {
00411 string uid, pwd, pwd1, fName, lName, currency, bdate, curr, simtill;
00412 bool success = true;
00413
00414 uid = cgi("email");
00415 pwd = cgi("password");
00416 pwd1 = cgi("password1");

```

```

00417 fName    = cgi("first_name");
00418 lName    = cgi("last_name");
00419 curr     = cgi("currency");
00420 bdate    = cgi("bdate");
00421 simtill  = cgi("simtill");
00422 #ifdef DEBUG
00423 debug_info += "<br>bdate: " + bdate;
00424 #endif
00425 // form_iterator fvalue1 = cgi.getElement(varName);
00426
00427 if (pwd == pwd1)
00428 // if (strcmp(pwd,pwd1) == 0)
00429 {
00430     if (set_user_name(uid) && set_first_name(fName) &&
set_last_name(lName) && set_password(pwd) &&
set_birth_date(bdate) && set_simulate_till_age(simtill) &&
set_currency(curr) && add_to_db())
00431     {
00432         investor_id = get_from_db(uid, pwd); //changes the error_message
00433 //         if (investor_id != 0) {cout << "Set-Cookie:iid=" << std::to_string(investor_id) << "\n";}
00434     }
00435     else
00436     {
00437         success = false;
00438     }
00439 }
00440 else
00441 {
00442     error_message = error_message + t_errMsg["pwdsMatch"];
00443     error_message = error_message + "<li>pwd: " + pwd + ", pwd1: " + pwd1 + "
</li>";
00444     success = false;
00445 }
00446 if (success)
00447 {
00448     hh.header(investor_id);
00449     hh.show_toolBar("goals",get_full_name());
00450 //xxx     oGoal.list(investor_id, true); // list existing and show input form for new
00451     show_goal_help_form_body();
00452     hh.aside(t_aside["goalInfo"]);
00453 //xxx     hh.footer("assets","goal");
00454     hh.footer("", "goal_help");
00455 }
00456 else
00457 {
00458     hh.header(0);
00459     hh.show_toolBar("register");
00460     show_register_form_body();
00461     hh.aside(t_aside["registerFail"]);
00462     hh.footer("register","register");
00463 }
00464 }
00465
00470 void investor_ui::account_save()
00471 {
00472     string uid, pwdold, pwd, pwd1, fName, lName, currency, bdate, curr, simtill;
00473     bool success = true;
00474
00475     uid      = cgi("email");
00476     pwdold   = cgi("password_old");
00477     pwd      = cgi("password");
00478     pwd1     = cgi("password1");
00479     fName    = cgi("first_name");
00480     lName    = cgi("last_name");
00481     curr     = cgi("currency");
00482     bdate    = cgi("bdate");
00483     simtill  = cgi("simtill");
00484 #ifdef DEBUG
00485     debug_info += "<br>bdate: " + bdate;
00486 #endif
00487 // form_iterator fvalue1 = cgi.getElement(varName);
00488
00489     if (set_user_name(uid) && set_first_name(fName) &&
set_last_name(lName) && set_birth_date(bdate) &&
set_currency(curr) && set_simulate_till_age(simtill) &&
save())
00490     {
00491         investor_id = get_from_db(uid, pwd); //changes the error_message
00492 //         if (investor_id != 0) {cout << "Set-Cookie:iid=" << std::to_string(investor_id) << "\n";}
00493     }
00494     else
00495     {
00496         success = false;
00497     }
00498
00499     if (pwdold != "" && pwd == pwd1)
00500     {

```

```

00501     cout << "<br>PASSWORDCHANGE not implemented yet, sorry!"; //TODO
00503     error_message = error_message + t_errMsg["pwdsMatch"];
00504     error_message = error_message + "<li>pwd: " + pwd + ", pwd1: " + pwd1 + "
</li>";
00505     success = false;
00506 }
00507 if (success)
00508 {
00509     hh.header(investor_id);
00510     hh.show_toolBar("goals",get_full_name());
00511     oGoal.list(investor_id, true); // list existing and show input form for new
00512     hh.aside(t_aside["welcomeBack1"] + get_full_name() +
t_aside["welcomeBack1"]);
00513     hh.footer("goals", "asset");
00514 }
00515 else
00516 {
00517     //show the account form again ...
00518     account_form();
00519 }
00520 }
00521
00522
00523 /* *****
00524 * show_login_form_body
00525 * *****
00526 */
00527 void investor_ui::show_login_form_body()
00528 {
00529     cout << "<h2>" << t_Login << "</h2>";
00530     cout << "<form class=\"loginform\" method=\"post\" action=\"" + oConfig.
soft_url + "\">";
00531     cout << "<fieldset id=\"login_form\">\n";
00532     cout << "<ol>\n";
00533     cout << "<li>\n";
00534     cout << "<label for=\"email\">e-mail</label><input type=\"email\" name=\"email\" autofocus=\"true\"
required placeholder=\"boss@universe.org\">\n";
00535     cout << "</li>\n<li>";
00536     cout << "<label for=\"password\">Password</label><input type=\"password\" name=\"password\" >\n";
00537     cout << "</li>\n<li>";
00538     cout << "<button class=\"btn btn-primary btn-block\" type=\"submit\" value=\"Submit\">" << t_Login << "
</button>\n\n";
00539     // cout << "<button class=\"btn btn-primary\" type=\"reset\" value=\"Reset\">Clear Form</button>\n\n";
00540     cout << "<input type=\"hidden\" name=\"a\" value=\"le\" />\n";
00541     cout << "</li></ol>\n";
00542     cout << "</fieldset>";
00543     cout << "</form>\n";
00544     // cout << "<p>(*) Required fields.</p>";
00545 }
00546
00547
00548 /* *****
00549 * login_form
00550 *
00551 * *****
00552 void investor_ui::login_form()
00553 {
00554     hh.header(0);
00555     hh.show_toolBar("login");
00556     hh.aside(t_aside["login"]);
00557     show_login_form_body();
00558     hh.footer("login", "login");
00559 }
00560
00561
00562 /* *****
00563 * login_exec
00564 * *****
00565 */
00566 void investor_ui::login_exec()
00567 {
00568     string uid, pwd;
00569     error_message = ""; // clear the error message of this object
00570     uid = cgi("email");
00571     pwd = cgi("password");
00572
00573     investor_id = get_from_db(uid, pwd); //changes the error_message
00574     if (investor_id != 0)
00575     {
00576         hh.header(investor_id);
00577         login();
00578         hh.show_toolBar("personalize",get_full_name());
00579         personalize_form();
00580         hh.aside(t_aside["welcomeBack1"] + get_full_name() +
t_aside["welcomeBack2"]);
00581         hh.footer("personalize", "personalize");
00582     }

```

```

00583     else
00584     {
00585         hh.header(0);
00586         hh.show_toolBar("login");
00587         show_login_form_body();
00588         hh.footer("login", "login");
00589     }
00590 }
00591
00595 void investor_ui::personalize_pref_table()
00596 {
00597     int i;
00598     this->load_preferences(); // loads the experience, knowledge and desirability (includes
                                set_max_exposure)
00599
00600     //cout << "<table width=\"100%\" class=\"prefTable\">";
00601     cout << "<tr><th>" << t_Asset_Class << "</th>"
00602           << "<th>" << t_Experience << "</th>"
00603           << "<th>" << t_Knowledge << "</th>"
00604           << "<th>" << t_Desirability << "</th>"
00605           << "<th>" << t_Max_Exp << "</th></tr>";
00606     for(i=1; i <= NBR_ASSET_CLASSES; i++)
00607     {
00608         cout << "<tr>";
00609         cout << "<td>" << asset_class_names[i] << "</td>";
00610         cout << "<td><input type=\"number\" min=\"1\" max=\"5\" name=\"experience\" << to_string(i) << "\"
                                required size=\"1\" value=\"\" << experience[i] << "\"></td>";
00611         cout << "<td><input type=\"number\" min=\"1\" max=\"5\" name=\"knowledge\" << to_string(i) << "\"
                                required size=\"1\" value=\"\" << knowledge[i] << "\"></td>";
00612         cout << "<td><input type=\"number\" min=\"1\" max=\"5\" name=\"desirability\" << to_string(i) << "\"
                                required size=\"1\" value=\"\" << desirability[i] << "\"></td>";
00613         cout << "<td>" << max_exposure[i] << "</td>"; //the limit in %
00614         cout << "</tr>";
00615     }
00616     // done by the close-function: cout << "</table>";
00617 }
00618
00619
00623 void investor_ui::personalize_ER_table()
00624 {
00625     int ac;
00626     // cout << "<table width=\"100%\" class=\"prefTable\">";
00627     cout << "<tr><th> Asset Class </th><th>" << t_personalize["ER"]["thead"] << "
                                </th><th></th></tr>";
00628     for(ac=1; ac <= NBR_ASSET_CLASSES; ac++)
00629     {
00630         cout << "<tr>";
00631         cout << "<td>" << asset_class_names[ac] << "</td>";
00632         cout << "<td><input step=\"0.1\" type=\"number\" min=\"-10\" max=\"25\" name=\"ER\" << to_string(ac) <<
                                "\" required size=\"1\" value=\"\" << round2str(ER[ac] * 100,1) << "\"></td><td>%</td>";
00633         cout << "</tr>";
00634     }
00635 }
00636
00640 void investor_ui::personalize_vol_table()
00641 {
00642     int ac;
00643     // cout << "<table width=\"100%\" class=\"prefTable\">";
00644     cout << "<tr><th>" << t_Asset_Class << "</th><th>" <<
                                t_personalize["vol"]["thead"] << "</th><th></th></tr>";
00645     for(ac=1; ac <= NBR_ASSET_CLASSES; ac++)
00646     {
00647         cout << "<tr>";
00648         cout << "<td>" << asset_class_names[ac] << "</td>";
00649         cout << "<td><input step=\"0.1\" type=\"number\" min=\"0\" max=\"100\" name=\"vol\" << to_string(ac) <<
                                "\" required size=\"1\" value=\"\" << round2str(sqrt(covar[ac][ac]) * 100,1) << "\"
                                ></td><td>%</td>";
00650         cout << "</tr>";
00651     }
00652 }
00653
00657 void investor_ui::personalize_ERvol_table()
00658 {
00659     int ac;
00660     cout << "<tr><th>" << t_Asset_Class << "</th>";
00661     cout << "<th>" << t_personalize["ER"]["thead"] << "</th>";
00662     cout << "<th>" << t_personalize["vol"]["thead"] << "</th>";
00663     cout << "</tr>";
00664     for(ac=1; ac <= NBR_ASSET_CLASSES; ac++)
00665     {
00666         cout << "<tr>";
00667         cout << "<td>" << asset_class_names[ac] << "</td>";
00668         cout << "<td><input step=\"0.1\" type=\"number\" min=\"-10\" max=\"25\" name=\"ER\" << to_string(ac) <<
                                "\" required size=\"1\" value=\"\" << round2str(ER[ac] * 100,1) << "\">%</td>";
00669         cout << "<td><input step=\"0.1\" type=\"number\" min=\"0\" max=\"100\" name=\"vol\" << to_string(ac) <<
                                "\" required size=\"1\" value=\"\" << round2str(sqrt(covar[ac][ac]) * 100,1) << "\">%</td>";
00670         cout << "</tr>";

```

```

00671     }
00672 }
00673
00674
00675 void investor_ui::personalize_covar_table()
00676 {
00677     int ac_i, ac_j;
00678     float covar_ij;
00679     // cout << "<table width=\"100%\" class=\"covar\">";
00680     cout << "<tr><th></th></tr>";
00681     for (ac_i=1; ac_i <= NBR_ASSET_CLASSES; ac_i++) cout << "<th>" <<
asset_class_names[ac_i] << "</th>";
00682     cout << "</tr>";
00683     for (ac_i=1; ac_i <= NBR_ASSET_CLASSES; ac_i++)
00684     {
00685         cout << "<tr>";
00686         cout << "<th>" << asset_class_names[ac_i] << "</th>";
00687         for (ac_j=1; ac_j <= NBR_ASSET_CLASSES; ac_j++)
00688         {
00689             //cout << "<covar[" << ac_i << ", " << ac_j << "] = " << covar[ac_i][ac_j];
00690             if (covar[ac_i][ac_i] != 0 && covar[ac_j][ac_j] != 0)
00691             {
00692                 covar_ij = covar[ac_i][ac_j] / (sqrt(covar[ac_i][ac_i] * covar[ac_j][ac_j]));
00693             }
00694             else
00695             {
00696                 covar_ij = 0;
00697                 error_message = error_message + "<li>ERRNO200: incorrect covar matrix</li>";
00698             }
00699             cout << "<td>" << "<input " << ((ac_i <= ac_j)?"disabled:":"") << " step=\"1\" type=\"number\" min=\"
-100\" max=\"100\" name=\"corr\" << add0(ac_i)<< add0(ac_j) << "\" required size=\"1\" value=\" <<
round2str(covar_ij * 100.0, 0) << "\"></td>";
00700         }
00701     }
00702     cout << "</tr>";
00703 }
00704 }
00705 }
00706 }
00707 }
00708 }
00709 void investor_ui::personalize_form()
00710 {
00711     asset_class_names = oAssetClass.get_assetList(); //
asset_class_names is a variable of the investor class
00712     init_personalize();
00713     cout << t_personalize1;
00714
00715     cout << "<div class=\"panel-group\" id=\"accordion2\">";
00716
00717     parse_form_open_personalize("general");
00718     cout << "<tr>\n";
00719
00720     cout << "<td><label for=\"currency\">Currency*</label></td><td>" << oCurrency.
dropDown_currency(currency) << "</td><td>" <<
t_info_curr << "</td>\n</tr>";
00721     cout << "<tr><td><label for=\"bdate\">birth date*</label></td><td><input class=\"datepicker\"
data-date-format=\"yyyy-mm-dd\" data-date-viewmode=\"years\" required placeholder=\"yyyy-mm-dd\" title=\"Enter your
birth date\" name=\"bdate\" value=\" << tm2DateStr(birth_date) << "\"></td><td>" <<
t_info_bdate << ".</td>\n</tr>";
00722
00723     cout << "<script>$(\".datepicker\").datepicker()</script>";
00724
00725     cout << "<tr><td><label for=\"simtill\">simulate till age*</label></td><td><input type=\"number\"
required placeholder=\"110\" title=\"Till what age is this simulation relevant\" name=\"simtill\" value=\" <<
simulate_till_age << "\" ></td><td>" << t_info_simtill << "</td></tr>";
00726     parse_form_close_personalize("general", false);
00727     /*
00728     *moved to extra info:
00729     cout << t_personalize_concepts;
00730     cout << t_personalize_scoring;
00731     */
00732     parse_form_open_personalize("prefs");
00733     personalize_pref_table();
00734     parse_form_close_personalize("prefs", false);
00735
00736     // cout << t_regFrm["requiredFields"];
00737     this->load_ER(this->investor_id); // loads the this->ER map
00738     this->load_covar(this->investor_id); // loads the this->covar map of maps
00739
00740     if (oConfig.merge_ER_vol)
00741     {
00742         parse_form_open_personalize("ERvol");
00743         personalize_ERvol_table();
00744         parse_form_close_personalize("ERvol");
00745     }
00746     else
00747     {
00748

```

```

00753     parse_form_open_personalize("ER");
00754     personalize_ER_table();
00755     parse_form_close_personalize("ER");
00756
00757
00758     parse_form_open_personalize("vol");
00759     personalize_vol_table();
00760     parse_form_close_personalize("vol");
00761 }
00762
00763     parse_form_open_personalize("corr");
00764     personalize_covar_table();
00765     parse_form_close_personalize("corr");
00766
00767     cout << "</div>"; // the accordion2
00768     cout << "<br clear=\"all\">";
00769 }
00770
00774 void investor_ui::parse_form_open_personalize(string the_content)
00775 {
00776     cout << "\n<div class=\"panel panel-default\">";
00777     cout << "\n<div class=\"panel-heading\">";
00778     cout << " <a data-toggle=\"collapse\" data-parent=\"#accordion2\" data-target=\"#collapse\" <<
the_content << "\n>";
00779     cout << "<h3>" << t_personalize[the_content]["title"] << " &nbsp;<span class=\"glyphicon
glyphicon-triangle-bottom\"></span></h3></a>";
00780     cout << "\n</div>";
00781     cout << "\n<div id=\"collapse\" << the_content << "\" class=\"panel-collapse collapse\">";
00782     cout << "\n<div class=\"panel-body\">";
00783     switch (oConfig.layout)
00784     {
00785         case '1' :
00786             cout << t_personalize[the_content]["info"];
00787             break;
00788         case '2' :
00789             // cout << t_personalize[the_content]["info"];
00790             bs_modal(t_show_extra_info, t_personalize[the_content]["info"], "modal" +
the_content);
00791             break;
00792         default:
00793             cout << t_personalize[the_content]["info"];
00794     }
00795     cout << "\n<form method=\"post\" action=\"\" + oConfig.soft_url + "\">\n";
00796     cout << "<fieldset>\n";
00797     cout << "\n<table class=\"\" << the_content << "\n>";
00798 }
00799
00800
00804 void investor_ui::parse_form_close_personalize(string the_content,
bool show_reload)
00805 {
00806     cout << "</table>";
00807     cout << "<button class=\"btn btn-primary\" type=\"submit\" name=\"a\" value=\"ps\">Save " <<
t_personalize[the_content]["content"] << "</button>\n&nbsp;";
00808     cout << "<button class=\"btn btn-primary\" type=\"reset\" value=\"reset\">Undo Last Changes</button>\n
&nbsp;";
00809     if (show_reload) cout << "<button class=\"btn btn-primary\" type=\"submit\" name=\"a\" value=\"\" << \"r\"
<< the_content << "\n\" << t_reload << " " << t_personalize[the_content]["content"] << "</button>\n
&nbsp;";
00810     //cout << button_to("r" + the_content, t_reload + " " + t_personalize[the_content]["content"], 's');
00811     cout << "</fieldset>";
00812     // cout << "<input type=\"hidden\" name=\"a\" value=\"ps\" />&nbsp;\n";
00813     cout << "<input type=\"hidden\" name=\"content\" value=\"\" << the_content << "\n\" />&nbsp;\n";
00814     cout << "<input type=\"hidden\" name=\"uid\" value=\"\" << investor_id << "\n\" />&nbsp;\n";
00815     cout << "</form>\n";
00816     cout << "<p><a data-toggle=\"collapse\" data-parent=\"#accordion2\" data-target=\"#collapse\" +
the_content + "\n\">";
00817     cout << " t_collapse << " &nbsp;<span class=\"glyphicon glyphicon-triangle-top\"
></span></a></p>";
00818     cout << "</div></div></div>" << endl;
00819 }
00820
00821
00822
00826 void investor_ui::personalize_screen()
00827 {
00828     hh.header(investor_id);
00829     hh.show_toolBar("personalize", get_full_name());
00830     hh.aside(t_aside["personalize"]);
00831     personalize_form();
00832     hh.footer("personalize", "personalize");
00833 }
00834
00838 void investor_ui::personalize_save()
00839 {
00840
00841
00842     string the_content = cgi("content"); // the content that is being updated

```

```

00843     //hh.header(investor_id);
00844     if (the_content == "general")    {personalize_set_general();
save_preferences();}
00845     else if (the_content == "prefs") {personalize_set_prefs();
save_preferences();}
00846     else if (the_content == "ER")    {personalize_set_ER();
save_ER(this->investor_id);}
00847     else if (the_content == "vol")   {personalize_set_vol();
save_covar(this->investor_id);}
00848     else if (the_content == "ERvol") // in case ER and vol were presented in one table
00849     {
00850         personalize_set_ER();
00851         personalize_set_vol();
00852         save_ER(this->investor_id);
00853         save_covar(this->investor_id);
00854     }
00855     else if (the_content == "corr") {personalize_set_corr();
save_covar(this->investor_id);}
00856
00857     set_max_exposure(); // to have them ready
00858     // if relevant show an error message and show the screen again
00859     personalize_screen();
00860 }
00861
00865 void investor_ui::personalize_set_general()
00866 {
00867     string s, uid, currency, bdate, curr, simtill, ac;
00868
00869     // process the general information
00870     uid     = cgi("uid");
00871     curr    = cgi("currency");
00872     bdate   = cgi("bdate");
00873     simtill = cgi("simtill");
00874
00875     if (set_birth_date(bdate) && set_currency(curr) &&
set_simulate_till_age(simtill) && save())
00876     {
00877         investor_id = load_from_db(uid);
00878     }
00879     else
00880     {
00881         error_message = error_message + "<li>I failed to save essential user
information, please contact an administrator -- #ErrNo009</li>";
00882     }
00883 }
00884
00885
00889 void investor_ui::personalize_set_prefs()
00890 {
00891     string s, ac;
00892     int i;
00893
00894     // process the asset allocation related information
00895     for(i=1; i <= NBR_ASSET_CLASSES; i++)
00896     {
00897         ac = to_string(i);
00898         s = cgi("experience" + ac);
00899         set_scale("experience", i, atoi(s.c_str()));
00900         s = cgi("knowledge" + ac);
00901         set_scale("knowledge", i, atoi(s.c_str()));
00902         s = cgi("desirability" + ac);
00903         set_scale("desirability",i, atoi(s.c_str()));
00904     }
00905 }
00906
00910 void investor_ui::personalize_set_ER()
00911 {
00912     int ac;
00913     for (ac=1; ac <= NBR_ASSET_CLASSES; ac++)
00914     {
00915         ER[ac] = atof((cgi("ER" + to_string(ac))).c_str()) / 100;
00916     }
00917 }
00918
00922 void investor_ui::personalize_set_vol()
00923 {
00924     load_covar(investor_id); // load all the values and overwrite in the next lines a
few
00925     int ac;
00926     for (ac=1; ac <= NBR_ASSET_CLASSES; ac++)
00927     {
00928         covar[ac][ac] = pow(atof((cgi("vol" + to_string(ac))).c_str()) / 100, 2);
00929     }
00930 }
00931
00935 void investor_ui::personalize_set_corr()
00936 {

```

```

00937 load_covar(investor_id); // needed because we need the volatilities
00938 int ac_i, ac_j;
00939 for (ac_i=1; ac_i <= NBR_ASSET_CLASSES; ac_i++)
00940 {
00941     for (ac_j=1; ac_j <= NBR_ASSET_CLASSES; ac_j++)
00942     {
00943         if (ac_i > ac_j)
00944         {
00945             covar[ac_i][ac_j] = covar[ac_j][ac_i] = atof((cgi("corr" + add0(ac_i) +
add0(ac_j))).c_str()) / 100 * sqrt(covar[ac_i][ac_i] * covar[ac_j][ac_j]);
00946             //cout << "<br>CGI: " << cgi("corr" + add0(ac_i) + add0(ac_j)) << " | covar[" << ac_i << "]"[" << ac_j
<< "] = " << covar[ac_i][ac_j] << " | covar_ii " << covar[ac_i][ac_i];
00947         }
00948     }
00949 }
00950 }
00951
00957 void investor_ui::personalize_reload(string action)
00958 {
00959     if (action == "rER") reload_ER(investor_id);
00960     else if (action == "rvol") reload_covar(investor_id, 'v');
00961     else if (action == "rERvol")
00962     {
00963         reload_ER(investor_id);
00964         reload_covar(investor_id, 'v');
00965     }
00966     else if (action == "rcorr") reload_covar(investor_id, 'c');
00967     else
00968     {
00969         error_message = error_message + "<li>unexpected action passed to reload --
Error Number ER656</li>";
00970     }
00971     personalize_screen();
00972     // error_message = error_message + "<LI>action=" + action + "</li>";
00973 }
00974
00975
00976
00977 /* *****
00978 * show_inventory_form
00979 *
00980 * *****/
00981 void investor_ui::show_inventory_form()
00982 {
00983     hh.header(investor_id);
00984     hh.show_toolBar("assets",get_full_name());
00985     show_inventory_form_body();
00986     hh.footer("assets","asset");
00987 }
00988
00989 /* *****
00990 * show_inventory_form_body
00991 *
00992 * *****/
00993 void investor_ui::show_inventory_form_body()
00994 {
00995     // cout << h2("Assets / Current Possessions");
00996     cout << "<form method=\"post\" action=\"" + oConfig.soft_url + "?a=aa\">";
00997     cout << "<table class=\"asset\">\n";
00998     cout << "<tr>";
00999     cout << "<th>Description</th><th>Asset Class</th><th>Amount</th><th>Currency</th><th>Actions</th>";
01000     cout << "</tr>\n";
01001     oAsset.list_assets(investor_id);
01002     cout << "<tr>";
01003     cout << "<td><input type=\"text\" name=\"description\" required ></td>";
01004     cout << "<td>" << oAssetClass.dropDown_assetClass() << "</td>";
01005     cout << "<td><input type=\"number\" min=\"0\" name=\"amount\" required placeholder=\"1000\" title=\"
Please enter a positive number\"></td>";
01006     cout << "<td>" << oCurrency.dropDown_currency() << "</td>";
01007
01008     switch (oConfig.layout)
01009     {
01010         case '1' :
01011             cout << "<td><button type=\"submit\" value=\"Submit\"><img src=\"" <<
oConfig.css_dir << "plus.svg\" alt=\"add\" width=\"16\"></button></td>";
01012             break;
01013         case '2' :
01014             cout << "<td><button type=\"submit\" value=\"Submit\" data-toggle=\"tooltip\" title=\"" <<
t_save << "\"><span class=\"glyphicon glyphicon-ok\" style=\"color:green\"></span></button>";
01015             cout << "<button type=\"submit\" value=\"Submit\" data-toggle=\"tooltip\" title=\"" <<
t_reset << "\"><span class=\"glyphicon glyphicon-chevron-left\" style=\"color:blue\"
></span></button>";
01016             break;
01017         default :
01018             cout << "<td><button type=\"reset\" data-toggle=\"tooltip\" title=\"" << t_reset << "\"><span
class=\"glyphicon glyphicon-chevron-left\" style=\"color:blue\"></span></button></td>";
01019             break;

```

```

01020     }
01021
01022     cout << "</table>\n";
01023     cout << "<input type=\"hidden\" name=\"iid\" value=\"" << investor_id << "\" />\n&nbsp;";
01024     cout << "</form>";
01025
01026 //   cout << h2("Future Savings (Cash Flows)"); //   cout << p("... to be added (not implemented yet).");
01027
01028 }
01029
01030
01031
01032 /* *****
01033 * asset_add
01034 * *****
01035 */
01036 void investor_ui::asset_add()
01037 //cgicc::Cgicc &formData
01038 {
01039     hh.header(investor_id);
01040 //   string iid       = cgi("iid");
01041     string desc      = cgi("description");
01042     string aclass    = cgi("asset_class");
01043     string amnt      = cgi("amount");
01044     string curr      = cgi("currency");
01045
01046     error_message    = "";
01047
01048     if (oAsset.set_investor(to_string(investor_id)) &&
oAsset.set_description(desc) && oAsset.
set_asset_class(aclass) && oAsset.set_amount(amnt) &&
oAsset.set_currency(curr) && oAsset.save())
01049     {
01050         cout << "<p class=\"markup\">New asset, \"" << desc << "\", successfully added.</p>";
01051     }
01052     hh.show_toolBar("assets",get_full_name());
01053     show_inventory_form_body();
01054     hh.footer("assets","asset");
01055 }
01056
01061 void investor_ui::asset_edit()
01062 {
01063     string iid = cgi("iid");
01064     string aid = cgi("aid");
01065     error_message = "";
01066
01067     set_investor_id(iid);
01068     hh.header(investor_id);
01069     hh.show_toolBar("assets",get_full_name());
01070
01071     if (oAsset.set_asset_id(aid))
01072     {
01073         oAsset.load();
01074         oAsset.show_edit_form(investor_id);
01075     }
01076
01077     hh.footer("");
01078 }
01079
01080 /* *****
01081 * asete_Save
01082 * *****
01083 */
01084 void investor_ui::asset_save()
01085 {
01086     string aid      = cgi("aid");
01087     string iid      = cgi("iid");
01088     string desc     = cgi("description");
01089     string aclass   = cgi("asset_class");
01090     string amnt     = cgi("amount");
01091     string curr     = cgi("currency");
01092
01093     if (oAsset.set_asset_id(aid) && oAsset.set_investor(iid) &&
oAsset.set_description(desc) && oAsset.
set_asset_class(aclass) && oAsset.set_amount(amnt) &&
oAsset.set_currency(curr) && oAsset.save())
01094     {
01095         cout << "<p class=\"markup\">Changes to asset, \"" << desc << "\", successfully saved.</p>";
01096     }
01097
01098     investor_id = atoi(iid.c_str());
01099     hh.header(investor_id);
01100     hh.show_toolBar("assets",get_full_name());
01101     show_inventory_form_body();
01102     hh.footer("assets","asset");
01103 }
01104

```

```

01105
01106 /* *****
01107 * asset_delete
01108 * *****
01109 */
01110 void investor_ui::asset_delete()
01111 {
01112     string iid      = cgi("iid");
01113     string aid      = cgi("aid");
01114
01115     if (set_investor_id(iid))
01116     {
01117         oAsset.set_asset_id(aid);
01118         if (oAsset.asset_delete())
01119         {
01120             std_message = std_message + "<p class=\"markup\">Asset #\" + aid + \"
successfully deleted.</p>";
01121         }
01122     }
01123     hh.header(investor_id);
01124     hh.show_toolBar("assets",get_full_name());
01125     show_inventory_form_body();
01126     hh.footer("assets","asset");
01127 }
01128
01129
01133 void investor_ui::goal_list()
01134 {
01135     hh.header(investor_id);
01136     hh.show_toolBar("goals",get_full_name());
01137     cout << "<div class=\"clearBoth\">\n";
01138     oGoal.list(investor_id, true);
01139     cout << "</div>\n";
01140     hh.aside(t_aside["goalInfo"]);
01141     hh.footer("goals","goal");
01142 }
01143
01144
01145 /* *****
01146 * goal_add
01147 * *****
01148 */
01149 void investor_ui::goal_add()
01150 //cgicc::Cgicc &formData
01151 {
01152     bool success = false;
01153     if (oGoal.load_fromEnv()) {success = oGoal.save();}
01154     if (success)
01155     {
01156         std_message = std_message + "<p class=\"markup\">New goal, \"\" +
oGoal.tbl[2].get_sValue() + "\", successfully added.</p>";
01157     }
01158     else
01159     {
01160         //TODO
01161     }
01162     hh.header(investor_id);
01163     hh.show_toolBar("goals",get_full_name());
01164     oGoal.list(investor_id);
01165     hh.footer("goals","goal");
01166 }
01167
01168
01169 /* *****
01170 * goal_edit
01171 * *****
01172 */
01173 void investor_ui::goal_edit()
01174 {
01175     oGoal.load_fromEnv();
01176
01177     hh.header(investor_id);
01178     hh.show_toolBar("goals",get_full_name());
01179
01180     if (oGoal.exists())
01181     {
01182         oGoal.load_fromDB();
01183         oGoal.show_edit_form(investor_id, true);
01184     }
01185
01186     hh.footer("goals","goal");
01187 }
01188
01189 /* *****
01190 * goale_Save
01191 * *****
01192 */

```

```

01193 void investor_ui::goal_save()
01194 {
01195     if (oGoal.load_fromEnv() && oGoal.save())
01196     {
01197         std_message = std_message + "<p class=\"markup\">Changes to goal, \"" +
oGoal.tbl[3].get_sValue() + "\", successfully saved.</p>";
01198     }
01199     hh.header(investor_id);
01200     hh.show_toolBar("goals",get_full_name());
01201     oGoal.list(investor_id, true);
01202     hh.footer("goals","goal");
01203 }
01204
01205 /* *****
01206 * goal_delete
01207 * *****
01208 */
01209 void investor_ui::goal_delete()
01210 {
01211     hh.header(investor_id);
01212     string gid = cgi("gid");
01213     if (oGoal.load_fromEnv() && oGoal.delete_bona())
01214     {
01215         cout << "<p class=\"markup\">Goal #" << gid << "; successfully deleted.</p>";
01216     }
01217     hh.show_toolBar("goals",get_full_name());
01218     oGoal.list(investor_id, true);
01219     hh.footer("goals","goal");
01220 }
01221
01222
01223
01224 /* *****
01225 * cash_flow_list *****
01226 * *****
01227 * *****/
01228 void investor_ui::cash_flow_list()
01229 {
01230     hh.header(investor_id);
01231     hh.show_toolBar("cfs",get_full_name());
01232     cout << "<div class=\"clearBoth\">\n";
01233     oCF.list(investor_id, true); // list existing ans show input form for new
01234     cout << "</div>\n";
01235     hh.aside(t_aside["cash_flowInfo"]);
01236     hh.footer("cfs","cash_flow");
01237 }
01238
01239
01240 /* *****
01241 * cash_flow_add
01242 * *****
01243 */
01244 void investor_ui::cash_flow_add()
01245 //cgicc::Cgicc &formData
01246 {
01247     bool success = false;
01248     if (oCF.load_fromEnv()) {success = oCF.save();}
01249     if (success)
01250     {
01251         std_message = std_message + "<p class=\"markup\">New cash flow, \"" +
oCF.tbl[2].get_sValue() + "\", successfully added.</p>";
01252     }
01253     hh.header(investor_id);
01254     hh.show_toolBar("cfs",get_full_name());
01255     oCF.list(investor_id);
01256     hh.footer("cfs","cash_flow");
01257 }
01258
01259
01260
01261 /* *****
01262 * cash_flow_edit
01263 * *****
01264 */
01265 void investor_ui::cash_flow_edit()
01266 {
01267     oCF.load_fromEnv();
01268     hh.header(investor_id);
01269     hh.show_toolBar("cfs",get_full_name());
01270     if (oCF.exists())
01271     {
01272         oCF.load_fromDB();
01273         oCF.show_edit_form(investor_id, true);
01274     }
01275 }
01276
01277

```

```

01278   hh.footer("cfs","cash_flow");
01279 }
01280
01281 /* *****
01282 * cash_flow_Save
01283 * *****
01284 */
01285 void investor_ui::cash_flow_save()
01286 {
01287     if (oCF.load_fromEnv() && oCF.save())
01288     {
01289         std_message = std_message + "<p class=\"markup\">Changes to cash flow, \" +
oCF.tbl[3].get_sValue() + "\", successfully saved.</p>";
01290     }
01291     hh.header(investor_id);
01292     hh.show_toolBar("cfs",get_full_name());
01293     oCF.list(investor_id, true);
01294     hh.footer("cfs","cash_flow");
01295 }
01296
01297 /* *****
01298 * cash_flow_delete
01299 * *****
01300 */
01301 void investor_ui::cash_flow_delete()
01302 {
01303     string gid = cgi("gid");
01304     if (oCF.load_fromEnv() && oCF.delete_bona())
01305     {
01306         std_message = std_message + "<p class=\"markup\">CF #" + gid + "; successfully
deleted.</p>";
01307     }
01308
01309     hh.header(investor_id);
01310     hh.show_toolBar("cfs",get_full_name());
01311     oCF.list(investor_id, true);
01312     hh.footer("cfs","cash_flow");
01313 }
01314
01315 /******
01320 /* *****
01321 * bool set_user_name(string s);
01322 *
01323 * *****
01324 */
01325 bool investor_ui::set_user_name(const std::string& s) {
01326     bool rc = true;
01327     if (s.length() < 4)
01328     {
01329         rc = false;
01330         error_message = error_message + "<li>user_name is too short (at least 4
characters required)</li>";
01331     }
01332     /* TODO: check if it is a valid email address *
01333     */
01334     if (rc) {user_name = s;}
01335     return rc;
01336 }
01337
01338 /* *****
01339 * bool set_first_name(const std::string& s);
01340 *
01341 * *****
01342 */
01343 bool investor_ui::set_first_name(const std::string& s) {
01344     bool rc = true;
01345     if (s.length() < 2)
01346     {
01347         rc = false;
01348         error_message = error_message + "<li>first_name is too short (at least 2
characters required)</li>";
01349     }
01350     if (rc) {first_name = s;}
01351     return rc;
01352 }
01353 /* *****
01354 * bool set_last_name(const std::string& s);
01355 *
01356 * *****
01357 */
01358 bool investor_ui::set_last_name(const std::string& s) {
01359     bool rc = true;
01360     if (s.length() < 2)
01361     {
01362         rc = false;
01363         error_message = error_message + "<li>last_name is too short (at least 2
characters required)</li>";

```

```

01364     }
01365     if (rc) {last_name = s;}
01366     return rc;
01367 }
01368 /* *****
01369 *   bool set_password(const std::string& s);
01370 *   *****
01371 *   *****
01372 */
01373 bool investor_ui::set_password(const std::string& s) {
01374     bool rc = true;
01375     if (s.length() < 4)
01376     {
01377         rc = false;
01378         error_message = error_message + "<li>password is too short (at least 4
characters required)</li>";
01379     }
01380     if (rc) {password = s;}
01381     return rc;
01382 }
01383
01384 bool investor_ui::set_currency(const std::string& s) {
01385     bool rc;
01386     if (oCurrency.set_curr(s)) //this returns true if the currency exists
01387     {
01388         this->currency = s;
01389         rc = true;
01390     }
01391     else
01392     {
01393         rc = false;
01394         error_message = error_message + "<li>not a valid currency</li>";
01395     }
01396     return rc;
01397 }
01398
01399 bool investor_ui::set_investor_id(const std::string& s) {
01400     int a = atoi(s.c_str());
01401     bool rc = true;
01402     if (this->exists(a))
01403     {
01404         investor_id = a;
01405     }
01406     else
01407     {
01408         rc = false;
01409         error_message = error_message + "<li>Sorry, invalid user</li>";
01410         investor_id = 0;
01411     }
01412     return rc;
01413 }
01414
01415 bool investor_ui::set_birth_date(const std::string& s)
01416 {
01417     bool rc = true;
01418     if (!is_valid_dateStr(s))
01419     {
01420         rc = false;
01421         error_message = error_message + "<li>invalid birth_date:" + s + "</li>";
01422     }
01423     // cout << "[" << s << "]";
01424     if (rc) {dateStr2tm(s, &birth_date);}
01425     return rc;
01426 }
01427
01428 bool investor_ui::set_simulate_till_age(const std::string& s)
01429 {
01430     bool rc = true;
01431     float sim_till = atof(s.c_str());
01432     if (sim_till <= age(0) + MIN_YEARS_TO_SIMULATE)
01433     {
01434         rc = false;
01435         error_message = error_message + "<li>\\"simulate till age\\" should be higher
than your current age + " + to_string(MIN_YEARS_TO_SIMULATE) + " years!</li>";
01436     }
01437     simulate_till_age = (rc) ? (int)round(sim_till) :
SIMULATE_TILL_DEFAULT;
01438     return rc;
01439 }
01440
01441 bool investor_ui::is_logged_in()
01442 {
01443     std::string s;
01444     bool rc = false;
01445     // first prune all users that should be logged-out!

```

```

01468 s = "UPDATE " + oConfig.tbl_prefix + "_investors SET is_logged_in = 0 WHERE
TIMESTAMPDIFF(MINUTE, last_activity_at, CURRENT_TIMESTAMP()) > 30;";
01469 if (db.runSQL(s)) {} else {error_message = error_message + "<li>Please
contact us.</li>";}
01470
01471 // then check if this particular user is logged in
01472 s = "SELECT is_logged_in FROM " + oConfig.tbl_prefix + "_investors WHERE investor_id = "
+ std::to_string(investor_id) + ";";
01473 db.res = db.stmt->executeQuery(s);
01474 while (db.res->next())
01475 {
01476     if (db.res->getString("is_logged_in") == "1") {rc = true;} else {rc = false;}
01477 }
01478 return rc;
01479 }
01480
01484 void investor_ui::logout()
01485 {
01486     std::string s;
01487     // first prune all users that should be logged-out!
01488     s = "UPDATE " + oConfig.tbl_prefix + "_investors SET is_logged_in = 0 WHERE investor_id
= " + to_string(investor_id) + ";";
01489     cout << s;
01490     if (db.runSQL(s)) {} else {error_message = error_message + "<li>Please
contact us, and provide the code \"i_ui::lo\".</li>";}
01491     home_screen();
01492 }
01493
01494 bool investor_ui::set_investor_id_fromEnv()
01495 {
01496     const CgiEnvironment& env = cgi.getEnvironment(); // const_form_iterator name =
cgi.getElement("name"); // const_form_iterator value = cgi.getElement("value");
01497     const_cookie_iterator iter;
01498     bool rc = false;
01499
01500     for(iter = env.getCookieList().begin(); iter != env.getCookieList().end(); ++iter)
01501     {if (iter->getName() == "iid") {rc = set_investor_id(iter->getValue());}}
01502
01503 #ifdef DEBUG
01504     debug_info = debug_info + "investor_id (from cookie) = " + to_string(
investor_id) + "<br/>";
01505 #endif
01506
01507     if (!(is_logged_in()))
01508     {
01509         investor_id = 0;
01510 #ifdef DEBUG
01511         debug_info = debug_info + "user is not logged in<br/>";
01512 #endif
01513     }
01514     return rc;
01515 }
01516
01517
01523 void investor_ui::login()
01524 {
01525     std::string s;
01526     s = "UPDATE " + oConfig.tbl_prefix + "_investors SET is_logged_in = 1, last_login_at =
NOW(), last_activity_at = NOW() WHERE investor_id=" + std::to_string(investor_id) + ";";
01527     if (db.runSQL(s)) {} else {error_message = error_message + "<li>Please
contact us - login error.</li>";}
01528 }
01529
01530
01531
01537 void investor_ui::update_last_activity_at()
01538 {
01539     std::string s;
01540     s = "UPDATE " + oConfig.tbl_prefix + "_investors SET is_logged_in = 1, last_activity_at
= NOW() WHERE investor_id=" + std::to_string(investor_id) + ";";
01541     if (db.runSQL(s)) {} else {error_message = error_message + "<li>Please
contact us - login error.</li>";} }
01542
01543
01549 void investor_ui::parse_goal_box(investment_problem *
oInvProbl, int g, string s_var, string s_desc)
01550 {
01551     string color_box = (goal_type_i2s[oInvProbl->goalZ[g].goal_type] == "unallocated") ? "
greyBox" : hh.get_box_color_class(oInvProbl->goalZ[g].color);
01552
01553     string goal_title = "<h3 class=\"" + color_box + "\">" + to_string(g) + ". " + oInvProbl->
goalZ[g].description + "&nbsp;<span class=\"glyphicon glyphicon-triangle-bottom\"></span></h3>";
01554     cout << hh.collapse_open("goalsAccordion", goal_title, g, false, true, "soft-" + color_box
);
01555
01556     // string collapse_open(string parent_id, string title, int nbr, bool add_triangle = true, bool is_open
= true, string xtra_body_class = "");

```

```

01557
01558
01559 /* switch (oConfig.layout)
01560 {
01561     case '2' :
01562
01563
01564         cout << "\n<div class=\"panel panel-default\">";
01565         cout << "\n<div class=\"panel-heading\">";
01566         cout << " <a data-toggle=\"collapse\" data-parent=\"#accordion2\" data-target=\"#collapse\" << g <<
"\n\">";
01567
01568
01569
01570         cout << "<h3 class=\"";
01571         hh.cout_box_color(oInvProbl->goalZ[g].color);
01572         cout << "\n\">" << oInvProbl->goalZ[g].description << " &nbsp;<span class=\"glyphicon
glyphicon-triangle-bottom\"></span></h3></a>";
01573         cout << "</div>";
01574         cout << "\n<div id=\"collapse\" << g << \"\" class=\"panel-collapse collapse in\">"; ///< note the "in":
the jqplot chart won't render in hidden div's :(
01575         cout << "\n<div class=\"panel-body\">";
01576         break;
01577         case '1':
01578         default :
01579         cout << "<div class=\"alert alert-info\">";
01580         cout << "<div class=\"";
01581         hh.cout_box_color(oInvProbl->goalZ[g].color);
01582         cout << "\n\">";
01583         cout << "<h4>" << g << ". " << oInvProbl->goalZ[g].description << "</h4> ";
01584         cout << "</div>";
01585     }*/
01586     std::cout.precision(2);
01587     cout << "<ul>";
01588     // -- description
01589     cout << "<li><b>description</b>:" << oInvProbl->goalZ[g] << "</li>";
01590
01591     if (oInvProbl->goalZ[g].realization_monthNbr < 0)
01592     {
01593     cout << "<li>This goal seems to be in the past ... please check</li>";
01594     }
01595     else
01596     {
01597     // -- benchmark
01598     cout << "<li><b>benchmark</b>:<br>";
01599     parse_bm(oInvProbl->goalZ[g].benchmark, g);
01600     cout << "</li>";
01601
01602     // -- savings plan
01603     cout << "<li><b>savings plan</b>: ";
01604     parse_savings_plan(oInvProbl, g);
01605     cout << "</li>";
01606     //----cout << "</div>" ; continue;
01607
01608     // -- plot
01609     cout << "<li><b>overview plot</b>";
01610     oInvProbl->javaGraph(g, s_var, s_desc);
01611     //-->     }
01612
01613     // -- remarks
01614     cout << "<li><b>remarks</b>: ";
01615     cout << oInvProbl->goalZ[g].remarks;
01616     cout << "</li>";
01617
01618     }
01619     cout << "</ul>";
01620
01621     cout << hh.collapse_close("goalsAccordion", g);
01622     // cout << "<script class=\"code\" type=\"text/javascript\">$(document).ready(function() {$('#collapse"
<< g << ").collapse('hide');})</scripts>";
01623 /*
switch (oConfig.layout)
01624 {
01625     case '2' :
01626         cout << "</div></div></div>";
01627         break;
01628     case '1' :
01629     default :
01630         cout << "</div>";
01631     }*/
01632
01633 }
01634
01635
01636
01642 void investor_ui::feedback()
01643 {
01644     int g;

```

```

01645 // load_from_db(to_string(investor_id));
01646
01647 hh.header(investor_id, true, true); // true as second param adds the javascript to
hide/show the plans
01648 // true as third param adds the jqPlot sources
01649 hh.show_toolBar("feedback",get_full_name());
01650
01651 hh.aside(t_aside["feedback"]);
01652
01653 investment_problem oInvestmentProbl(this->investor_id);
01654
01655 try
01656 {
01657     oInvestmentProbl.solve();
01658 }
01659 catch (int e)
01660 {
01661     #ifdef DEBUG
01662     cout << "# error in investment_problem.solve(): " << __FILE__ << "(" << __FUNCTION__
01663         << ")" on line " << to_string(__LINE__) << "<br># ERR: " << e;
01664     #endif
01665 }
01666
01667 //cout << oInvestmentProbl.get_nbr_goals();
01668 // now show the feedback on the screen
01669
01670 cout << this->parse_user_info(true);
01671
01672 cout << "<div class=\"panel-group\" id=\"goalsAccordion\">";
01673
01674 for (g = 1; g <= oInvestmentProbl.get_nbr_goals(); g++)
01675 {
01676     parse_goal_box(&oInvestmentProbl, g);
01677     /*std::cout.precision(2);
01678
01679     cout << "<div class=\"alert alert-info\">";
01680     cout << "<div class=\"";
01681     hh.cout_box_color(oInvestmentProbl.goalZ[g].color);
01682     cout << "\">";
01683     cout << "<h4>" << g << ". " << oInvestmentProbl.goalZ[g].description << "</h4> ";
01684     cout << "</div>";
01685     cout << "<ul>";
01686     // -- description
01687     cout << "<li><b>description</b>:" << oInvestmentProbl.goalZ[g] << "</li>";
01688
01689     if (oInvestmentProbl.goalZ[g].realization_monthNbr < 0)
01690     {
01691     cout << "<li><b>This goal seems to be in the past ... please check</li>";
01692     }
01693     else
01694     {
01695     // -- benchmark
01696     cout << "<li><b>benchmark</b>:<br>";
01697     parse_bm(oInvestmentProbl.goalZ[g].benchmark, g);
01698     cout << "</li>";
01699
01700     // -- savings plan
01701     cout << "<li><b>savings plan</b>:";
01702     parse_savings_plan(&oInvestmentProbl, g);
01703     cout << "</li>";
01704     //----cout << "</div>" ; continue;
01705
01706     // -- plot
01707     cout << "<li><b>overview plot</b>";
01708     oInvestmentProbl.javaGraph(g);
01709     //-->     }
01710
01711     // -- remarks
01712     cout << "<li><b>remarks</b>:";
01713     cout << oInvestmentProbl.goalZ[g].remarks;
01714     cout << "</li>";
01715
01716     }
01717     cout << "</ul>";
01718     cout << "</div>"; // alert alert-info*/
01719 } // for loop per goal
01720 cout << "</div>"; // goalsAccordion
01721
01722 show_total_portfolio(&oInvestmentProbl);
01723 hh.footer("feedback", "feedback");
01724 }
01725
01726
01731 void investor_ui::show_simulation()
01732 {
01733     int g;
01734     // load_from_db(to_string(investor_id)); //load the investor object

```

```

01735 hh.header(investor_id, true, true); // false as second param should not add
javascript to hide/show the plans
01736
01737 hh.show_toolBar("simulation",get_full_name());
01738 hh.aside(t_aside["simulation"]);
01739
01740 //int i; cout << "<BR>before init"; for (i=1; i <= 20; i++) cout << " -- " << i;
01741
01742 simulation oSimulation(this->investor_id);
01743 //cout << "<BR>before simulate"; for (i=1; i <= 20; i++) cout << " -- " << i;
01744
01745 oSimulation.simulate();
01746
01747 //cout << "<BR>after simulate"; for (i=1; i <= 20; i++) cout << " -- " << i;
01748
01749 cout << "<br>";
01750 cout << "<p>This page presents <strong>a simulation</strong>. A simulation is just <strong>one</strong>
possible scenario (something that could happen). Press the button [Simulate Again] to see another
simulation.</p><p>Last session You stated " << oSimulation.get_nbr_goals() << " goals, below you see what
might happen.</p>";
01751 cout << "<br>" << endl;
01752
01754 parse_simulation_duration_form(oSimulation.
get_nbrMonths2simulate());
01755
01757 parse_simulation_insight(&oSimulation);
01758
01760 //solve is included in simulate() {above} // oSimulation.solve();
01761 for (g = 1; g <= oSimulation.get_nbr_goals(); g++)
01762 {
01763     parse_goal_box(&oSimulation, g, oSimulation.goalZ[g].simulation_string, "simulated
evolution");
01764 /*
01765     //goalZ[g].load(); // TODO: load goals in stead of recalculating !!!!!!!!!!!!!!!!!!!!!!!!!!!!!
01766     cout << "<div class=\"alert alert-info\">";
01767     cout << "<div class=\"";
01768     hh.cout_box_color(oSimulation.goalZ[g].color);
01769     cout << "\">";
01770     cout << "<h4>" << g << ". " << oSimulation.goalZ[g].description << "</h4> ";
01771     cout << "</div>";
01772     cout << "<ul>";
01773     // -- description
01774     cout << "<li><b>description</b>:" << oSimulation.goalZ[g] << "</li>";
01775     // -- benchmark
01776     cout << "<li><b>benchmark</b>:<br>";
01777     parse_bm(oSimulation.goalZ[g].benchmark, g);
01778     cout << "</li>";
01779
01780     // -- savings plan
01781     cout << "<li><b>savings plan</b>:";
01782     parse_savings_plan(&oSimulation, g);
01783
01784
01785     //-- graph
01786     oSimulation.javaGraph(g, oSimulation.goalZ[g].simulation_string, "simulated evolution");
01787     cout << "<li><b>plot of the simulation</b>:</li>";
01788 //     cout << "<div id=\"chartOverview\"<g<<\" style=\"height:350px; width:650px;\"></div>";
01789     cout << "</ul>";
01790     cout << "</div>"; // alert alert-info
01791     */
01792 }
01793 cout << "<br>";
01794 hh.footer("simulation", "simulation");
01795 }
01796
01804 void investor_ui::show_follow_up()
01805 {
01806 // struct tm theDate;
01807 int g, simulation_length_mnths;
01808 string mnth_max, the_comment;
01809 // load_from_db(to_string(investor_id)); //load the investor object
01810 hh.header(investor_id, false, true); // false as second param should not add
javascript to hide/show the plans
01811 // second param
01812 hh.show_toolBar("followup",get_full_name());
01813 hh.aside(t_aside["follow_up"]);
01814
01815 follow_up oFollowUp(this->investor_id);
01816
01817 oFollowUp.simulate();
01818 cout << "<br>";
01819 cout << "<p>This page presents <strong>a simulation</strong> of what can possible happen in the future.
In other words it is a simulation of how a dashboard to follow up on investments could look like in the
future (given a certain market evolution). So, in an operational version of this software you would not have to
put in a number of months to simulate, the system would rather drop you an emeail with this overview updated
with today's market values.</p><p>Last session You stated " << oFollowUp.
get_nbr_goals() << " goals. Below you find more information on any recommended actions.</p>";

```

```

01820
01822     parse_simulation_duration_form(oFollowUp.
01823     get_nbrMonths2simulate(), "fu");
01823
01825     parse_simulation_insight(&oFollowUp);
01826
01828 // cout << "<BR> oFollowUp.get_nbrMonths2simulate() = " << oFollowUp.get_nbrMonths2simulate();
01829     for (g = 1; g <= oFollowUp.get_nbr_goals(); g++) oFollowUp.
01830     prepare_javaVars_colors(g, oFollowUp.
01831     get_nbrMonths2simulate());
01832     oFollowUp.parse_dashboard();
01833
01833     cout << "<h2>Detailed Follow-Up per Goal</h2>";
01835     for (g = 1; g <= oFollowUp.get_nbr_goals(); g++)
01836     {
01837         cout << "<div class=\"alert alert-info\">";
01838         cout << "<div class=\"";
01839         hh.cout_box_color(oFollowUp.goalZ[g].color);
01840         cout << "\">";
01841         cout << "<h4>" << g << ". " << oFollowUp.goalZ[g].description << "</h4> ";
01842         cout << "</div>";
01843         cout << "<ul>";
01844         // -- description
01845         cout << "<li><b>Description</b>:" << oFollowUp.goalZ[g] << "</li>";
01846         /* // -- benchmark
01847         cout << "<li><b>Benchmark</b>:<br>";
01848         parse_bm(oFollowUp.goalZ[g].benchmark, g);
01849         cout <<"</li>";
01850
01851         cout << "<li><b>Savings Plan</b>:<br>";
01852         parse_savings_plan(&oFollowUp, g);
01853         cout <<"</li>";
01854         */
01855         // -- simulation plot
01856         cout << "<li><b>How it might looks like in " << mnth_max <<":</b>";
01857         //oFollowUp.javaGraph(g, oFollowUp.goalZ[g].simulation_string, "simulated evolution",
01858         oFollowUp.get_nbrMonths2simulate()); // this sets also the followup color
01859         oFollowUp.parse_javaGraph( g, oFollowUp.goalZ[g].simulation_string,
01860         t_plotlabelsevol[5]);
01861         simulation_length_mnths = min(oFollowUp.get_nbrMonths2simulate(), ((
01862         goal_type_i2s[oFollowUp.goalZ[g].goal_type] == "rainy day savings") ?
01863         months2simulate() : oFollowUp.goalZ[g].realization_monthNbr));
01864         simulation_length_mnths = min(oFollowUp.get_nbrMonths2simulate(), oFollowUp.
01865         goalZ[g].realization_monthNbr);
01866         mnth_max = months2yms(simulation_length_mnths);
01867         the_comment = (simulation_length_mnths < oFollowUp.goalZ[g].realization_monthNbr) ?
01868         t_feedback[oFollowUp.goalZ[g].color] : t_feedback_post[oFollowUp.
01869         goalZ[g].color_followup] ;
01870         if (goal_type_i2s[oFollowUp.goalZ[g].goal_type] == "rainy day savings") the_comment
01871         = "Since this is your &quot;buffer&quot; investment it is not possible to say that this is a good of a bad
01872         situation now. The future will tell."; // to force
01873
01874         cout << "</li>";
01875         cout << "<li><span class=\"\" << hh.get_box_color_class(oFollowUp.
01876         goalZ[g].color_followup) << "\">The new color of this goal is <b>" << oFollowUp.
01877         goalZ[g].color_followup << "</b>. This means that after " << mnth_max << " and <b>if</b> you carefully
01878         followed the plan, this it <b>might</b> be that the dvice at that future moment would be: <b><i>"<<
01879         the_comment <<"</i></b></p><p>Note that this is only <b>one</b> possible outcome, press [enter], [F5] or the
01880         refresh button to see another possible simulation (eventually change the simulation horizon).<br>Note that not
01881         only you might put in different saving amounts or follow a different benchmark, also reasonable expectations
01882         about return, volatility and correlations might change!</li>";
01883         cout << "</ul>";
01884         cout << "</div>"; // alert alert-info
01885     }
01886
01887     cout << "<br>";
01888     hh.footer("followup", "follow_up");
01889 }
01890
01891 void investor_ui::parse_bm(std::map<int, float> benchmark, int goal_number)
01892 {
01893     std::map<int, string> asset_list = oAssetClass.get_assetList();
01894     unsigned i;
01895     float the_total = 0;
01896
01897     //normalize the benchmark:
01898     for (i = 1; i <= benchmark.size(); i++) the_total += benchmark[i];
01899     for (i = 1; i <= benchmark.size(); i++) benchmark[i] = benchmark[i] / the_total;
01900
01901     //xx cout << "\n<table class=\"bm\"><tr><td>";
01902
01903     // the table:
01904     /* cout << "\n<table style=\"font-size:0.8em;\" border=\"1\">";
01905
01906     for(i=1; i <= NBR_ASSET_CLASSES; i++)

```

```

01897 {
01898     cout << "<tr><td>" << asset_list[i] << "</td><td>" << round(benchmark[i]*10000)/100 << "%</td></tr>";
01899 }
01900 cout << "</table>";
01901 */
01902 //the pie chart:
01903 //xx cout << "</td><td>";
01904 // - the pie chart of the benchmark
01905 cout << "\n<script class=\"code\" type=\"text/javascript\">$(document).ready(function(){var data = [";
01906 for(i=1; i <= NBR_ASSET_CLASSES; i++)
01907 {
01908     cout << "[" << asset_list[i] << " : " << round(benchmark[i]*10000)/100 << "%'," << round(benchmark[i]
*10000)/100 << "];";
01909     if (i < NBR_ASSET_CLASSES) cout << ", ";
01910 }
01911 cout << "];";
01912 cout << "var plot" << goal_number << " = jQuery.jqplot ('chart" << goal_number << "', [data], ";";
01913 cout << "{";
01914 cout << "seriesColors : " << hh.assetClassColors << ", ";
01915 cout << "seriesDefaults: {";
01916     // Make this a pie chart.
01917 cout << "renderer: jQuery.jqplot.PieRenderer,";
01918 cout << "rendererOptions: {";
01919     // Put data labels on the pie slices.
01920     // By default, labels show the percentage of the slice.
01921 cout << "showDataLabels: true }},";
01922 cout << "legend: { show:true, location: 'e', showLabels:true}";
01923 cout << "};";
01924 cout << "});";
01925 //     cout << "<span id=\"floatLeft\" style=\"width:40%;\">";
01926 cout << "<div id=\"chart" << goal_number << "\" style=\"height:" << to_string(
NBR_ASSET_CLASSES * 34) << "px; width:600px; tr height:1em;\"></div>";
01927 //xx cout << "</td></tr></table>";
01928 }
01929
01930
01931 void investor_ui::show_goal_help_form_body()
01932 {
01933     cout << "<h3>Easy Goal Definition Screen</h3>";
01934     cout << "<p>Here are some common goals that might help you to set your goals.</p>";
01935
01936     cout << "\n<form method=\"post\" action=\"" + oConfig.soft_url + "?a=ghs\" class=\"
helpForm\">\n";
01937
01938     cout << "<div class=\"goalHelpDiv\">";
01939
01940     cout << hh.collapse_open("goalHelpDiv", "Retirement", 1, true, true);
01941     //cout << "<div class=\"alert alert-info\">";
01942     // cout << "<h3>Retirement</h3>\n";
01943     cout << "<p>Many people will consider their own retirement as the most important goal to save for.<br>In
order to plan for your retirement, you will need to do the following steps:";
01944     cout << "<ul><li>find out how much retirement income you will get from any sources outside this plan
(minimal income, legal retirement, etc.)</li>";
01945     cout << "<li>find out how much monthly expenses you will have (housing, housing related costs, food,
clothes, presents, etc.)</li>";
01946     cout << "<li>make the difference of the two above: that difference is the amount that you will need every
month. Fill it in below.</li></ul></p>";
01947     cout << "<fieldset>\n";
01948     cout << "<ol>\n";
01949     cout << "<li>\n";
01950     cout << "<label for=\"R_from_age\">age when you plan to retire</label><input type=\"number\" name=\"
R_from_age\" autofocus=\"true\" placeholder=\"65\" value=\"\"> the age at which you start to need the additional
income\n";
01951     cout << "</li>\n";
01952     cout << "<label for=\"R_till_age\">age till when the income is needed</label><input type=\"number\" name=
\"R_till_age\" autofocus=\"false\" placeholder=\"110\" value=\"110\"> 110 years old should be a safe limit,
but feel free to provide your own.\n";
01953     cout << "</li>\n";
01954     cout << "<label for=\"R_income\">monthly additional income desired</label><input type=\"number\" name=\"
R_income\" autofocus=\"false\" placeholder=\"1000\" value=\"\">\n";
01955     cout << "</li>\n";
01956     cout << "</ol>\n";
01957     cout << "</fieldset>";
01958     //cout << "</div>\n";
01959     cout << hh.collapse_close("goalHelpDiv", 1);
01960
01961     cout << hh.collapse_open("goalHelpDiv", "Rainy Day Savings", 2, true, false);
01962     // cout << "<div class=\"alert alert-info\">";
01963     // cout << "<h3>Rainy Day Savings</h3>\n";
01964     cout << "<p>A second investment goal that might come to your attention is \"a saving for the rainy day\".
It is an amount that you will always want to keep liquid and available in case something unforeseen
happens.";
01965     cout << "unforeseen reasons why you would need money are per definition those that you did do not plan
for (as separate goals). Examples are urgent repair on the house, legal assistance, repatriation,
hospitalisation costs (not covered by any insurance or the amount that you might have to advance), etc.</p>";
01966     cout << "<p>Troubles to decide on the amount? Try on of the following rule of thumb: two monthly salaries
plus two months rent (or equivalent rental potential if you live in your own place)</p>";

```

```

01972 cout << "<fieldset>\n";
01973 cout << "<ol>\n";
01974 cout << "<li>\n"; error_message = "";
01975
01976 cout << "<label for=\"RDS_amount\">amount of the rainy day savings:</label><input type=\"number\" name=\"
RDS_amount\" autofocus=\"true\" placeholder=\"5000\" value=\"\">\n";
01977 cout << "</li>\n";
01978 cout << "</ol>\n";
01979 cout << "</fieldset>";
01980 // cout << "</div>\n";
01981 cout << hh.collapse_close("goalHelpDiv", 2, true);
01982
01984 cout << hh.collapse_open("goalHelpDiv", "Insurances", 3, true, false);
01985 // cout << "<div class=\"alert alert-info\">";
01986 // cout << "<h3>Insurances</h3>\n";
01987 cout << "<p>While it is probably easier to keep insurances out of this analysis (ie. deduce insurance
payments that you plan to add from your saving capacity),";
01988 cout << "<p>it is very important to consider if you would need any of the following insurances and make
sure that you have them in place!</p>";
01989 cout << "<ul>";
01990 cout << "<li>life insurance: provides a certain amount of money for your close ones in case you pass
away, not only to cover cost related to that unfortunate moment, but also to make sure they can live the life
you want them to live without your contribution.</li>";
01991 cout << "<li>hospitalisation insurance: pays (preferably as third payer) for the event of
hospitalisation</li>";
01992 cout << "<li>income insurance: in case you lose your income due to loss of business or loss of ability to
work, you might want to have some replacement income in place.</li>";
01993 cout << "<li>insurance on any important properties: use as a rule of thumb that any utilitaraiian object
that you can not easily by back should be insured (eg. if you loose your computer it might be no problem to
replace it but if your house burns down you might not be able to recover from that: so you need an insurance
on the house!</li>";
01994 cout << "<li>TPL for your and those people for whom you're legally responsible</li>";
01995 cout << "<p>Any relevant professional insurance (eg. you might be legally responsible for the advice
that you give as consultant)</li>";
01996 cout << "</ul>";
01997 // cout << "</div>\n";
01998 cout << hh.collapse_close("goalHelpDiv", 3);
01999
02001 cout << hh.collapse_open("goalHelpDiv", "Savings for (grand-) children", 4, true, false);
02002 // cout << "<div class=\"alert alert-info\">";
02003 // cout << "<h3>Savings for (grand-) children</h3>\n";
02004 cout << "<p>Many people will want to set aside some money for their loved ones. Typical examples are kids
or grand-kids. Most probably you will want to save for their studies or provide them with a certain amount
to start their lives</p>";
02005 cout << "<fieldset>\n";
02006 cout << "<ul>\n";
02007 cout << "<li><b>First Kid</b>\n";
02008 cout << "<ol>\n";
02009 cout << "<li><label for=\"K1_desc\">description / name:</label><input type=\"text\" name=\"K1_desc\"
placeholder=\"Amelia\" value=\"\"></li>\n";
02010 cout << "<li><label for=\"K1_amount\">amount:</label><input type=\"number\" name=\"K1_amount\"
placeholder=\"10000\" value=\"\"></li>\n";
02011 cout << "<li><label for=\"K1_years\">due in</label><input type=\"number\" name=\"K1_years\"
placeholder=\"15\" value=\"\"> years</li>\n";
02012 cout << "</ol></li>\n";
02013 cout << "<li><b>Second Kid</b>\n";
02014 cout << "<ol>\n";
02015 cout << "<li><label for=\"K2_desc\">description / name:</label><input type=\"text\" name=\"K2_desc\"
placeholder=\"Amelia\" value=\"\"></li>\n";
02016 cout << "<li><label for=\"K2_amount\">amount:</label><input type=\"number\" name=\"K2_amount\"
placeholder=\"10000\" value=\"\"></li>\n";
02017 cout << "<li><label for=\"K2_years\">due in</label><input type=\"number\" name=\"K2_years\"
placeholder=\"15\" value=\"\"> years</li>\n";
02018 cout << "</ol></li>\n";
02019 cout << "<li><b>Third Kid</b>\n";
02020 cout << "<ol>\n";
02021 cout << "<li><label for=\"K3_desc\">description / name:</label><input type=\"text\" name=\"K3_desc\"
placeholder=\"Amelia\" value=\"\"></li>\n";
02022 cout << "<li><label for=\"K3_amount\">amount:</label><input type=\"number\" name=\"K3_amount\"
placeholder=\"10000\" value=\"\"></li>\n";
02023 cout << "<li><label for=\"K3_years\">due in</label><input type=\"number\" name=\"K3_years\"
placeholder=\"15\" value=\"\"> years</li>\n";
02024 cout << "</ol></li>\n";
02025 cout << "</ul>\n";
02026 cout << "</fieldset>";
02027 // cout << "</div>\n";
02028 cout << hh.collapse_close("goalHelpDiv", 4);
02029
02030
02032 cout << hh.collapse_open("goalHelpDiv", "Foreseeable expenses", 5, true, false);
02033 // cout << "<div class=\"alert alert-info\">";
02034 // cout << "<h3>Foreseeable expenses</h3>\n";
02035 cout << "Examples are:";
02036 cout << "<ul><li>repair to your dwelling house</li>";
02037 cout << "<li>a new car</li>";
02038 cout << "<li>etc.</li></ul>";
02039 cout << "<fieldset>\n";

```

```

02040     cout << "<ul>\n";
02041     cout << "    <li><b>First Expense</b>\n";
02042     cout << "    <ol>\n";
02043     cout << "    <li><label for=\"FE1_desc\">description:</label><input type=\"text\" name=\"FE1_desc\"
placeholder=\"Repair House\" value=\"\"></li>\n";
02044     cout << "    <li><label for=\"FE1_amount\">amount:</label><input type=\"number\" name=\"FE1_amount\"
placeholder=\"10000\" value=\"\"></li>\n";
02045     cout << "    <li><label for=\"FE1_years\">due in</label><input type=\"number\" name=\"FE1_years\"
placeholder=\"15\" value=\"\"> years</li>\n";
02046     cout << "  </ol></li>\n";
02047     cout << "    <li><b>Second Expense</b>\n";
02048     cout << "    <ol>\n";
02049     cout << "    <li><label for=\"FE2_desc\">description:</label><input type=\"text\" name=\"FE2_desc\"
placeholder=\"New Car\" value=\"\"></li>\n";
02050     cout << "    <li><label for=\"FE2_amount\">amount:</label><input type=\"number\" name=\"FE2_amount\"
placeholder=\"10000\" value=\"\"></li>\n";
02051     cout << "    <li><label for=\"FE2_years\">due in</label><input type=\"number\" name=\"FE2_years\"
placeholder=\"15\" value=\"\"> years</li>\n";
02052     cout << "  </ol></li>\n";
02053     cout << "    <li><b>Third Expense</b>\n";
02054     cout << "    <ol>\n";
02055     cout << "    <li><label for=\"FE3_desc\">description:</label><input type=\"text\" name=\"FE3_desc\"
placeholder=\"Maintainance Yacht\" value=\"\"></li>\n";
02056     cout << "    <li><label for=\"FE3_amount\">amount:</label><input type=\"number\" name=\"FE3_amount\"
placeholder=\"10000\" value=\"\"></li>\n";
02057     cout << "    <li><label for=\"FE3_years\">due in</label><input type=\"number\" name=\"FE3_years\"
placeholder=\"15\" value=\"\"> years</li>\n";
02058     cout << "  </ol></li>\n";
02059     cout << "</ul>\n";
02060     cout << "</fieldset>";
02061 //   cout << "</div>\n";
02062     cout << hh.collapse_close("goalHelpDiv",5);
02063
02065     cout << hh.collapse_open("goalHelpDiv", "Foreseeable expenses", 6, true, false);
02066 //   cout << "<div class=\"alert alert-info\">";
02067 //   cout << "<h3>Life Dream</h3>\n";
02068     cout << "<p>Your personal life dream: art purchase, travel, yacht, a gift to a good cause, etc.</p>";
02069     cout << "<fieldset>\n";
02070     cout << "<ul>\n";
02071     cout << "    <li><b>First Life Dream</b>\n";
02072     cout << "    <ol>\n";
02073     cout << "    <li><label for=\"LD1_desc\">description:</label><input type=\"text\" name=\"LD1_desc\"
placeholder=\"Yacht\" value=\"\"></li>\n";
02074     cout << "    <li><label for=\"LD1_amount\">amount:</label><input type=\"number\" name=\"LD1_amount\"
placeholder=\"100000\" value=\"\"></li>\n";
02075     cout << "    <li><label for=\"LD1_age\">age to realize this dream</label><input type=\"number\" name=\"
LD1_age\" placeholder=\"55\" value=\"\"> years old</li>\n";
02076     cout << "  </ol></li>\n";
02077     cout << "    <li><b>Second Life Dream</b>\n";
02078     cout << "    <ol>\n";
02079     cout << "    <li><label for=\"LD2_desc\">description:</label><input type=\"text\" name=\"LD2_desc\"
placeholder=\"World Travel\" value=\"\"></li>\n";
02080     cout << "    <li><label for=\"LD2_amount\">amount:</label><input type=\"number\" name=\"LD2_amount\"
placeholder=\"100000\" value=\"\"></li>\n";
02081     cout << "    <li><label for=\"LD2_age\">age to realize this dream</label><input type=\"number\" name=\"
LD2_age\" placeholder=\"65\" value=\"\"> years old</li>\n";
02082     cout << "  </ol></li>\n";
02083     cout << "    <li><b>Third Life Dream</b>\n";
02084     cout << "    <ol>\n";
02085     cout << "    <li><label for=\"LD3_desc\">description:</label><input type=\"text\" name=\"LD3_desc\"
placeholder=\"Gift to Good Cause\" value=\"\"></li>\n";
02086     cout << "    <li><label for=\"LD3_amount\">amount:</label><input type=\"number\" name=\"LD3_amount\"
placeholder=\"100000\" value=\"\"></li>\n";
02087     cout << "    <li><label for=\"LD3_age\">age to realize this dream</label><input type=\"number\" name=\"
LD3_age\" placeholder=\"75\" value=\"\"> years old</li>\n";
02088     cout << "  </ol></li>\n";
02089     cout << "</ul>\n";
02090     cout << "</fieldset>";
02091 //   cout << "</div>\n";
02092     cout << hh.collapse_close("goalHelpDiv",6);
02093
02094     cout << "<button class=\"btn btn-primary\" type=\"submit\" value=\"Submit\">ADD ALL the goals that are
filled out</button>\n";
02095     cout << "<button class=\"btn btn-primary\" type=\"reset\" value=\"reset\"></button>";
02096     cout << "<button class=\"btn btn-primary\" onclick=\"window.location.href='/'\" <<
oConfig.soft_url << \"a=gl\">\" << t_cancel << "</button>";
02097     cout << "<input type=\"hidden\" name=\"a\" value=\"ghs\" />\n";
02098     cout << "</form>\n";
02099
02100     cout << "</div>"; // goalHelpDiv
02101 }
02102
02106 void investor_ui::goal_help_form()
02107 {
02108     hh.header(investor_id);
02109     show_goal_help_form_body();

```

```

02110     hh.footer("", "");
02111 }
02112
02113
02118 void investor_ui::goal_help_save()
02119 {
02120     aGOAL the_goal(investor_id);
02121     int k;
02122
02124     string R_from_age = cgi("R_from_age");
02125     string R_till_age = cgi("R_till_age");
02126     string R_income = cgi("R_income");
02127     if((R_from_age != "") && (R_till_age != "") && (R_income != ""))
02128     {
02129         float from_age = atof(R_from_age.c_str());
02130         float till_age = atof(R_till_age.c_str());
02131         if (from_age <= age())
02132         {
02133             from_age = this->age(0);
02134             std_message = std_message + "<li>I changed the from_date of the retirement goal
to today (one cannot plan for past income)</li>";
02135         }
02136         if (from_age < till_age)
02137         {
02138             string f_dateStr = mNbr2dateStr(age2monthNbr(from_age));
02139             string t_dateStr = mNbr2dateStr(age2monthNbr(till_age));
02140 #ifndef DEBUG
02141             debug_info = debug_info + "<br>f_dateStr = " + f_dateStr + ", mNbr = " + to_string(
age2monthNbr(from_age)) + ", age = " + to_string(from_age);
02142             debug_info = debug_info + "<br>t_dateStr = " + t_dateStr + ", mNbr = " + to_string(
age2monthNbr(till_age)) + ", age = " + to_string(till_age);
02143 #endif
02144             if (the_goal.set_amount(R_income) && the_goal.set_from_date(f_dateStr) &&
the_goal.set_till_date(t_dateStr))
02145             {
02146                 the_goal.set_goal_type(to_string(goal_type_s2i["income from/to"]));
02147                 the_goal.set_description("Extra income during retirement");
02148                 the_goal.set_currency(this->currency);
02149                 the_goal.set_priority("1");
02150                 the_goal.set_frequency("M");
02151                 the_goal.set_realization_age(dateStr2Age(t_dateStr), this->
age(0));
02152                 the_goal.set_max_shortfall("0");
02153                 the_goal.add_to_db();
02154             }
02155         }
02156         else
02157         {
02158             error_message = error_message + "<li>Sorry, I failed to add the RETIREMENT
goal (note that the from-date must be in the future but before the end date; provided: from_age=" +
R_from_age + ", till_age=" + R_till_age + "), please retry!</li>";
02159         }
02160     }
02161
02163     string RDS_amount = cgi("RDS_amount");
02164     //the_goal = new aGOAL; //xxxxxxxxxxxxxxx
02165     if(RDS_amount != "")
02166     {
02167         if (atof(RDS_amount.c_str()) > 0)
02168         {
02169             string end_dateStr = mNbr2dateStr(age2monthNbr(
simulate_till_age));
02170             the_goal.set_amount(RDS_amount);
02171             the_goal.set_from_date(tm2DateStr(get_tm(0)));
02172             the_goal.set_till_date(end_dateStr);
02173             the_goal.set_goal_type(to_string(goal_type_s2i["rainy day savings"]));
02174             the_goal.set_description("Rainy day savings");
02175             the_goal.set_currency(this->currency);
02176             the_goal.set_priority("3");
02177             the_goal.set_frequency("0");
02178             the_goal.set_realization_age(to_string(
simulate_till_age), this->age(0));
02179             the_goal.set_max_shortfall("0");
02180             the_goal.add_to_db();
02181         }
02182         else
02183         {
02184             error_message = error_message + "<li>Sorry, the RAINY DAY savings should be
a positive amount.</li>";
02185         }
02186     }
02187
02188     string g_desc;
02189     string g_amount;
02190     string g_years ;
02191     string g_age;
02192

```

```

02193 float g_age_fl;
02194
02195
02197 for (k = 1; k <=3; k++)
02198 {
02199     g_desc = cgi("K" + to_string(k) + "_desc");
02200     g_amount = cgi("K" + to_string(k) + "_amount");
02201     g_years = cgi("K" + to_string(k) + "_years");
02202
02203     if (g_desc + g_amount + g_years != "")
02204     {
02205         if (atof(g_years.c_str()) > 0)
02206         {
02207             the_goal.set_amount(g_amount);
02208             g_age_fl = this->age(0) + atof(g_years.c_str());
02209             the_goal.set_realization_age(to_string(g_age_fl), this->
age(0));
02210             the_goal.set_from_date(mNbr2dateStr(
age2monthNbr(g_age_fl)));
02211             the_goal.set_till_date(the_goal.from_date);
02212             the_goal.set_goal_type(to_string(goal_type_s2i["amount@date"]));
02213             the_goal.set_description(g_desc);
02214             the_goal.set_currency(this->currency);
02215             the_goal.set_priority("5");
02216             the_goal.set_frequency("0");
02217             the_goal.set_max_shortfall("0");
02218             the_goal.add_to_db();
02219         }
02220     }
02221     else
02222     {
02223         error_message = error_message + "<li>The spending for your kids has to be
in the FUTURE.</li>";
02224     }
02225 }
02226
02228 for (k = 1; k <=3; k++)
02229 {
02230     g_desc = cgi("FE" + to_string(k) + "_desc");
02231     g_amount = cgi("FE" + to_string(k) + "_amount");
02232     g_years = cgi("FE" + to_string(k) + "_years");
02233     if (g_desc + g_amount + g_years != "")
02234     {
02235
02236         if (atof(g_years.c_str()) > 0)
02237         {
02238             the_goal.set_amount(g_amount);
02239             g_age_fl = this->age(0) + atof(g_years.c_str());
02240             the_goal.set_realization_age(to_string(g_age_fl), this->
age(0));
02241             the_goal.set_from_date(mNbr2dateStr(
age2monthNbr(g_age_fl)));
02242             the_goal.set_till_date(the_goal.from_date);
02243             the_goal.set_goal_type(to_string(goal_type_s2i["amount@date"]));
02244             the_goal.set_description(g_desc);
02245             the_goal.set_currency(this->currency);
02246             the_goal.set_priority("7");
02247             the_goal.set_frequency("0");
02248             the_goal.set_max_shortfall("0");
02249             the_goal.add_to_db();
02250         }
02251     }
02252     else
02253     {
02254         error_message = error_message + "<li>The foreseeable spendings have to be
in the FUTURE.</li>";
02255     }
02256 }
02257
02259 for (k = 1; k <=3; k++)
02260 {
02261     g_desc = cgi("LD" + to_string(k) + "_desc");
02262     g_amount = cgi("LD" + to_string(k) + "_amount");
02263     g_age = cgi("LD" + to_string(k) + "_age");
02264
02265     if (g_desc + g_amount + g_age != "")
02266     {
02267
02268         g_age_fl = atof(g_age.c_str());
02269         if (g_age_fl > age(0))
02270         {
02271             the_goal.set_amount(g_amount);
02272             the_goal.set_realization_age(g_age, this->age(0));
02273             the_goal.set_from_date(mNbr2dateStr(
age2monthNbr(g_age_fl)));
02274             the_goal.set_till_date(the_goal.from_date);
02275             the_goal.set_goal_type(to_string(goal_type_s2i["amount@date"]));

```

```

02276     the_goal.set_description(g_desc);
02277     the_goal.set_currency(this->currency);
02278     the_goal.set_priority("9");
02279     the_goal.set_frequency("0");
02280     the_goal.set_max_shortfall("0");
02281     the_goal.add_to_db();
02282 }
02283 else
02284 {
02285     error_message = error_message + "<li>The spending for your kids has to be
in the FUTURE.</li>";
02286 }
02287 }
02288 }
02289
02290 if (error_message == "")
02291 {
02292     hh.header(investor_id);
02293     hh.show_toolBar("goals",get_full_name());
02294     oGoal.list(investor_id, true); // show the t_before_list and list existing
and show input form for new
02295     hh.aside(t_aside["goalInfo"]);
02296     hh.footer("goals", "goal");
02297 }
02298 else
02299 {
02300     hh.header(0);
02301     hh.show_toolBar("register");
02302     show_goal_help_form_body();
02303     hh.aside(t_aside["goalHelpFail"]);
02304     hh.footer("goals", "goal");
02305 }
02306 }
02307 } // goal_help_save
02308
02314 void investor_ui::parse_savings_plan(
investment_problem* oInvProbl, int g)
02315 {
02316     struct tm theDate = get_tm(0);
02317     int i;
02318     cout << endl;
02319     cout << "<button id=\"btnShowSP\" << g << "\>Show</button>";
02320     cout << "<button id=\"btnHideSP\" << g << "\>Hide</button>";
02321     cout << "<div id=\"savingsPlan\" << g << "\>";
02322     cout << endl;
02323     cout << "<table border=\"1\" class=\"savingPlan\">";
02324     //cout << "<tr><th>Date</th><th>Amount</th><th>Currency</th></tr>" << endl;
02325     cout << "<tr><th>Date</th><th>Amount</th></tr>" << endl;
02326     for (i=0; i<= oInvProbl->goalZ[g].realization_monthNbr;i++)
02327     {
02328         if (oInvProbl->goalZ[g].saving_plan[i] > 0)
02329         { // only show an entry in the table if there are savings
02330             cout << "<tr>";
02331             cout << "<td>" << tm2DateStr(theDate) << "</td>";
02332             //cout << "<td>" << to_string(std::printf("%.2f",oInvProbl->goalZ[g].saving_plan[i])) << "</td>";
02333             cout << "<td>" << curr_format(oInvProbl->goalZ[g].saving_plan[i], oInvProbl->
goalZ[g].currency) << "</td>";
02334             //cout << "<td>" << oInvProbl->goalZ[g].currency << "</td>";
02335             cout << "</tr>" << endl;
02336         }
02337         if (theDate.tm_mon <= 10) theDate.tm_mon++;
02338         else
02339         {
02340             theDate.tm_year++;
02341             theDate.tm_mon = 0;
02342         }
02343     }
02344     cout << "</table></div>";
02345     cout << "<button id=\"btnHide2SP\" << g << "\>Hide Savings Plan</button>"; // second hide button under
the list
02346 }
02347
02348
02356 void investor_ui::show_total_portfolio(
investment_problem * oInvProbl)
02357 {
02358     int g, m;
02359     unsigned mNbr, ac;
02360     std::map<int, float> portfolio_composition;
02361     float savings_for_goal;
02362
02363     cout << "<div class=\"alert alert-info\">\n";
02364     cout << "<h2>Total Portfolio</h2>\n";
02365     cout << "<p>While all goals can be managed in separate portfolios, it might as well suit you to keep all
your assets in one portfolio. If you choose to do so, it should look more or less as follows.<br>Below we
show the content of your ideal portfolio on different short time horizons. This should allow you to decide on
what benchmark you need for the next year.</p>";

```

```

02366     cout << "<p>On longer horizons, we recommended to re-run and update this tool in the future (eg. once per
year) and adapt your portfolio accordingly.</p>\n";
02367     cout << "<p>Below we show your average composition on " + to_string(oConfig.
total_portf_monthNbrs.size()) + " different horizons so that you can choose to hold a
portfolio equal to the average or make it evolve.</p>\n";
02368     cout << "<ul>";
02369     for (mNbr = 1; mNbr <= oConfig.total_portf_monthNbrs.size(); mNbr++)
02370     {
02371         for(ac=1; ac <= NBR_ASSET_CLASSES; ac++) portfolio_composition[ac] = 0;
02372         for (g = 1; g <= oInvProbl->get_nbr_goals(); g++)
02373         {
02374             savings_for_goal = 0;
02375             for (m = 0; m <= min(oConfig.total_portf_monthNbrs[mNbr-1], oInvProbl->
goalZ[g].realization_monthNbr); m++)
02376             {
02377                 savings_for_goal += (oInvProbl->goalZ[g].saving_plan[m]); // - oInvProbl->goalZ[g].means_goal[m] -
((m == oInvProbl->goalZ[g].realization_monthNbr)?oInvProbl->goalZ[g].amount:0.0));
02378                 //cout << "<br>oInvProbl->goalZ[" << g << "].means_goal[" << m << "])" <<
oInvProbl->goalZ[g].means_goal[m];
02379             }
02380             for (ac=1; ac <= NBR_ASSET_CLASSES; ac++)
02381             {
02382                 portfolio_composition[ac] += oInvProbl->goalZ[g].benchmark[ac] * savings_for_goal;
02383             }
02384         }
02385         // output the results
02386         cout << "<li>On a horizon of " << to_string(oConfig.
total_portf_monthNbrs[mNbr-1]) << " months your portfolio should look approximately as
follows:<br>";
02387         parse_bm(portfolio_composition, 1000 + mNbr); // add 1000 to avoid re-using an existing number
(assuming the user uses less than 1000 goals
02388         cout << "</li>\n";
02389     }
02390     cout << "</ul>";
02391     cout << "</div>\n";
02392 }
02393 }
02394
02398 void investor_ui::parse_simulation_duration_form(int nbrMonths,
string action)
02399 {
02400     int nYrs = (int)floor(nbrMonths / 12);
02401     int nMnths = nbrMonths - nYrs * 12;
02402     cout << "<div class=\"alert alert-info\">";
02403     cout << "<n<form method=\"post\" action=\"" + oConfig.soft_url + "\" id=\"
simulation_duration\">\n";
02404     cout << "<fieldset>\n";
02405     cout << "<h2>" << t_simulation_duration << "</h2>";
02406     cout << "<ol>\n";
02407     cout << "<li>\n";
02408     cout << "<label for=\"nYears\" id=\"follow_up_nbrYrs\" align=\"left\">years</label>"
02409     << "<input type=\"number\" name=\"nYears\" autofocus=\"true\" required min=\"0\" max=\"" << (int)(
MAX_MNTHS_2_SIMULATE / 12) << "\" placeholder=\"1\" value=\"" << nYrs << "\">\n"
";
02410     cout << "</li>\n<li>";
02411     cout << "<label for=\"nMonths\" id=\"follow_up_nbrMonths\" >months</label><input type=\"number\" name=\"
nMonths\" autofocus=\"false\" required min=\"0\" max=\"" << MAX_MNTHS_2_SIMULATE << "\"
placeholder=\"0\" value=\"" << nMnths << "\">\n";
02412     cout << "</li>\n<li>";
02413     cout << "<button class=\"btn btn-primary\" type=\"submit\" value=\"Submit\" id=\"sim_dur\">Simulate
Again</button>\n\n";
02414     // cout << "<button type=\"reset\" value=\"reset\">Reset</button>\n\n";
02415     cout << "</li></ol>\n";
02416     cout << "</fieldset>";
02417     cout << "<input type=\"hidden\" name=\"a\" value=\"" << action << "\" />\n";
02418     cout << "</form>\n";
02419     cout << "</div>";
02420 }
02421
02425 void investor_ui::parse_simulation_insight (
simulation *oSIm)
02426 {
02427     cout << "<div class=\"alert alert-info\">";
02428     cout << "<H2>How this simulation over " << months2yms(oSim->
get_nbrMonths2simulate()) << " worked out</H2>";
02429     oSim->plot_marketevol();
02430     cout << "<div id=\"chartMEvol\" style=\"height:350px; width:650px;\"></div>";
02431     cout << "<p>To see another simulation: " << "<button class=\"btn btn-primary\" type=\"submit\" value=\"
Submit\" id=\"sim_dur\">Simulate Again</button>\n\n";
02432     cout << "</div>";
02433 }

```

8.39 market.class.cpp File Reference

Classes

- class [market](#)

8.40 market.class.cpp

```

00001
00011 class market
00012 {
00013 public:
00014     market (int person = 0);
00015
00016     float* assetClass_covar = new float[(NBR_ASSET_CLASSES + 1) * (
NBR_ASSET_CLASSES + 1)];
00017     float  assetClass_mu[(NBR_ASSET_CLASSES + 1)];
00018     std::map <int, string> assetClass_name;
00019
00020 /** void set_mu(int person);
00021 void set_covar(int person);
00022 */
00023
00024     //the personalized expected returns and covariances
00025     std::map<int, float> ER;
00026     std::map<int, std::map<int, float> > covar;
00027     void load_ER (int person = 0, char person_type = 'i');
00028     void save_ER (int person = 0, char person_type = 'i');
00029     void load_covar(int person, char person_type = 'i');
00030     void save_covar(int person, char person_type = 'i');
00031
00032
00033
00034 protected:
00035     void reload_ER (int person, char person_type = 'i');
00036     void reload_covar(int person, char what2do, char person_type = 'i');
00037
00038
00039 };
00040
00044 market::market(int person)
00045 {
00046     // if person > 0
00047     // THEN
00048     // Check if personalized assumptions exist,
00049     // - if yes, load them, ie pass person to the set-functions
00050     // - if not, then load the defaults (pass person = 0 to the set-functions.
00051     // ELSE : pass 0 to the set-functions
00052
00053     // load_ER(person);
00054     // load_covar(person);
00055 }
00056
00057
00061 void market::load_ER(int person, char person_type)
00062 {
00063     int ac, count = 0;
00064     string s;
00065
00066
00067     //cout << "<BR>PERSON:" << person;
00068     if (person != 0 )
00069     {
00070         s = "SELECT * FROM " + oConfig.tbl_prefix + "_ER_person WHERE person = " + to_string(
person) + " AND person_type = 'i'";
00071         //cout << s;
00072         db.res = db.stmt->executeQuery(s);
00073         while (db.res->next ())
00074         {
00075             ac = db.getInt("asset_class");
00076             ER[ac] = db.getFloat("E_R");
00077             assetClass_mu[ac] = db.getFloat("E_R"); //TODO : eliminate this
00078             //cout << "<br>ER_person[" << ac << "]= " << ER[ac];
00079             count++;
00080         }
00081         // get all the asset class names
00082         string s = "SELECT * FROM " + oConfig.tbl_prefix + "_asset_classes";
00083         db.res = db.stmt->executeQuery(s);
00084         while (db.res->next ())
00085         {
00086             ac = db.getInt("asset_class_id");
00087             assetClass_name[ac] = db.res->getString("asset_class_name");
00088         }
00089     }
00090
00091     if (count < NBR_ASSET_CLASSES)

```

```

00092 { // if we did not find a personalized expectation for each asset class, then we load the default values
00093 // TODO allow this to be per organization, advisor, etc.
00094 s = "SELECT * FROM " + oConfig.tbl_prefix + "_asset_classes";
00095 db.res = db.stmt->executeQuery(s);
00096 while (db.res->next())
00097 {
00098     ac = db.getInt("asset_class_id");
00099     ER[ac] = db.getFloat("E_R");
00100     assetClass_mu[ac] = db.getFloat("E_R"); //TODO : eliminate this
00101     assetClass_name[ac] = db.res->getString("asset_class_name");
00102 }
00103 }
00104 }
00105
00109 void market::save_ER(int person, char person_type)
00110 {
00111     int ac;
00112     string s;
00113     s = "DELETE FROM " + oConfig.tbl_prefix + "_ER_person WHERE person = " + to_string(
00114 person) + " AND person_type = 'i'";
00114     if (db.runSQL(s) {} else {}){error_message = error_message + "<li>Please contact us. #errno
E015</li>";}
00115     s = "INSERT INTO " + oConfig.tbl_prefix + "_ER_person (asset_class, E_R, person,
00116 person_type) VALUES ";
00117     for(ac=1; ac <= NBR_ASSET_CLASSES; ac++)
00118     {
00119         if (ac > 1) s = s + ", ";
00119         s = s + "(" + to_string(ac) + ", " + to_string(ER[ac]) + ", "
00120             + to_string(person) + ", \"i\"";
00121     }
00122     s = s + ";";
00123     //error_message = error_message + "<br>" + s;
00124     if (db.runSQL(s) {}
00125     else
00126     {
00127         error_message = error_message + "<li>Please contact us. #errno E015</li>";
00128     }
00129 }
00130
00135 void market::load_covar(int person, char person_type)
00136 {
00137     int ac_i, ac_j, count = 0;
00138     if (person != 0)
00139     {
00140         string s = "SELECT * FROM " + oConfig.tbl_prefix + "_covar_person WHERE person = " +
00141 to_string(person) + " AND person_type = 'i'";
00142         //cout << s;
00143         db.res = db.stmt->executeQuery(s);
00144         while (db.res->next())
00145         {
00146             ac_i = db.getInt("asset_class_i");
00147             ac_j = db.getInt("asset_class_j");
00148             covar[ac_i][ac_j] = db.getFloat("covar");
00149             //cout << "<br>covar[" << ac_i << ", " << ac_j << "] = " << covar[ac_i][ac_j];
00150             assetClass_covar[INDEX(ac_i,ac_j)] = assetClass_covar[
00151 INDEX(ac_j,ac_i)] = db.getFloat("covar");
00152             count++;
00153         }
00154         if (count < NBR_ASSET_CLASSES * NBR_ASSET_CLASSES)
00155         {
00156             // TODO allow this to be per organization, advisor, etc.
00157             string s = "SELECT * FROM " + oConfig.tbl_prefix + "_covar";
00158             db.res = db.stmt->executeQuery(s);
00159             while (db.res->next())
00160             {
00161                 ac_i = db.getInt("asset_class_i");
00162                 ac_j = db.getInt("asset_class_j");
00163                 covar[ac_i][ac_j] = db.getFloat("covar");
00164                 assetClass_covar[INDEX(ac_i,ac_j)] = assetClass_covar[
00165 INDEX(ac_j,ac_i)] = db.getFloat("covar");
00166             }
00167         }
00168
00172 void market::save_covar(int person, char person_type)
00173 {
00174     int ac_i, ac_j;
00175     string s;
00176     s = "DELETE FROM " + oConfig.tbl_prefix + "_covar_person WHERE person = " + to_string(
00177 person) + " AND person_type = 'i'";
00177     if (db.runSQL(s) {} else {}){error_message = error_message + "<li>Please contact us. #errno
E018</li>";}
00178     s = "INSERT INTO " + oConfig.tbl_prefix + "_covar_person (asset_class_i, asset_class_j,
00179 covar, person, person_type) VALUES ";
00179     for(ac_j = 1; ac_j <= NBR_ASSET_CLASSES; ac_j++)

```

```

00180 {
00181     for(ac_i=1; ac_i <= NBR_ASSET_CLASSES; ac_i++)
00182     {
00183         if ((ac_i > 1) || (ac_j > 1)) s = s + ", ";
00184         s = s + "(" + to_string(ac_i) + ", " + to_string(ac_j) + ", " + to_string(
covar[ac_i][ac_j]) + ", "
00185             + to_string(person) + ", \"i\")";
00186     }
00187 }
00188 s = s + ";";
00189 //error_message = error_message + "<br>" + s;
00190 if (db.runSQL(s) {})
00191 else
00192 {
00193     error_message = error_message + "<li>Please contact us. #errno E018</li>";
00194 }
00195 }
00196
00197
00198
00199
00205 /*void market::set_mu(int person)
00206 {
00207     int nbr = 0;
00208     db.res = db.stmt->executeQuery("SELECT * FROM " + oConfig.tbl_prefix + "_asset_classes;");
00209     while (db.res->next())
00210     {
00211         nbr
            = atoi(db.res->getString("asset_class_id").c_str());
00212         this->assetClass_mu[nbr] = atof(db.res->getString("E_R").c_str());
00213         //assetClass_mu[nbr] = pow((assetClass_mu[nbr] + 1), 0.0833333) - 1;
00214         this->assetClass_name[nbr] = db.res->getString("asset_class_name");
00215     }
00216 }*/
00222 /*void market::set_covar(int person)
00223 {
00224     float sigma_ij = 0;
00225     int i, j;
00226     db.res = db.stmt->executeQuery("SELECT * FROM " + oConfig.tbl_prefix + "_covar;");
00227     while (db.res->next())
00228     {
00229         i
            = atoi(db.res->getString("asset_class_i").c_str());
00230         j
            = atoi(db.res->getString("asset_class_j").c_str());
00231         sigma_ij
            = atof(db.res->getString("covar").c_str());
00232         this->assetClass_covar[INDEX(i,j)] = sigma_ij;
00233         this->assetClass_covar[INDEX(j,i)] = sigma_ij;
00234     }
00235     //TODO : check if the matrix is positive definite and if the determinant is > 0
00236 }
00237 */
00238
00247 void market::reload_ER(int person, char person_type)
00248 {
00249     string s;
00250
00251     //cout << "<BR>PERSON:" << person;
00252     if (person != 0 )
00253     {
00254         s = "DELETE FROM " + oConfig.tbl_prefix + "_ER_person WHERE person = " + to_string(
person) + " AND person_type = '" + string(1, person_type) + "'";
00255         try
00256         {
00257             db.res = db.stmt->executeQuery(s);
00258         }
00259         catch (sql::SQLException &e)
00260         {
00261             #ifdef DEBUG
00262                 cout << endl;
00263                 cout << endl << "SQL STATEMENT: " << s << endl;
00264                 cout << "# ERR: SQLException in " << __FILE__;
00265                 cout << "(" << __FUNCTION__ << ") on line " << __LINE__ << endl;
00266                 cout << "# ERR: " << e.what();
00267                 cout << " (MySQL error code: " << e.getErrorCode();
00268                 cout << ", SQLState: " << e.getSQLState() << ")" << endl;
00269             #endif
00270         }
00271     }
00272 }
00273
00281 void market::reload_covar(int person, char what2do, char person_type)
00282 {
00283     std::map<int, std::map<int, float> > default_covar;
00284     int ac_i, ac_j;
00285     //1. load the standard covariances into covar[i][j]
00286     load_covar(0);
00287
00288     //2. save the devalut values in default_covar
00289     for (ac_i = 1; ac_i <= NBR_ASSET_CLASSES; ac_i++)

```

```

00290 {
00291     for (ac_j = 1; ac_j <= NBR_ASSET_CLASSES; ac_j++) default_covar[ac_i][ac_j] =
covar[ac_i][ac_j];
00292 }
00293
00294 //3. load the user covar
00295 load_covar(person, person_type);
00296
00297 //4. change the user-volatilities and or covar to default
00298 for (ac_i = 1; ac_i <= NBR_ASSET_CLASSES; ac_i++)
00299 {
00300     for (ac_j = 1; ac_j <= NBR_ASSET_CLASSES; ac_j++)
00301     {
00302         switch (what2do)
00303         {
00304             case 'b' :
00305                 covar[ac_i][ac_j] = default_covar[ac_i][ac_j];
00306                 break;
00307             case 'v' :
00308                 covar[ac_i][ac_j] = covar[ac_i][ac_j] / sqrt (covar[ac_i][ac_i] *
covar[ac_j][ac_j]) * sqrt (default_covar[ac_i][ac_i] * default_covar[ac_j][ac_j]);
00309                 break;
00310             case 'c' :
00311                 if (ac_i != ac_j) covar[ac_i][ac_j] = default_covar[ac_i][ac_j] / sqrt (default_covar[ac_i][ac_i]
] * default_covar[ac_j][ac_j]) * sqrt (covar[ac_i][ac_i] * covar[ac_j][ac_j]);
00312                 break;
00313                 default :
00314                     {}
00315                 }
00316             }
00317         }
00318     }
00319 //5. save the user covar matrix
00320 save_covar(person, person_type);
00321
00322 }

```

8.41 normdist.h File Reference

Macros

- #define [LOW](#) 0.02425
- #define [HIGH](#) 0.97575

Functions

- template<class type >
type [norminv](#) (type p)
- template<class type >
type [phi](#) (type x)
- float [fast_erfinv](#) (float x)
- double [erfinv](#) (double x)

Variables

- static const double [a](#) []
- static const double [b](#) []
- static const double [c](#) []
- static const double [d](#) []

8.41.1 Macro Definition Documentation

8.41.1.1 #define HIGH 0.97575

Definition at line 72 of file [normdist.h](#).

8.41.1.2 #define LOW 0.02425

norminv(x)

Lower tail quantile for standard normal distribution function.

This function returns an approximation of the inverse cumulative standard normal distribution function. I.e., given P , it returns an approximation to the X satisfying $P = \Pr\{Z \leq X\}$ where Z is a random variable from the standard normal distribution.

The algorithm uses a minimax approximation by rational functions and the result has a relative error whose absolute value is less than $1.15e-9$.

Author: Peter John Acklam Time-stamp: 2002-06-09 18:45:44 +0200 E-mail: jacklam@math.uio.no WWW URL: <http://www.math.uio.no/~jacklam>

C implementation adapted from Peter's Perl version by Philippe De Brouwer and re-written as a function template (in stead over function overloading as this results in more compact code)

note:

- requires
 - include <math.h>
 - include <errno.h>

Definition at line 71 of file [normdist.h](#).

8.41.2 Function Documentation**8.41.2.1 double erfinv (double x)**

erfinv

calculates the inverse error function. Uses fast_erfinv and performs a two steps of Newton-Raphson correction to achieve full accuracy.

Definition at line 182 of file [normdist.h](#).

8.41.2.2 float fast_erfinv (float x) [inline]

fast_erfinv

calculates the inverse error function (correct up to 6 digits)

Definition at line 154 of file [normdist.h](#).

8.41.2.3 template<class type > type norminv (type p)

Definition at line 73 of file [normdist.h](#).

8.41.2.4 template<class type > type phi (type x)

Phi(x)

calculates the inverse cumulative normal distribution note: - requires <cmath>

- in stead of function overloading we use a template (shorter code and only one version)

Definition at line 123 of file [normdist.h](#).

8.41.3 Variable Documentation

8.41.3.1 const double a[] [static]

Initial value:

```
=
{
    -3.969683028665376e+01,
    2.2094609884245205e+02,
    -2.759285104469687e+02,
    1.383577518672690e+02,
    -3.066479806614716e+01,
    2.506628277459239e+00
}
```

Coefficients in rational approximations. needed for [norminv\(\)](#)

Definition at line 5 of file [normdist.h](#).

8.41.3.2 const double b[] [static]

Initial value:

```
=
{
    -5.447609879822406e+01,
    1.615858368580409e+02,
    -1.556989798598866e+02,
    6.680131188771972e+01,
    -1.328068155288572e+01
}
```

Definition at line 15 of file [normdist.h](#).

8.41.3.3 const double c[] [static]

Initial value:

```
=
{
    -7.784894002430293e-03,
    -3.223964580411365e-01,
    -2.400758277161838e+00,
    -2.549732539343734e+00,
    4.374664141464968e+00,
    2.938163982698783e+00
}
```

Definition at line 24 of file [normdist.h](#).

8.41.3.4 const double d[] [static]

Initial value:

```
=
{
    7.784695709041462e-03,
    3.224671290700398e-01,
    2.445134137142996e+00,
    3.754408661907416e+00
}
```

Definition at line 34 of file [normdist.h](#).

8.42 normdist.h

```

00001
00005 static const double a[] =
00006 {
00007     -3.969683028665376e+01,
00008     2.209460984245205e+02,
00009     -2.759285104469687e+02,
00010     1.383577518672690e+02,
00011     -3.066479806614716e+01,
00012     2.506628277459239e+00
00013 };
00014
00015 static const double b[] =
00016 {
00017     -5.447609879822406e+01,
00018     1.615858368580409e+02,
00019     -1.556989798598866e+02,
00020     6.680131188771972e+01,
00021     -1.328068155288572e+01
00022 };
00023
00024 static const double c[] =
00025 {
00026     -7.784894002430293e-03,
00027     -3.224671290700398e-01,
00028     -2.400758277161838e+00,
00029     -2.549732539343734e+00,
00030     4.374664141464968e+00,
00031     2.938163982698783e+00
00032 };
00033
00034 static const double d[] =
00035 {
00036     7.784695709041462e-03,
00037     3.224671290700398e-01,
00038     2.445134137142996e+00,
00039     3.754408661907416e+00
00040 };
00041
00071 #define LOW 0.02425
00072 #define HIGH 0.97575
00073 template <class type> type norminv(type p)
00074 {
00075     type q, r;
00076     errno = 0;
00077     if (p < 0 || p > 1)
00078     {
00079         errno = EDOM;
00080         return 0.0;
00081     }
00082     else if (p == 0)
00083     {
00084         errno = ERANGE;
00085         return -HUGE_VAL /* minus "infinity" */;
00086     }
00087     else if (p == 1)
00088     {
00089         errno = ERANGE;
00090         return HUGE_VAL /* "infinity" */;
00091     }
00092     else if (p < LOW)
00093     {
00094         /* Rational approximation for lower region */
00095         q = sqrt(-2*log(p));
00096         return (((c[0]*q+c[1])*q+c[2])*q+c[3])*q+c[4])*q+c[5] /
00097             (((d[0]*q+d[1])*q+d[2])*q+d[3])*q+1;
00098     }
00099     else if (p > HIGH)
00100     {
00101         /* Rational approximation for upper region */
00102         q = sqrt(-2*log(1-p));
00103         return -((((c[0]*q+c[1])*q+c[2])*q+c[3])*q+c[4])*q+c[5] /
00104             (((d[0]*q+d[1])*q+d[2])*q+d[3])*q+1;
00105     }
00106     else
00107     {
00108         /* Rational approximation for central region */
00109         q = p - 0.5;
00110         r = q*q;
00111         return (((a[0]*r+a[1])*r+a[2])*r+a[3])*r+a[4])*r+a[5])*q /
00112             (((b[0]*r+b[1])*r+b[2])*r+b[3])*r+b[4])*r+1;
00113     }
00114 }
00115
00123 template <class type> type phi(type x)

```

```

00124 {
00125     // constants
00126     type a1 = 0.254829592;
00127     type a2 = -0.284496736;
00128     type a3 = 1.421413741;
00129     type a4 = -1.453152027;
00130     type a5 = 1.061405429;
00131     type p = 0.3275911;
00132
00133     // Save the sign of x
00134     int sign = 1;
00135     if (x < 0)
00136         sign = -1;
00137     x = fabs(x)/sqrt(2.0);
00138
00139     // A&S formula 7.1.26
00140     type t = 1.0/(1.0 + p*x);
00141     type y = 1.0 - (((a5*t + a4)*t) + a3)*t + a2)*t + a1)*t*exp(-x*x);
00142
00143     return 0.5*(1.0 + sign*y);
00144 }
00145
00146
00154 inline float fast_erfinv(float x)
00155 {
00156     float tmp;
00157     int neg;
00158
00159     if((neg=(x < 0.0))) x = -x;
00160     if(x <= 0.7)
00161     {
00162         tmp = x*x;
00163         x *= (((-0.140543331*tmp+0.914624893)*tmp-1.645349621)*tmp+0.886226899)/(((0.012229801*tmp-0.3290975
15)*tmp+1.442710462)*tmp-2.118377725)*tmp+1.0);
00164     }
00165     else
00166     {
00167         tmp = sqrt(-log(0.5*(1.0-x)));
00168         x = (((1.641345311*tmp+3.429567803)*tmp-1.624906493)*tmp-1.970840454)/((1.637067800*tmp+3.543889200)*
tmp+1.0);
00169     }
00170     return(neg?-x:x);
00171 }
00172
00173
00182 double erfinv(double x)
00183 {
00184     double res;
00185
00186     res = fast_erfinv(x);
00187     res -= (erf(res)-x)*exp(res*res)*0.886226925452757941;
00188     res -= (erf(res)-x)*exp(res*res)*0.886226925452757941;
00189     return(res);
00190 }

```

8.43 portfolio.class.cpp File Reference

Classes

- class [portfolio](#)

8.44 portfolio.class.cpp

```

00001
00011 class portfolio
00012 {
00013 public:
00014     int portfolio_id;
00015     string description;
00017     float weights[NBR_ASSET_CLASSES];
00018
00019     bool get_portf_from_db(int id);
00020     string get_description();
00021 // float get_risk(float alpha, float (*riskFunction)(float alpha, float (*AssetClass_mu), float
(*AssetClass_varCov)), float (*AssetClass_mu), float (*AssetClass_varCov));
00022     void set_mu(float (*AssetClass_mu));
00023     void set_sigma(float (*AssetClass_varCov));

```

```

00024 float pMu;
00025 float pLogR;
00026 float pSigma;
00027 };
00028
00032 bool portfolio::get_portf_from_db(int id)
00033 {
00034     string s = "";
00035     int k;
00037     try{
00038         sql::Driver *driver;
00039         sql::Connection *con;
00040         sql::Statement *stmt;
00041         sql::ResultSet *res;
00042         // sql::PreparedStatement *pstmt;
00043         // Create a connection
00044         driver = get_driver_instance();
00045         // con = driver->connect("tcp://127.0.0.1:3306","efp","efp.cger");
00046         con = driver->connect(oConfig.db_host,oConfig.db_user,
oConfig.db_pwd);
00047         // con->setSchema("efp");
00048         con->setSchema(db.db);
00049
00050         stmt = con->createStatement();
00051
00052         // -- the portfolio_id:
00053         this->portfolio_id = id;
00054         // -- the description:
00055         s = "SELECT * FROM " + oConfig.tbl_prefix + "_portfolios WHERE portfolio_id = " +
std::to_string(id) + ";";
00056         res = db.stmt->executeQuery(s);
00057         while (res->next()) {this->description = res->getString("description");}
00058         // -- the portfolio composition in terms of asset classes:
00059         for (k=0; k < NBR_ASSET_CLASSES; k++) {this->weights[k] = 0;}
00060         s = "SELECT * FROM " + oConfig.tbl_prefix + "_portfolio_compositions WHERE portfolio = "
+ std::to_string(id) + ";";
00061         res = stmt->executeQuery(s);
00062         while (res->next())
00063             {
00064                 k = res->getInt("asset_class");
00065                 this->weights[k-1] = atof(res->getString("percentage").c_str());
00066             }
00067         /*XXX if (id == 10) {int kk; for (kk=1;kk<=NBR_ASSET_CLASSES;kk++) cout << "<br>weights[" << kk << "]" <<
weights[kk];}
00068         */
00069     } // try
00070     catch (sql::SQLException &e) {
00071         error_message = error_message + t_errMsg["retrieve_portfolio_info"];
00072 #ifndef DEBUG
00073         cout << "# ERR: SQLException in " << __FILE__;
00074         cout << "(" << __FUNCTION__ << ")" on line " << __LINE__ << endl;
00075         cout << "# ERR: " << e.what();
00076         cout << " (MySQL error code: " << e.getErrorCode();
00077         cout << ", SQLState: " << e.getSQLState() << ")" << endl;
00078 #endif
00079     }
00080     return true; //TODO: add checks and return true if it is indeed correctly load and if there are no
definition errors (such as asymmetric covar matrix)
00081 }
00082
00086 string portfolio::get_description()
00087 {
00088     return this->description;
00089 }
00090
00094 void portfolio::set_mu(float *AssetClass_mu)
00095 {
00096     int k;
00097     float m = 0;
00098     for (k = 1; k <= NBR_ASSET_CLASSES; k++)
00099     {
00100         m += *(AssetClass_mu + k) * weights[k-1];
00101         //cout << "<br>[" << k << "]*(AssetClass_mu + k)=" << *(AssetClass_mu + k) << " * weights[k-1]=" << weights[k-1] << "
||Mu*w= " << *(AssetClass_mu + k) * weights[k-1];
00102     }
00103
00104     this->pMu = pow(1 + m,0.08333333) - 1; // 0.0833 = 1/12
00105     //cout << "<br>pMu=" << pMu;
00106     this->pLogR = log(1+ this->pMu);
00107 }
00108
00112 void portfolio::set_sigma(float *AssetClass_varCov)
00113 {
00114     int i, j;
00115     float s = 0;
00116     for (i = 1; i <= NBR_ASSET_CLASSES; i++)
00117     {

```

```

00118     for (j = 1; j <= NBR_ASSET_CLASSES; j++)
00119     {
00120         s+= weights[i-1] * (*(AssetClass_varCov + i * NBR_ASSET_CLASSES + j)) *
weights[j-1];
00121     }
00122 }
00123 if (s >= 0)
00124 {
00125     this->pSigma = (float) sqrt(s); // / sqrt(12);
00126 }
00127 else
00128 {
00129     this->pSigma = 0.0;
00130     error_message = error_message + "Error nbr Erno2332, invalid portfolio
parameters.";
00131 }
00132 // cout << "<BR>pSigma=" << pSigma;
00133 }

```

8.45 simulation.class.cpp File Reference

```
#include <stdlib.h>
```

Classes

- class [simulation](#)

8.46 simulation.class.cpp

```

00001
00010 #include <stdlib.h>
00011
00012 class simulation : public investment_problem
00013 {
00014 public:
00015     void show_simulation ();
00016     simulation(int investor);
00017     void simulate ();
00018     int get_nbrMonths2simulate ();
00019     void plot_market_evol ();
00020
00021 protected:
00022     int nbrMonths;
00023     // vector< vector<float> > market_return; ///< market_return [asset_class] [month]
00024     // float* market_return= new float[NBR_ASSET_CLASSES * 1];
00025     std::map <int, std::map <int, float> > market_return;
00026
00027 private:
00028     void set_nbrMonths2simulate_from_env ();
00029     void simulate_market ();
00030     void simulate_portfolios ();
00031
00032 };
00033
00037 simulation::simulation(int investor)
00038 : investment_problem(investor)
00039 {
00040     set_nbrMonths2simulate_from_env ();
00041 }
00042
00043
00049 void simulation::simulate ()
00050 {
00051     solve ();
00052     simulate_market ();
00053     simulate_portfolios ();
00054 }
00055
00056
00062 void simulation::simulate_market ()
00063 {
00064     int ac, m;
00065     float mu, sigma;
00066     /* float* new_hist = NULL;

```

```

00067     new_hist = (float*) realloc(market_return, (NBR_ASSET_CLASSES + 1) * (nbrMonths + 1) * sizeof(float));
00068     if (new_hist != NULL)
00069     {
00070         market_return = new_hist;
00071     }
00072     else
00073     {
00074         free(market_return);
00075 #ifdef DEBUG
00076     debug_info = debug_info + "<br/>realloc() failed in simulate_market()";
00077 #endif
00078     }
00079 */
00080     srand (time(NULL)); //seed the random number generator
00081
00082     for (ac = 1; ac <= NBR_ASSET_CLASSES; ac++)
00083     {
00084 //     market_return[INDEX(ac,0)] = 0;
00085 //     market_return.push_back(vector<float>());
00086     mu = pow(1 + assetClass_mu[ac], 0.0833333) - 1;
00087     sigma = sqrt(assetClass_covar[INDEX(ac,ac)]);
00088     for (m = 1; m <= nbrMonths; m++)
00089     {
00090         //market_return[ac].push_back((float)norminv((float)(rand() / RAND_MAX)) * sigma + mu);
00091         //mm market_return[INDEX(ac, m)] = (float)(norminv((float)rand() / (float)RAND_MAX)) * sigma + mu);
00092         market_return[ac][m] = (float)(norminv((float)rand() / (float)RAND_MAX)) * sigma
+ mu);
00093         //cout << "<br> market_return[INDEX("<<ac<<","<<m<<")]=" << market_return[ac][m] ;
00094     }
00095 }
00096 }
00097
00103 void simulation::simulate_portfolios()
00104 {
00105     int g, m, ac;
00106     float V, x, curr_age = age(0);
00107     string comma;
00108     for (g = 1; g <= get_nbr_goals(); g++)
00109     {
00110         V = goalZ[g].saving_plan[0] - goalZ[g].means_goal[0];
00111         goalZ[g].simulation_string = "[" + to_string(curr_age) + "," + to_string(V) + "];";
00112
00113
00114         for (m = 1; m <= min((goal_type_i2s[goalZ[g].goal_type] == "rainy day savings") ?
months2simulate() : goalZ[g].realization_monthNbr,
nbrMonths); m++)
00115         {
00116             //R = w . mu
00117             for (ac = 1; ac <= NBR_ASSET_CLASSES; ac++)
00118             {
00119                 //mm V *= exp(market_return[INDEX(ac,m)] * goalZ[g].benchmark[ac]);
00120                 V *= exp(market_return[ac][m] * goalZ[g].benchmark[ac]);
00121                 //cout << "<br>g=" <<g<<"," m="<<m<<"," ac="<<ac<<":V="<<V <<" | | w="<< goalZ[g].benchmark[ac] << "
| market_return[INDEX(ac,m)]=" << market_return[ac][m];
00122             }
00123             V += goalZ[g].saving_plan[m] - goalZ[g].means_goal[m];
00124 //     cout << "<br>saving_plan[goal" << g << "]" << m << "]"=" <<
to_string(goalZ[g].saving_plan[m])<<"-->V="<<V;
00125             x = curr_age + (float)m / 12;
00126             goalZ[g].simulation_string = goalZ[g].simulation_string + "," + to_string(x) + "," +
to_string(V) + "];";
00127             goalZ[g].simulation_endvalue = V;
00128         }
00129     }
00130 }
00131
00132
00138 void simulation::plot_market_evol()
00139 {
00140     int m, ac;
00141     float x;
00142     float curr_age = age(0);
00143     std::map<int, std::string> evol_str;
00144     vector < float > evol_value (NBR_ASSET_CLASSES+1);
00145     string comma;
00146
00147     for (ac = 1; ac <= NBR_ASSET_CLASSES; ac++)
00148     {
00149         evol_str[ac] = "";
00150         evol_value.at(ac) = 100;
00151     }
00152
00153
00154
00156     if (oConfig.simul_show_table)
00157     {
00158         cout << "<table border=\\"1\"><tr><th>Asset Class</th><th>Expected Return</th><th>Estimated
Volatility</th></tr>";

```

```

00159     for (ac = 1; ac <= NBR_ASSET_CLASSES; ac++)
00160     {
00161         cout << "<tr><td>";
00162         cout << assetClass_name[ac] << "</td><td>" << round(
assetClass_mu[ac] * 100) << "%</td><td>" << round(sqrt(12 *
assetClass_covar[INDEX(ac,ac)]) * 100) << "%</td>";
00163         cout << "</tr>";
00164     }
00165     cout << "</table>" << endl;
00166 }
00168 for (m = 1; m <= nbrMonths + 1; m++)
00169 {
00170     comma = (m == 1)? "" : ", ";
00171     x = curr_age + (float)m / 12;
00172     for (ac= 1; ac <= NBR_ASSET_CLASSES; ac++)
00173     {
00174 // cout << " market_return[INDEX("<ac<<","<m<<")]=" << market_return[INDEX(ac,m)] << " | ev=" <<
evol_value[ac] << "<BR>";
00175         evol_str[ac] = evol_str[ac] + comma + "[" + to_string(x) + ", " + to_string(evol_value[ac]) + "];";
00176 //mm         evol_value[ac] = evol_value[ac] * exp(market_return[INDEX(ac, m)]);
00177         evol_value[ac] = evol_value[ac] * exp(market_return[ac][m]);
00178     }
00179 }
00181 cout << "<script class=\"code\" type=\"text/javascript\">$(document).ready(function(){ ";
00182 for (ac= 1; ac <= NBR_ASSET_CLASSES; ac++)
00183 {
00184     cout << "var l" << ac << " = [" + evol_str[ac] + "]; ";
00185 }
00186 cout << "var plotME = $.jqplot('chartMEvol', [";
00187 for (ac = 1; ac <= NBR_ASSET_CLASSES; ac++)
00188 {
00189     comma = (ac == 1)? "" : ", ";
00190     cout << comma << " l" << ac;
00191 }
00192 cout << "], {";
00193 cout << "seriesColors:" << hh.assetClassColors << ",";
00194 cout << "seriesDefaults: {rendererOptions: { smooth: false }},";
00195 cout << "legend: {show: true, location: 'e', placement: 'outside', marginRight: \"600px\", ";
00196 cout << "labels:[";
00197     for (ac= 1; ac <= NBR_ASSET_CLASSES; ac++)
00198     {
00199         comma = (ac == 1)? "" : ", ";
00200         cout << comma << "' " << assetClass_name[ac] << "' ";
00201     }
00202 cout << "],";
00203 cout << "series:[";
00204     for (ac= 1; ac <= NBR_ASSET_CLASSES; ac++)
00205     {
00206         comma = (ac == 1)? "" : ", ";
00207         cout << comma << "{showMarker: false}";
00208     }
00209     cout << "],";
00210
00211     cout << "axesDefaults: {labelRenderer: $.jqplot.CanvasAxisLabelRenderer},";
00212     cout << "axes: {xaxis: {label: \"age\" }, yaxis: {label: \"market evolution\"}}"; // , pad: 0
00213     cout << " } ); }); </script>";
00214 }
00215
00216
00223 void simulation::set_nbrMonths2simulate_from_env()
00224 {
00225     int n;
00226     int defaultValue = 12;
00228     vector<FormEntry>::iterator nYears = cgi.getElement("nYears");
00229     if(nYears != cgi.getElements().end())
00230     {
00231         n = atoi(cgi("nYears").c_str()) * 12;
00232     }
00233     else
00234     {
00235         n = 0;
00236     }
00237     vector<FormEntry>::iterator nMonths = cgi.getElement("nMonths");
00238     if(nMonths != cgi.getElements().end())
00239     {
00240         n = n + atoi(cgi("nMonths").c_str());
00241     }
00242     if ((n >= MIN_MNTHS_2_SIMULATE) && (n <=
MAX_MNTHS_2_SIMULATE))
00243     {
00244         nbrMonths = n;
00245     }
00246     else
00247     {
00248         nbrMonths = defaultValue;
00249         error_message = error_message + "<li>Please enter a number of months between
" + to_string(MIN_MNTHS_2_SIMULATE) + " and " + to_string(

```

```

MAX_MNTHS_2_SIMULATE) + " (= " + to_string((int)(
MAX_MNTHS_2_SIMULATE/12)) + " years)</li>";
00250     }
00251 }
00252
00253
00259 int simulation::get_nbrMonths2simulate()
00260 {
00261     return nbrMonths;
00262 }

```

8.47 t_disclaimer_Eng_UK.h File Reference

Variables

- const string `t_disclaimer`

8.47.1 Variable Documentation

8.47.1.1 const string t_disclaimer

Initial value:

```

= "<br> \
<h1>Disclaimers and Policies</h1> \
<p>The disclaimer that appears on the bottom of each screen is to be considered part of the general terms
and conditions.</p> \
<p>In summary, we gather only the information that is necessary to recognize you in the tool (so that you
can find the information that you provide back later. We ask also a little more information that login and
password) and the necessary information to provide some advice on the investment portoflio that you want to
talk about. The reason for that is that this tool will help you to define the best fitted investment portfolio
for you. That is a very personal thing, and so we need a few personal things. If you provide wrong
information, then the results will be wrong (well that does not exclude that this might be interesting to test of
course).</p> \
We will not sell this data or use it otherwise than use it in the tool to show you its potential and by
sending relevant messages for the investment case that you have put into the system. \
<p>This disclaimer is up subject to changes without prior notice.</p> \
<p></p> \
\
<a name=\"use\"><h2>Terms of Use</h2></a> \
<p>You declare to use this software at your own risk. This means that the author(s) cannot be held
responsible for the investment advice, time waste, or any other discomfort or otherwise disadvantage resulting from
using this website. Also the authors cannot be held responsible for eventual losses or discomfort when the
website is not available.</p> \
<p>The reason is that we are not financial advisors as we are not regulated by any financial authority, we
are only providing a platform to check out the latest technology, we are not selling investment advice. We
provide advice about investment advice!</p> \
\
<a name=\"privacy\"><h2>Privacy Policy</h2></a> \
<p>We will never sell your email, nor spam you. We might send you a newsletter from time to time and all
data that you provide in the system will be kept there till you delete your account or overwrite it. We do not
keep any other data than what is visible and changeable to you.</p> \
<p>You are free to take additional measures to protect your privacy usch as use fake email, name, first
name and even make up the numbers (such as putting all in thousands (so putting 10 in stead of 10'000) in order
to keep your identity secret. This will not affect the functioning of te tool nor the advice that it
gives.</p> \
<p>Of course you declare not to use the email of anyone else!</p> \
\
<a name=\"cookie\"><h2>Cookie Policy</h2></a> \
<p>This site uses cookies for the management of sessions. In other words the site will put a cookie on your
computer to remember that you are properly logged in. We do not track your behaviour. Later we might use
cookies for your comfort (such as remembering your country or language settings.</p> \
<p><a class=\"button\" href=\"javascript:history.back()\">Confirm Agreement</a> or <a class=\"button\"
href=\"http://en.wikipedia.org/wiki/Maslowian_portfolio_theory\">get me out of here</a></p>"

```

Definition at line 1 of file `t_disclaimer_Eng_UK.h`.

8.48 t_disclaimer_Eng_UK.h

```

00001 const string t_disclaimer = "<br> \
00002 <h1>Disclaimers and Policies</h1> \

```

```

00003 <p>The disclaimer that appears on the bottom of each screen is to be considered part of the general terms
and conditions.</p> \
00004 <p>In summary, we gather only the information that is necessary to recognize you in the tool (so that you
can find the information that you provide back later. We ask also a little more information that login and
password) and the necessary information to provide some advice on the investment portoflio that you want to
talk about. The reason for that is that this tool will help you to define the best fitted investment portfolio
for you. That is a very personal thing, and so we need a few personal things. If you provide wrong
information, then the results will be wrong (well that does not exclude that this might be interesting to test of
course).</p> \
00005 We will not sell this data or use it otherwise than use it in the tool to show you its potential and by
sending relevant messages for the investment case that you have put into the system. \
00006 <p>This disclaimer is up subject to changes without prior notice.</p> \
00007 <p></p> \
00008 \
00009 <a name="use"><h2>Terms of Use</h2></a> \
00010 <p>You declare to use this software at your own risk. This means that the author(s) cannot be held
responsible for the investment advice, time waste, or any other discomfort or otherwise disadvantage resulting from
using this website. Also the authors cannot be held responsible for eventual losses or discomfort when the
website is not available.</p> \
00011 <p>The reason is that we are not financial advisors as we are not regulated by any financial authority, we
are only providing a platform to check out the latest technology, we are not selling investment advice. We
provide advice about investment advice!</p> \
00012 \
00013 <a name="privacy"><h2>Privacy Policy</h2></a> \
00014 <p>We will never sell your email, nor spam you. We might send you a newsletter from time to time and all
data that you provide in the system will be kept there till you delete your account or overwrite it. We do not
keep any other data than what is visible and changeable to you.</p> \
00015 <p>You are free to take additional measures to protect your privacy usch as use fake email, name, first
name and even make up the numbers (such as putting all in thousands (so putting 10 in stead of 10'000) in order
to keep your identity secret. This will not affect the functioning of te tool nor the advice that it
gives.</p> \
00016 <p>Of course you declare not to use the email of anyone else!</p> \
00017 \
00018 <a name="cookie"><h2>Cookie Policy</h2></a> \
00019 <p>This site uses cookies for the management of sessions. In other words the site will put a cookie on your
computer to remember that you are properly logged in. We do not track your behaviour. Later we might use
cookies for your comfort (such as remembering your country or language settings.</p> \
00020 <p><a class="button" href="javascript:history.back()">Confirm Agreement</a> or <a class="button"
href="http://en.wikipedia.org/wiki/Maslowian_portfolio_theory">get me out of here</a></p>;

```

8.49 t_personalize_Eng_UK.h File Reference

Functions

- void [init_personalize](#) ()

Variables

- const string [t_personalize1](#) = "<p>For Accurate advice, fill out this form very carefully.</p><p>NOT-E: make sure to save changes in each section before moving on to another section! Otherwise those changes will be lost!</p></p>"
- const string [t_personalize_concepts](#)
- const string [t_personalize_scoring](#)
- const string [t_Knowledge](#) = "knowledge"
- const string [t_Experience](#) = "Experience"
- const string [t_Desirability](#) = "Desirability"
- const string [t_Max_Exp](#) = "Maximum Exposure"
- const string [t_info_curr](#) = "this is your base currency (all amounts in other currencies will be converted to this one (NOT ACTIVE YET!!)"
- const string [t_info_bdate](#) = "Your birthdate is necessary to allow you to enter a goal by only providing your age at which you want to attain this goal."
- const string [t_info_simtill](#) = "this is the age till which simulations will run. Be realistic here, sure one will want to become 200 years old, but it does not make sense to simulate too long in the future."
- const string [t_reload](#) = "Reload Initial "
- unordered_map< string, unordered_map< string, string > > [t_personalize](#)

8.49.1 Function Documentation

8.49.1.1 void init_personalize ()

Definition at line 36 of file [t_personalize_Eng_UK.h](#).

8.49.2 Variable Documentation

8.49.2.1 const string t_Desirability = "Desirability"

Definition at line 23 of file [t_personalize_Eng_UK.h](#).

8.49.2.2 const string t_Experience = "Experience"

Definition at line 22 of file [t_personalize_Eng_UK.h](#).

8.49.2.3 const string t_info_bdate = "Your birthdate is necessary to allow you to enter a goal by only providing your age at which you want to attain this goal."

Definition at line 27 of file [t_personalize_Eng_UK.h](#).

8.49.2.4 const string t_info_curr = "this is your base currency (all amounts in other currencies will be converted to this one (NOT ACTIVE YET!!)"

Definition at line 26 of file [t_personalize_Eng_UK.h](#).

8.49.2.5 const string t_info_simtill = "this is the age till which simulations will run. Be realistic here, sure one will want to become 200 years old, but it does not make sense to simulate too long in the future."

Definition at line 28 of file [t_personalize_Eng_UK.h](#).

8.49.2.6 const string t_Knowledge = "knowledge"

Definition at line 21 of file [t_personalize_Eng_UK.h](#).

8.49.2.7 const string t_Max_Exp = "Maximum Exposure"

Definition at line 24 of file [t_personalize_Eng_UK.h](#).

8.49.2.8 unordered_map<string, unordered_map<string, string> > t_personalize

Definition at line 35 of file [t_personalize_Eng_UK.h](#).

8.49.2.9 const string t_personalize1 = "<p>For Accurate advice, fill out this form very carefully.</p><p>NOTE: make sure to save changes in each section before moving on to another section! Otherwise those changes will be lost!</p></p>"

Definition at line 1 of file [t_personalize_Eng_UK.h](#).

8.49.2.10 const string t_personalize_concepts

Initial value:

```
= "<h2>On Setting the Limits for Each Asset Class</h2> \
<p>Here you can use your past experience, knowledge and desirability to set limits for all asset classes.
  Please bear in mind that \
<ul><li>the sum of the weights should be at least 100% (best make sure that there is significantly more
  freedom),</li> \
<li>Also bear in mind that each limitation is moving your portoflio farther away from being optional: even
  if you don't like an asset class, it can be a very valuable diversification to your portoflio. Our system
  will only select asset classes if if they fit in the portfolio (reduce the risk of not reaching your
  goal).</li></ul> \
<p>You can use the following concepts to allow the system to suggest some limits:</p> \
<ul><li><b>experience:</b> assesses whether you have used these asset class in the past. If you used these
  asset class in your own portfolio, at work in your actual or previous job and especially if you used to
  trade frequently high volumes, you might rate yourself high on this.</li> \
<li><b>knowledge:</b> do you understand what these financial asset class do? Did they appear in your
  education (high school, university). Do you understand how they behave? Score yourself high if you know very well
  what these asset class do, how they behave and how they can be risky.</li> \
<li><b>desirability:</b> regardless of how much you know about this asset class, you might still not like
  them too much. Rate this low if you don't like the asset class</li> \
</ul> \
"
```

Definition at line 2 of file [t_personalize_Eng_UK.h](#).

8.49.2.11 const string t_personalize_scoring

Initial value:

```
= "<p>Use the folowing scale to rate your knowledge, experience and desirability of having an asset class
  in your portfolio</p> \
<ul><li><b>1</b>: none / not at all</li> \
<li><b>2</b>: a little</li> \
<li><b>3</b>: average</li> \
<li><b>4</b>: much</li> \
<li><b>5</b>: all / best / very good</li> \
</ul> \
"
```

Definition at line 12 of file [t_personalize_Eng_UK.h](#).

8.49.2.12 const string t_reload = "Reload Initial "

Definition at line 34 of file [t_personalize_Eng_UK.h](#).

8.50 t_personalize_Eng_UK.h

```
00001 const string t_personalizel = "<p>For Accurate advice, fill out this form very
  carefully.</p><p><b>NOTE</b>: make sure to save changes in each section before moving on to another section! Otherwise
  those changes will be lost!</p></p>";
00002 const string t_personalize_concepts = "<h2>On Setting the Limits for Each Asset
  Class</h2> \
00003 <p>Here you can use your past experience, knowledge and desirability to set limits for all asset classes.
  Please bear in mind that \
00004 <ul><li>the sum of the weights should be at least 100% (best make sure that there is significantly more
  freedom),</li> \
00005 <li>Also bear in mind that each limitation is moving your portoflio farther away from being optional: even
  if you don't like an asset class, it can be a very valuable diversification to your portoflio. Our system
  will only select asset classes if if they fit in the portfolio (reduce the risk of not reaching your
  goal).</li></ul> \
00006 <p>You can use the following concepts to allow the system to suggest some limits:</p> \
00007 <ul><li><b>experience:</b> assesses whether you have used these asset class in the past. If you used these
  asset class in your own portfolio, at work in your actual or previous job and especially if you used to
  trade frequently high volumes, you might rate yourself high on this.</li> \
00008 <li><b>knowledge:</b> do you understand what these financial asset class do? Did they appear in your
  education (high school, university). Do you understand how they behave? Score yourself high if you know very well
  what these asset class do, how they behave and how they can be risky.</li> \
00009 <li><b>desirability:</b> regardless of how much you know about this asset class, you might still not like
  them too much. Rate this low if you don't like the asset class</li> \
```

```

00010 </ul> \
00011 ";
00012 const string t_personalize_scoring = "<p>Use the folowing scale to rate your
      knowledge, experience and desirability of having an asset class in your portfolio</p> \
00013 <ul><li><b>1</b>: none / not at all</li> \
00014 <li><b>2</b>: a little</li> \
00015 <li><b>3</b>: average</li> \
00016 <li><b>4</b>: much</li> \
00017 <li><b>5</b>: all / best / very good</li> \
00018 </ul> \
00019 ";
00020
00021 const string t_Knowledge = "knowledge";
00022 const string t_Experience = "Experience";
00023 const string t_Desirability = "Desirability";
00024 const string t_Max_Exp = "Maximum Exposure";
00025
00026 const string t_info_curr = "this is your base currency (all amounts in other currencies will
      be converted to this one (NOT ACTIVE YET!!)";
00027 const string t_info_bdate = "Your birthdate is necessary to allow you to enter a goal by only
      providing your age at which you want to attain this goal.";
00028 const string t_info_simtill = "this is the age till which simulations will run. Be realistic
      here, sure one will want to become 200 years old, but it does not make sense to simulate too long in the
      future.";
00029
00030 //typedef unordered_map<string, string> stringMAP;
00031 //unordered_map<string, stringMAP> t_personalize;
00032 //std::map< std::string, std::map<std::string, std::string> > t_personalize;
00033
00034 const string t_reload = "Reload Initial ";
00035 unordered_map<string, unordered_map<string, string> > t_personalize;
00036 void init_personalize()
00037 { // these initializations will only work within a function
00038   t_personalize["general"]["title"] = "General Parameters";
00039   t_personalize["general"]["info"] = "";
00040   t_personalize["general"]["content"] = "General Parameters";
00041   t_personalize["prefs"]["title"] = "Rate On a Scale of 1 to 5";
00042   t_personalize["prefs"]["info"] = "<p>Rate your experience, knowledge and how much you
      desire it for each asset class. Use a scale from 1 to 5 (eg. &quot;l&quot; its the choice if you have no
      experience, no knowledge and you do not want it. Five would be the choice if you know a lot about it, used it
      before and have no issues in owning it).</p><p>As you will see in the &quot;Maximum Exposure&quot; column,
      this information is used to set limits on the maximum exposure to each asset class individually.</p>";
00043   t_personalize["prefs"]["content"] = "Preferences";
00044   t_personalize["ER"]["title"] = "Your Expectations on Return";
00045   t_personalize["ER"]["info"] = "<p>Your expectations for the returns to expect (in
      percentage per annum!</p><p><b>NOTE</b>: these expectations are AFTER inflation (so if you expect cash to
      yield 2% and inflation to be 2%, then the expected return of cash is -1% (indeed negative)!</p>";
00046   t_personalize["ER"]["content"] = "Expected Returns";
00047   t_personalize["ER"]["thead"] = "Annual Expected Returns";
00048   t_personalize["vol"]["title"] = "Your Expectations on Volatility";
00049   t_personalize["vol"]["info"] = "<p>Your expectations for the volatilities to expect
      (in percentage per annum)</p><p><b>NOTE</b>: rounding errors might occur, because the software stores
      volatilities and correlations combined as a &quot;covariance matrix&quot;!</p>";
00050   t_personalize["vol"]["content"] = "Expected volatilities";
00051   t_personalize["vol"]["thead"] = "Expected volatilities";
00052   t_personalize["corr"]["title"] = "Your Expectations on Correlations";
00053   t_personalize["corr"]["info"] = "<p>Your expectations for the correlations to expect
      (in percentage and for annual returns)</p><p><b>NOTE</b>: rounding errors might occur, because the software
      stores volatilities and correlations combined as a &quot;covariance matrix&quot;!</p>";
00054   t_personalize["corr"]["content"] = "Expected Correlations";
00055   t_personalize["ERvol"]["title"] = "Your Expectations on Return and Volatility";
00056   t_personalize["ERvol"]["info"] = t_personalize["ER"]["info"] + " " +
      t_personalize["vol"]["info"];
00057   t_personalize["ERvol"]["content"] = "Expected Returns and volatilities";
00058 }

```

8.51 test.js File Reference

Functions

- document [ready](#) (function(){var l0=[6, 11, 10, 13, 11, 7];var l1=[3, 6, 7, 7, 5, 3];var l2=[4, 8, 9, 10, 8, 6];var l3=[9, 13, 14, 16, 17, 19];var l4=[15, 17, 16, 18, 13, 11];var plot1=\$.jqplot("chartOverview1", [l0, l1, l2, l3, l4],{seriesDefaults:{rendererOptions:{smooth:false}}});})

8.51.1 Function Documentation

```
8.51.1.1 document ready ( function(){var l0=[6, 11, 10, 13, 11, 7];var l1=[3, 6, 7, 7, 5, 3];var l2=[4, 8, 9, 10, 8, 6];var
l3=[9, 13, 14, 16, 17, 19];var l4=[15, 17, 16, 18, 13, 11];var plot1=$.jqplot("chartOverview1", [l0, l1, l2, l3,
l4],{seriesDefaults:{rendererOptions:{smooth:false}}}); } )
```

8.52 test.js

```
00001 $(document).ready(function(){
00002   var l0 = [6, 11, 10, 13, 11, 7];
00003   var l1 = [3, 6, 7, 7, 5, 3];
00004   var l2 = [4, 8, 9, 10, 8, 6];
00005   var l3 = [9, 13, 14, 16, 17, 19];
00006   var l4 = [15, 17, 16, 18, 13, 11];
00007   var plot1 = $.jqplot("chartOverview1", [l0, l1, l2, l3, l4], {seriesDefaults: {rendererOptions: { smooth:
false }} } }); });
00008
00009 $(document).ready(function(){
00010   var l0 = [6, 11, 10, 13, 11, 7];
00011   var l1 = [3, 6, 7, 7, 5, 3];
00012   var l2 = [4, 8, 9, 10, 8, 6];
00013   var l3 = [9, 13, 14, 16, 17, 19];
00014   var l4 = [15, 17, 16, 18, 13, 11];
00015
00016   var plot1 = $.jqplot("chart1", [l0, l1, l2, l3, l4], {
00017     title: "Fill between 2 lines",
00018     axesDefaults: {
00019       pad: 1.05
00020     },
00021     // Use the fillBetween option to control fill between two
00022     // lines on a plot.
00023     fillBetween: {
00024       // series1: Required, if missing won't fill.
00025       series1: 1,
00026       // series2: Required, if missing won't fill.
00027       series2: 3,
00028       // color: Optional, defaults to fillColor of series1.
00029       color: "rgba(227, 167, 111, 0.7)",
00030       // baseSeries: Optional. Put fill on a layer below this series
00031       // index. Defaults to 0 (first series). If an index higher than 0 is
00032       // used, fill will hide series below it.
00033       baseSeries: 0,
00034       // fill: Optional, defaults to true. False to turn off fill.
00035       fill: true
00036     },
00037     seriesDefaults: {
00038       rendererOptions: {
00039         // Turn on line smoothing. By default, a constrained cubic spline
00040         // interpolation algorithm is used which will not overshoot or
00041         // undershoot any data points.
00042         smooth: true
00043       }
00044     }
00045   });
00046 });
00047
00048
00049
00050
00051
00052 });
00053
00054
00055
00056
00057
00058
00059
```

8.53 texts_Eng_UK.h File Reference

Variables

- const string `t_author` = "Dr. Philippe De Brouwer"
- const string `t_title` = "Goal-Ranking"
- const string `t_warning` = "<p>This software is provided for research purposes only, it is not intended to be used as a foundation for your personal financial decisions! Please talk to the author or to a financial advisor.</p><p>The goal of this website is to show advisers (such as independent financial planners, bankers, insurance agents, etc.) how real responsible personal financial advice can look like. The author, " + `t_author` + ", invites you to contact him by e-mail to discuss how you can make your

advice more responsible, forge more durable relations with your customers and become a leader in the digital race.</p>"

- const string `t_month` = "month"
- General words.*
- const string `t_year` = "year"
- const string `t_months` = "months"
- const string `t_years` = "years"
- const string `t_and` = "and"
- const string `t_Asset_Class` = "Asset Class"
- const string `t_prev_step` = "prev step"
- const string `t_next_step` = "next step"
- const string `t_reset` = "reset"
- const string `t_save` = "save"
- const string `t_edit` = "edit"
- const string `t_delete` = "delete"
- const string `t_cancel` = "cancel"
- const string `t_clear_form` = "clear form"
- const string `t_collapse` = "collapse"
- const string `t_Login` = "Login"
- const string `t_Register` = "Register"
- const string `t_simulation_duration` = "Simulation Duration"
- const string `t_currency` = "currency"
- const string `t_b_currency` = "base currency"
- const string `t_b_date` = "birth date"
- const string `t_sim_till` = "simulate till age"
- const string `t_breadcrumb` = "You are here: ";"
- unordered_map< string, string > `t_aside`
- unordered_map< string, string > `t_titles`
- unordered_map< string, string > `t_regFrm`
- unordered_map< string, string > `t_errMsg`
- const string `t_show_extra_info` = "Show More Info"
- const string `t_hide_extra_info` = "Hide More Info"
- const string `t_showhide_extra_info` = "Show/Hide Additional Information"
- const string `t_extra_info` = "Extra Information"
- unordered_map< string, string > `t_extraInfo`
- unordered_map< string, string > `t_feedback`
- unordered_map< string, string > `t_feedback_post`
- const string `t_dashboard` = "Dashboard"
- unordered_map< string, string > `t_before_list`
- unordered_map< string, string > `t_after_title`
- unordered_map< int, string > `t_scale125i2s`
- unordered_map< int, string > `t_plotlabels_evol`

8.53.1 Variable Documentation

8.53.1.1 unordered_map<string, string> t_after_title

Initial value:

```
=
{
  {"Home", "<h4>Welcome to real investment advice!</h4><p>Here You find personalized advice that rivals the
    best private bank, but fully automated, because we start from your life-goals and take your goals
    seriously.</p>"}
}
```

`t_after_title`

Definition at line 179 of file `texts_Eng_UK.h`.

8.53.1.2 const string t_and = "and"

Definition at line 15 of file [texts_Eng_UK.h](#).

8.53.1.3 unordered_map<string, string> t_aside

Initial value:

```
=
{
  {"welcome", "Welcome"},
  {"login", "Login"},
  {"register", "Register"},
  {"account", "Account Settings"},
  {"goalInfo", "Try to quantify your important life-goals. It is better to state a goal that is 60% correct
    than stating no goal at all!"},
  {"goalHelpFail", "Please try again. Fill out only the goals that you want to use and use realistic data!"
  },
  {"cash_flowInfo", "What is your savings-capacity? How will that change?"},
  {"feedback", "We hope that you liked this solution and that it was worth your time!<br>Please let us know
    <a href=\"mailto:philippe@de-brouwer.com\">via email</a> and spread the word!<br>Thank you!"},
  {"simulation", "Here we show you what can happen if you would not follow up the portfolio."},
  {"follow_up", "Please make sure that you follow up your goals regularly!<br>Wouldn't it be nice to have a
    webservice sending you this dashboard every month?<br>Please let us know what you did not like via <a href=
    \"mailto:philippe@de-brouwer.com\">via email</a> and tell your friends what you did like!<br>Thank you!"},
  {"disclaimer", "We will never sell your email, nor spam you. We might send you a newsletter from time to
    time and all data that you provide in the system will be kept there till you delete your account or
    overwrite it."}
}
```

Definition at line 40 of file [texts_Eng_UK.h](#).

8.53.1.4 const string t_Asset_Class = "Asset Class"

Definition at line 16 of file [texts_Eng_UK.h](#).

8.53.1.5 const string t_author = "Dr. Philippe De Brouwer"

Definition at line 6 of file [texts_Eng_UK.h](#).

8.53.1.6 const string t_b_currency = "base currency"

Definition at line 32 of file [texts_Eng_UK.h](#).

8.53.1.7 const string t_b_date = "birth date"

Definition at line 33 of file [texts_Eng_UK.h](#).

8.53.1.8 unordered_map<string, string> t_before_list

Initial value:

```
=
{
  {"goal", "Need a help? Use the <a class=\"button\" href=\"" + oConfig.
    soft_url + "?a=gh\">support screen to add goals</a>."}
}
```

t_before_list

text to be printed before a list

Definition at line 171 of file [texts_Eng_UK.h](#).

8.53.1.9 `const string t_breadcrumb = "You are here: ";`

Definition at line 35 of file [texts_Eng_UK.h](#).

8.53.1.10 `const string t_cancel = "cancel"`

Definition at line 23 of file [texts_Eng_UK.h](#).

8.53.1.11 `const string t_clear_form = "clear form"`

Definition at line 24 of file [texts_Eng_UK.h](#).

8.53.1.12 `const string t_collapse = "collapse"`

Definition at line 25 of file [texts_Eng_UK.h](#).

8.53.1.13 `const string t_currency = "currency"`

Definition at line 31 of file [texts_Eng_UK.h](#).

8.53.1.14 `const string t_dashboard = "Dashboard"`

Definition at line 163 of file [texts_Eng_UK.h](#).

8.53.1.15 `const string t_delete = "delete"`

Definition at line 22 of file [texts_Eng_UK.h](#).

8.53.1.16 `const string t_edit = "edit"`

Definition at line 21 of file [texts_Eng_UK.h](#).

8.53.1.17 `unordered_map<string, string> t_errMsg`

Initial value:

```
=
{
  {"loginFirst",          "<li>You need to login before accessing these sections!</li>"},
  {"errorSavingUser",    "<li>Undefined error saving this user information.</li>"},
  {"userNameTaken1",     "<li>Sorry, the user name &quot;"},
  {"userNameTaken2",     "&quot; is already taken!</li>"},
  {"pwsMatch",           "<li>the passwords provided do not match</li>"},
  {"pwdUserCombination", "<li>Sorry, I could not retrieve this combination of userID and password
    :-(<br/>Please check the if CapsLock is not on and retype the password.</li>"},
  {"retrievingInvestor", "<li>Error occurred retrieving this user from the database, contact
    administrator</li>"},
  {"no_goals",           "<li>Please state at least ONE goal first!</li>"},
  {"no_assets",          "<li>Please define assets and future savings capacity!</li>"}
}
```

Definition at line 86 of file [texts_Eng_UK.h](#).

8.53.1.18 `const string t_extra_info = "Extra Information"`

Definition at line 107 of file [texts_Eng_UK.h](#).

8.53.1.19 unordered_map<string, string> t_extrInfo

Initial value:

```

=
{
  {"welcome", "<p><h4>Welcome</h4>This simple process consists of three easy steps:</p><ol><li>Register or Login</li><li>STEP 1: Make an inventory of your assets and main cash-flows (income and expenses)</li><li>STEP 2: Set Your personal financial goals</li><li>STEP 3: See your personalized Feedback and eventual step 1 and/or 2.</li></ol>The basic idea is to provide real investor centric investment advice by starting from your personal goals and savings. At the end of this excercise, you should have feedback about which goals are realistic, what savings-plan you need and eventually what composition each portfolio should have (benchmark).<h4>When Registering</h4><p>Note that this sign-up is only to be able to recognize you in the tool (so that you can find the information that you provide back later. We ask also a little more information that login and password. The reason for that is that this tool will help you to define the best fitted investment portfolio for <strong>you</strong>. That is a very personal thing, and so we need a few personal things. If you provide wrong information, then the results will be wrong.)</p><p>However, in order to guarantee your anonymity you can provide for example an alternate name and multiply all your goals and investments with a certain multiplier that only you know (e.g times 5).</p>"},

  {"goal", "<h4>Investment Goals</h4><p>Here you need to inform the system about what you would like to obtain in your life. This is the most important part of this software!</p><p><b>Important:</b> This software is only a demo version, we appologize for the inconvenience, but please bear in min that:</p><ul><li>for the goal type <b>income from/to</b></li><li>the from-date, till-date and frequency are key</li><li>for the goal-type <b>rainy day savings</b></li><li>no date information is necessary, and</li><li>the goal-type <b>unallocated</b></li></ul><p>is actually not to be used. The software will use it to allocate all assets that are not any other goal. So you would need only one <b>unallocated</b> goal.</li></ul><p>All the information that we ask for is needed in order to select for you the right investment portfolio and savings plan. There are two important remarks to take into account</p><ol><li><strong>First: there is a thin line between <b>Cash Flows</b> and plans.</strong> If you need to buy for example a new car next year (and that expense is certain, necessary and fixed ... then it is possible to add it as well as a cash flow and as a goal. However in this case (short term, certain and no degree of flexibility) it is advisable to first add it as a cash flow and hope it all works out. Only when you decide that the car can eventually cost less in order to finance other goals, then it should be considered as a goal.</li><li><strong>Second: in this early demo version, we do not guide you too much about which goal should be the most important</strong> (in other words: sorry, but we do not use the need level yet). So the order that you provide in <b>ranking</b> is very important! Please put a number (eg. one) for your most important goal and put higher numbers for the others (yes in this early demo it is not important what numbers you use)</li></ol><p>Remember the old addagio that a goal has to be SMART (Specific, Measurable, Attainable, Relevant and Time Bound).</p>"},

  {"cash_flow", "<h4>Cash flows</h4>Cash flows are your additional savings. It can simply be the average amount that you can save monthly. However if other amounts might be useful to include. But, be careful which point of view you take!. For example if you save now 500,- per month and your kids are about to leave college in 3 years (which will give you an additional saving capacity of 750,- per month. Well in that case you could book one cash flow of 500,- starting today and ending when you retire, and another one of 750,- starting within three years and also till you retire.<br>If on the other hand your kids will go to college in 5 years, then it is wise to foresee <i>negative</i> cash flow starting in 5 years and running for 4 years (eg. -660,- (mind the minus sign!)).<br><br>HINT:</b> in order to make a cash flow start right away: leave the <b>From Date</b> empty; in order to make a cash flow available till your retirement: leave the <b>Till Date</b> empty."},

  {"register", "<h4>Thanks for signing up and testing our tool!</h4><p>Note that this sign-up is only to be able to recognize you in the tool (so that you can find the information that you provide back later. We ask also a little more information that login and password. The reason for that is that this tool will help you to define the best fitted investment portfolio for <strong>you</strong>. That is a very personal thing, and so we need a few personal things. If you provide wrong information, then the results will be wrong (well that does not exclude that this might be interesting to test of course).<br>So, indeed, signup up here is just to use the tool, this means that we can of course show you its potential by sending relevant messages for the investment case that you have put into the system. We will not use your email to send <b>marketing mails</b>; that's why we have our newsletter! You can sign-up for it soon ;-)</p>"},

  {"account", "<h4>Keep your account information up to date!</h4><p>Note that this sign-up is only to be able to recognize you in the tool (so that you can find the information that you provide back later. We ask also a little more information that login and password. The reason for that is that this tool will help you to define the best fitted investment portfolio for <strong>you</strong>. That is a very personal thing, and so we need a few personal things. If you provide wrong information, then the results will be wrong (well that does not exclude that this might be interesting to test of course).<br>So, indeed, signup up here is just to use the tool, this means that we can of course show you its potential by sending relevant messages for the investment case that you have put into the system. We will not use your email to send <b>marketing mails</b>; that's why we have our newsletter! You can sign-up for it soon ;-)</p>"},

  {"login", "<h4>Welcome Back!</h4>Use this screen only if you have registered before and you already have a login and password. Otherwise, please register first!<br>Also remember that the fact that you have a login to test our tools is no guarantee that you will get any information from us. To be kept up to date, please sign up for our newsletter [not available yet]"},

  {"personalize", t_personalize_concepts + t_personalize_scoring},

  {"feedback", "So, here you are! But what now?<br>What do you think? Does it look good? Can you reach the goals that you have put yourself? Maybe you have still money left, then you can add more goals and/or increase the amounts for certain goals.<br>But maybe you notice that you have more goals than money. The positive side of this is that you have an interesting life and that you have plenty of plans. Now you can do two things<ol><li>decrease the amounts of the goals</li><li>accept more shortfall on the goals</li><li>or simply leave the less important goals out</li></ol>. Note also that this feedback is shaped by your personal

```

```

        selections for and preferences (you might have excluded certain asset classes in the [Preferences] screen for
        example). <p>In any case: thank you very much for your time, we hope that it was an interesting experience. Please
        sign up for our newsletter and we promise to keep you informed."},

{"asset","Describe your assets here. They will be used as an initial amount that can be used to invest.
  We will not use the existing portfolio composition as a starting point, but largely describing your assets
  here will help later to inform you about what to sell and what to buy. Please enter here all assets that you
  consider to be available for sale. If for example you intend to keep your house as long as you live and then
  pass it on to your children, then you should not mention it here. Of course, it is up to you to decide this!
  Maybe leave it out and see if you can reach your goals, but maybe you could consider a smaller place to
  live but reach those important goals? Up to you, it's your life!"),

{"simulation","<h4>Simulaton</h4>In this screen we show you what might possibly happen to your
  investments. Of course, this assumes that <ul><li>you stick to the savings plan as proposed,</li><li>you invest in a
  portfolio that follows the benchmark (ie. is rebalanced every month), and that </li><li>the financial markets
  behave as expected (following the expected returns and correlations as you have put in the
  system).</li></ul> Press <F5> (or press [simulate again]) to see another simulation, or change the simulation duration
  (on top of the page) in order to see the simulation over a different time horizon."},

{"follow_up","<h4>Following Up Your Life Goals</h4>It is essential to follow up on the performance of your
  portfolios at least once per year. We recommend to do this twice per year. If the dashboard is all green,
  then no actions are required, if some goals light up in amber or red, You should reconsider. Also note that
  goals that are in green might have a lower risk than you would want ... that is then an opportunity to re-run
  the exercise and do something more for other or even new goals."}

}

```

Definition at line 108 of file [texts_Eng_UK.h](#).

8.53.1.20 unordered_map<string, string> t_feedback

Initial value:

```

=
{
{"brightgreen", "This goal looks realistic and safe. Follow the suggested benchmark and savings plan and
  follow the suggested benchmark, and do not forget to review your situaton annually."},
{"green", "This goal should be reasonably safe, however in a worst case scenario, there it is not easy to
  re-allocate more money to it. Follow the suggested benchmark and savings plan and follow up the plan
  regularly."},
{"amber", "While it is not impossible to reach the goal stated, it is not very save: chances are that you
  won't make it. Consider revising the goal, and follow up closely."},
{"red", "This goal is not realistic. Please reconsider to reduce your ambitions on this or higher goals
  in order to make it realistic."},
{"darkred", "There are no means left to obtain this goal. Please review your plan and priorities."}
}

```

t_feedback

the feedback for the goals (right now)

Definition at line 140 of file [texts_Eng_UK.h](#).

8.53.1.21 unordered_map<string, string> t_feedback_post

Initial value:

```

=
{
{"brightgreen", "Congratuelations, this did work out! However, this is just one simulation."},
{"green", "Congratuelations, this did work out! However, this is just one simulation."},
{"amber", "This <b>almost</b> worked out! However, this is just one simulation."},
{"red", "Bad Luck in this simulation! This underlines the importance of a regular review of your
  portfolio!"},
{"darkred", "Bad Luck in this simulation! This underlines the importance of a regular review of your
  portfolio!"}
}

```

t_feedback_post

the feedback for the goals (after the goal has been realized – in a simulation)

Definition at line 154 of file [texts_Eng_UK.h](#).

8.53.1.22 `const string t_hide_extra_info = "Hide More Info"`

Definition at line 105 of file [texts_Eng_UK.h](#).

8.53.1.23 `const string t_Login = "Login"`

Definition at line 27 of file [texts_Eng_UK.h](#).

8.53.1.24 `const string t_month = "month"`

General words.

Definition at line 11 of file [texts_Eng_UK.h](#).

8.53.1.25 `const string t_months = "months"`

Definition at line 13 of file [texts_Eng_UK.h](#).

8.53.1.26 `const string t_next_step = "next step"`

Definition at line 18 of file [texts_Eng_UK.h](#).

8.53.1.27 `unordered_map<int, string> t_plotlabels_evol`

Initial value:

```
=
{
  {1, "Negative Scenario"},

  {2, "Median"},

  {3, "Positive Scenario"},
  {4, "YOUR GOAL#"},
  {5, "Simulated Evolution"}
}
```

`t_plotlabels_evol`

Definition at line 199 of file [texts_Eng_UK.h](#).

8.53.1.28 `const string t_prev_step = "prev step"`

Definition at line 17 of file [texts_Eng_UK.h](#).

8.53.1.29 `unordered_map<string, string> t_regFrm`

Initial value:

```
=
{
  {"policyNote", "<p class=\"markup\">Note that we place one cookie on your device in order to track who is
  logged in. Pressing submit means that you agree with this.</p>"},
  {"requiredFields", "<p>[*] Required fields.</p>"}
}
```

Definition at line 77 of file [texts_Eng_UK.h](#).

8.53.1.30 `const string t_Register = "Register"`

Definition at line 28 of file [texts_Eng_UK.h](#).

8.53.1.31 `const string t_reset = "reset"`

Definition at line 19 of file [texts_Eng_UK.h](#).

8.53.1.32 `const string t_save = "save"`

Definition at line 20 of file [texts_Eng_UK.h](#).

8.53.1.33 `unordered_map<int, string> t_scale125i2s`

Initial value:

```
=  
{  
  {1, "none / not at all"},  
  {2, "a little"},  
  {3, "average"},  
  {4, "much / more than half"},  
  {5, "all / best / very good"}  
}
```

`t_scale125i2s` a scale from one to five

Definition at line 188 of file [texts_Eng_UK.h](#).

8.53.1.34 `const string t_show_extra_info = "Show More Info"`

`t_extraInfo` : the extra help on each screen that can be rolled up

note that the key field is the `bona_class.class_name`

Definition at line 104 of file [texts_Eng_UK.h](#).

8.53.1.35 `const string t_showhide_extra_info = "Show/Hide Additional Information"`

Definition at line 106 of file [texts_Eng_UK.h](#).

8.53.1.36 `const string t_sim_till = "simulate till age"`

Definition at line 34 of file [texts_Eng_UK.h](#).

8.53.1.37 `const string t_simulation_duration = "Simulation Duration"`

Definition at line 29 of file [texts_Eng_UK.h](#).

8.53.1.38 `const string t_title = "Goal-Ranking"`

Definition at line 7 of file [texts_Eng_UK.h](#).

8.53.1.39 unordered_map<string, string> t_titles

Initial value:

```
=
{
  {"welcome",      "Welcome!"},
  {"goal",         "Your Personal Goals"},
  {"inventory",    "The Means To Obtain Your Goals"},
  {"cash_flow",    "Your FUTURE Savings"},
  {"asset",        "Your ACTUAL Savings and Investments"},
  {"register",     "Register"},
  {"account",     "Account Settings"},
  {"login",        "Login"},
  {"personalize", "Personalize Your Experience"},
  {"feedback",    "Your Personal Financial Plan"},
  {"simulation",   "What Can Happen: ONE Possible Scenario"},
  {"follow_up",   "Mockup Future Report"}
}
```

t_titles : the titles for each section

Definition at line 58 of file [texts_Eng_UK.h](#).

8.53.1.40 `const string t_warning = "<p>This software is provided for research purposes only, it is not intended to be used as a foundation for your personal financial decisions! Please talk to the author or to a financial advisor.</p><p>The goal of this website is to show advisers (such as independent financial planners, bankers, insurance agents, etc.) how real responsible personal financial advice can look like. The author, " + t_author + ", invites you to contact him by e-mail to discuss how you can make your advice more responsible, forge more durable relations with your customers and become a leader in the digital race.</p>"`

Definition at line 8 of file [texts_Eng_UK.h](#).

8.53.1.41 `const string t_year = "year"`

Definition at line 12 of file [texts_Eng_UK.h](#).

8.53.1.42 `const string t_years = "years"`

Definition at line 14 of file [texts_Eng_UK.h](#).

8.54 texts_Eng_UK.h

```
00001 /*****
00002 * texts_Eng_UK.h
00003 *
00004 * header file that contains text (html) messages
00005 * *****/
00006 const string t_author = "<a href=\"mailto:philippe@de-brouwer.com\">Dr. Philippe De Brouwer</a>";
00007 const string t_title = "Goal-Ranking";
00008 const string t_warning = "<p>This software is provided for research purposes only, it is not
intended to be used as a foundation for your personal financial decisions! Please talk to the author or to a
financial advisor.</p><p>The goal of this website is to show advisers (such as independent financial planners,
bankers, insurance agents, etc.) how <strong>real responsible personal financial advice</strong> can look
like. The author, " + t_author + ", invites you to contact him by e-mail to discuss how you can make
your advice more responsible, forge more durable relations with your customers and become a leader in the
digital race.</p>";
00009
00011 const string t_month      = "month";
00012 const string t_year       = "year";
00013 const string t_months    = "months";
00014 const string t_years     = "years";
00015 const string t_and       = "and";
00016 const string t_Asset_Class = "Asset Class";
00017 const string t_prev_step  = "prev step";
00018 const string t_next_step  = "next step";
00019 const string t_reset     = "reset";
```

```

00020 const string t_save      = "save";
00021 const string t_edit      = "edit";
00022 const string t_delete    = "delete";
00023 const string t_cancel    = "cancel";
00024 const string t_clear_form = "clear form";
00025 const string t_collapse  = "collapse";
00026
00027 const string t_Login      = "Login";
00028 const string t_Register   = "Register";
00029 const string t_simulation_duration = "Simulation Duration";
00030
00031 const string t_currency   = "currency";
00032 const string t_b_currency = "base currency";
00033 const string t_b_date     = "birth date";
00034 const string t_sim_till   = "simulate till age";
00035 const string t_breadcrumb = "You are here:&nbsp;";
00036
00037 /* *****
00038  * t_aside : the aside remarks
00039  * *****/
00040 unordered_map<string, string> t_aside =
00041 {
00042     {"welcome", "Welcome"},
00043     {"login", "Login"},
00044     {"register", "Register"},
00045     {"account", "Account Settings"},
00046     {"goalInfo", "Try to quantify your important life-goals. It is better to state a goal that is 60% correct
00047     than stating no goal at all!"},
00047     {"goalHelpFail", "Please try again. Fill out only the goals that you want to use and use realistic data!"
00048     },
00048     {"cash_flowInfo", "What is your savings-capacity? How will that change?"},
00049     {"feedback", "We hope that you liked this solution and that it was worth your time!<br>Please let us know
00049     <a href=\"mailto:philippe@de-brouwer.com\">via email</a> and spread the word!<br>Thank you!"},
00050     {"simulation", "Here we show you what can happen if you would not follow up the portfolio."},
00051     {"follow_up", "Please make sure that you follow up your goals regularly!<br>Wouldn't it be nice to have a
00051     webservice sending you this dashboard every month?<br>Please let us know what you did not like via <a href=
00052     \"mailto:philippe@de-brouwer.com\">via email</a> and tell your friends what you did like!<br>Thank you!"},
00052     {"disclaimer", "We will never sell your email, nor spam you. We might send you a newsletter from time to
00052     time and all data that you provide in the system will be kept there till you delete your account or
00052     overwrite it."}
00053 };
00054
00058 unordered_map<string, string> t_titles =
00059 {
00060     {"welcome", "Welcome!"},
00061     {"goal", "Your Personal Goals"},
00062     {"inventory", "The Means To Obtain Your Goals"},
00063     {"cash_flow", "Your FUTURE Savings"},
00064     {"asset", "Your ACTUAL Savings and Investments"},
00065     {"register", "Register"},
00066     {"account", "Account Settings"},
00067     {"login", "Login"},
00068     {"personalize", "Personalize Your Experience"},
00069     {"feedback", "Your Personal Financial Plan"},
00070     {"simulation", "What Can Happen: ONE Possible Scenario"},
00071     {"follow_up", "Mockup Future Report"}
00072 };
00073
00074 /* *****
00075  * t_regFrm : the registration form
00076  * *****/
00077 unordered_map<string, string> t_regFrm =
00078 {
00079     {"policyNote", "<p class=\"markup\">Note that we place one cookie on your device in order to track who is
00079     logged in. Pressing submit means that you agree with this.</p>"},
00080     {"requiredFields", "<p>[*] Required fields.</p>"}
00081 };
00082
00083 /* *****
00084  * t_errMsg : the error messages
00085  * *****/
00086 unordered_map<string, string> t_errMsg =
00087 {
00088     {"loginFirst", "<li>You need to login before accessing these sections!</li>"},
00089     {"errorSavingUser", "<li>Undefined error saving this user information.</li>"},
00090     {"userNameTaken1", "<li>Sorry, the user name &quot;"},
00091     {"userNameTaken2", "&quot; is already taken!</li>"},
00092     {"pwdsMatch", "<li>the passwords provided do not match</li>"},
00093     {"pwdUserCombination", "<li>Sorry, I could not retrieve this combination of userID and password
00093     :-(<br>Please check the if CapsLock is not on and retype the password.</li>"},
00094     {"retrievingInvestor", "<li>Error occurred retrieving this user from the database, contact
00094     administrator</li>"},
00095     {"no_goals", "<li>Please state at least ONE goal first!</li>"},
00096     {"no_assets", "<li>Please define assets and future savings capacity!</li>"}
00097 };
00098
00104 const string t_show_extra_info = "Show More Info";

```

```

00105 const string t_hide_extra_info      = "Hide More Info";
00106 const string t_showhide_extra_info   = "Show/Hide Additional Information";
00107 const string t_extra_info           = "Extra Information";
00108 unordered_map<string, string> t_extraInfo =
00109 {
00110     {"welcome", "<p><h4>Welcome</h4>This simple process consists of three easy steps:</p><ol><li>Register or
Login</li><li>STEP 1: Make an inventory of your assets and main cash-flows (income and
expenses)</li><li>STEP 2: Set Your personal financial goals</li><li>STEP 3: See your personalized Feedback and eventual
step 1 and/or 2.</li></ol>The basic idea is to provide real investor centric investment advice by starting
from your personal goals and savings. At the end of this exercercise, you should have feedback about which
goals are realistic, what savings-plan you need and eventually what composition each portfolio should have
(benchmark).<h4>When Registering</h4><p>Note that this sign-up is only to be able to recognize you in the tool
(so that you can find the information that you provide back later. We ask also a little more information that
login and password. The reason for that is that this tool will help you to define the best fitted investment
portfolio for <strong>you</strong>. That is a very personal thing, and so we need a few personal things. If
you provide wrong information, then the results will be wrong.)</p><p>However, in order to guarantee your
anonymity you can provide for example an alternate name and multiply all your goals and investments with a
certain multiplier that only you know (e.g times 5).</p>"},

00111     {"goal", "<h4>Investment Goals</h4><p>Here you need to inform the system about what you would like to
obtain in your life. This is the most important part of this software!</p><p><b>Important:</b> This software is
only a demo version, we appologize for the inconvenience, but please bear in min that:</p><ul><li>for the
goal type <quot><b>amount@date</b><quot> the age of realization is key,</li><li>for the goal type
<quot><b>income from/to</b><quot> the from-date, till-date and frequency are key</li><li>for the goal-type
<b><quot>rainy day savings</b><quot> no date information is necessary, and</li><li>the goal-type
<quot><b>unallocated</b><quot> is actually not to be used. The software will use it to allocate all assets that are not
any other goal. So you would need only one <quot>unallocated<quot> goal.</li></ul><p>All the information that
we ask for is needed in order to select for you the right investment portfolio and savings plan. There are
two important remarks to take into account</p><ol><li><strong>First: there is a thin line between <quot>Cash
Flows<quot> and plans.</strong> If you need to buy for example a new car next year (and that expense is
certain, necessary and fixed ... then it is possible to add it as well as a cash flow and as a goal. However in
this case (short term, certain and no degree of flexibility) it is advisable to first add it as a cash flow
and hope it all works out. Only when you decide that the car can eventually cost less in order to finance
other goals, then it should be considered as a goal.</li><li><strong>Second: in this early demo version, we
do not guide you too much about which goal should be the most important</strong> (in other words: sorry, but
we do not use the need level yet). So the order that you provide in <quot>ranking<quot> is very important!
Please put a number (eg. one) for your most important goal and put higher numbers for the others (yes in this
early demo it is not important what numbers you use)</li></ol><p>Remember the old addagio that a goal has
to be SMART (Specific, Measurable, Attainable, Relevant and Time Bound).</p>"},

00113     {"cash_flow", "<h4>Cash flows</h4>Cash flows are your additional savings. It can simply be the average
amount that you can save monthly. However if other amounts might be useful to include. But, be careful which
point of view you take!. For example if you save now 500,- per month and your kids are about to leave college
in 3 years (which will give you an additional saving capacity of 750,- per month. Well in that case you
could book one cash flow of 500,- starting today and ending when you retire, and another one of 750,- starting
within three years and also till you retire.<br>If on the other hand your kids will go to college in 5
years, then it is wise to foresee <i>negative</i> cash flow starting in 5 years and running for 4 years (eg.
-660,- (mind the minus sign!)).<br><br>HINT:</b></ol><p>in order to make a cash flow start right away: leave the
<quot>From Date<quot> empty; in order to make a cash flow available till your retirement: leave the <quot>Till
Date<quot> empty."},

00115     {"register", "<h4>Thanks for signing up and testing our tool!</h4><p>Note that this sign-up is only to be
able to recognize you in the tool (so that you can find the information that you provide back later. We ask
also a little more information that login and password. The reason for that is that this tool will help you
to define the best fitted investment portfolio for <strong>you</strong>. That is a very personal thing, and
so we need a few personal things. If you provide wrong information, then the results will be wrong (well
that does not exclude that this might be interesting to test of course).<br>So, indeed, signup up here is just
to use the tool, this means that we can of course show you its potential by sending relevant messages for
the investment case that you have put into the system. We will not use your email to send <quot>marketing
mails<quot>; that's why we have our newsletter! You can sign-up for it soon ;-)</p>"},

00117     {"account", "<h4>Keep your account information up to date!</h4><p>Note that this sign-up is only to be
able to recognize you in the tool(so that you can find the information that you provide back later. We ask
also a little more information that login and password. The reason for that is that this tool will help you to
define the best fitted investment portfolio for <strong>you</strong>. That is a very personal thing, and so
we need a few personal things. If you provide wrong information, then the results will be wrong (well that
does not exclude that this might be interesting to test of course).<br>So, indeed, signup up here is just to
use the tool, this means that we can of course show you its potential by sending relevant messages for the
investment case that you have put into the system. We will not use your email to send <quot>marketing
mails<quot>; that's why we have our newsletter! You can sign-up for it soon ;-)</p>"},

00119     {"login", "<h4>Welcome Back!</h4>Use this screen only if you have registered before and you already have
a login and password. Otherwise, please register first!<br>Also remember that the fact that you have a login
to test our tools is no guarantee that you will get any information from us. To be kept up to date, please
sign up for our newsletter [not available yet]"},

00121     {"personalize", t_personalize_concepts +
00122         t_personalize_scoring},

00123     {"feedback", "So, here you are! But what now?<br>What do you think? Does it look good? Can you reach the
00124         goals that you have put yourself? Maybe you have still money left, then you can add more goals and/or
increase the amounts for certain goals.<br>But maybe you notice that you have more goals than money. The positive
side of this is that you have an interesting life and that you have plenty of plans. Now you can do two
things<ol><li>decrease the amounts of the goals</li><li>accept more shortfall on the goals</li><li>or simply
leave the less important goals out</li></ol>. Note also that this feedback is shaped by your personal
selections for and preferences (you might have excluded certain asset classes in the [Preferences] screen for
example). <p>In any case: thank you very much for your time, we hope that it was an interesting experience. Please

```

```

sign up for our newsletter and we promise to keep you informed."},
00125
00126 {"asset","Describe your assets here. They will be used as an initial amount that can be used to invest.
We will not use the existing portfolio composition as a starting point, but largely describing your assets
here will help later to inform you about what to sell and what to buy. Please enter here all assets that you
consider to be available for sale. If for example you intend to keep your house as long as you live and then
pass it on to your children, then you should not mention it here. Of course, it is up to you to decide this!
Maybe leave it out and see if you can reach your goals, but maybe you could consider a smaller place to
live but reach those important goals? Up to you, it's your life!"},
00127
00128 {"simulation","<h4>Simulaton</h4>In this screen we show you what might possibly happen to your
investments. Of course, this assumes that <ul><li>you stick to the savings plan as proposed,</li><li>you invest in a
portfolio that follows the benchmark (ie. is rebalanced every month), and that </li><li>the financial markets
behave as expected (following the expected returns and correlations as you have put in the
system).</li></ul> Press <F5> (or press [simulate again]) to see another simulation, or change the simulation duration
(on top of the page) in order to see the simulation over a different time horizon."},
00129
00130 {"follow_up","<h4>Following Up Your Life Goals</h4>It is essential to follow up on the perfrmance of your
portfolios at least once per year. We recommend to do this twice per year. If the dashboard is all green,
then no actions are required, if some goals light up in amber or red, You should reconsider. Also note that
goals that are in green might have a lower risk than you would want ... that is then an opportunity to re-run
the exercercise and do something more for other or even new goals."}

00131
00132 };
00133
00134
00140 unordered_map<string, string> t_feedback =
00141 {
00142 {"brightgreen", "This goal looks realistic and safe. Follow the suggested benchmark and savings plan and
follow the suggested benchmark, and do not forget to review your situation annually."},
00143 {"green", "This goal should be reasonably safe, however in a worst case scenario, there it is not easy to
re-allocate more money to it. Follow the suggested benchmark and savings plan and follow up the plan
regularly."},
00144 {"amber", "While it is not impossible to reach the goal stated, it is not very save: chances are that you
won't make it. Consider revising the goal, and follow up closely."},
00145 {"red", "This goal is not realistic. Please reconsider to reduce your ambitions on this or higher goals
in order to make it realistic."},
00146 {"darkred", "There are no means left to obtain this goal. Please review your plan and priorities."}
00147 };
00148
00154 unordered_map<string, string> t_feedback_post =
00155 {
00156 {"brightgreen", "Congratuelations, this did work out! However, this is just one simulation."},
00157 {"green", "Congratuelations, this did work out! However, this is just one simulation."},
00158 {"amber", "This <b>almost</b> worked out! However, this is just one simulation."},
00159 {"red", "Bad Luck in this simulation! This underlines the importance of a regular review of your
portfolio!"},
00160 {"darkred", "Bad Luck in this simulation! This underlines the importance of a regular review of your
portfolio!"}
00161 };
00162
00163 const string t_dashboard = "Dashboard";
00164
00171 unordered_map<string, string> t_before_list =
00172 {
00173 {"goal", "Need a help? Use the <a class=\"button\" href=\"\" + oConfig.
soft_url + \"?a=gh\">support screen to add goals</a>."}
00174 };
00175
00179 unordered_map<string, string> t_after_title =
00180 {
00181 {"Home", "<h4>Welcome to real investment advice!</h4><p>Here You find personalized advice that rivals the
best private bank, but fully automated, because we start from your life-goals and take your goals
seriously.</p>"}
00182 };
00183
00188 unordered_map<int, string> t_scale125i2s =
00189 {
00190 {1, "none / not at all"},
00191 {2, "a little"},
00192 {3, "average"},
00193 {4, "much / more than half"},
00194 {5, "all / best / very good"}
00195 };
00199 unordered_map<int, string> t_plotlabels_evov =
00200 {
00201 {1, "Negative Scenario"},
00202 #ifdef SHOW_Vexp_not_Vmed
00203 {2, "Expected Evolution"},
00204 #else
00205 {2, "Median"},
00206 #endif
00207 {3, "Positive Scenario"},
00208 {4, "YOUR GOAL#"}, //note that the number of the goals will be collated to this
00209 {5, "Simulated Evolution"}
00210 };

```

Index

- ~db_helper
 - db_helper, 38
- a
 - normdist.h, 229
- aGOAL, 13
 - aGOAL, 14
 - add_to_db, 15
 - aGOAL, 14
 - alpha, 16
 - amount, 16
 - benchmark, 16
 - color, 16
 - color_followup, 17
 - currency, 17
 - description, 17
 - frequency, 17
 - from_date, 17
 - goal_id, 17
 - goal_type, 17
 - investor, 17
 - max_shortfall, 17
 - means_goal, 17
 - operator<<, 16
 - optimal_portf, 17
 - priority, 18
 - realization_age, 18
 - realization_monthNbr, 18
 - realized_shortfall, 18
 - remarks, 18
 - s_exp, 18
 - s_goal, 18
 - s_high, 18
 - s_low, 18
 - save, 15
 - save_results, 15
 - saving_plan, 18
 - set_amount, 15
 - set_currency, 15
 - set_description, 15
 - set_frequency, 15
 - set_from_date, 15
 - set_from_db, 15
 - set_goal_type, 15
 - set_max_shortfall, 15
 - set_priority, 15
 - set_realization_age, 16
 - set_till_date, 16
 - set_to_pstmnt, 16
 - show_graph_till_MNbr, 16
 - simulation_endvalue, 18
 - simulation_string, 19
 - success, 19
 - target_amount, 19
 - till_date, 19
- aGOAL.struct.cpp, 115
 - operator<<, 115
- ALPHA_LOWER_LIMIT
 - efp.cpp, 144
- ALPHA_PLOT_HIGH
 - efp.cpp, 144
- ALPHA_PLOT_LOW
 - efp.cpp, 144
- ALPHA_UPPER_LIMIT
 - efp.cpp, 144
- account_form
 - investor_ui, 85
- account_save
 - investor_ui, 85
- action_ref
 - bona, 25
 - cash_flow, 28
 - goal, 52
- active_action
 - investor_ui, 96
- add0
 - global_functions.h, 155
- add_to_db
 - aGOAL, 15
 - follow_up, 42
 - investment_problem, 64
 - investor, 77
 - investor_ui, 85
 - simulation, 105
- age
 - follow_up, 42
 - investment_problem, 64
 - investor, 77
 - investor_ui, 85
 - simulation, 105
- age2monthNbr
 - follow_up, 42
 - investment_problem, 64
 - investor, 78
 - investor_ui, 85
 - simulation, 105
- allocate_to_unallocated_goal
 - investment_problem, 64
- alpha

- aGOAL, 16
- amount
 - aGOAL, 16
 - asset, 21
- aside
 - html_helper, 54
- asset, 19
 - amount, 21
 - asset_class, 21
 - asset_delete, 20
 - asset_id, 21
 - currency, 21
 - description, 21
 - exists, 20
 - investor, 21
 - list_assets, 20
 - load, 20
 - save, 20
 - set_amount, 20
 - set_asset_class, 20
 - set_asset_id, 20
 - set_currency, 21
 - set_description, 21
 - set_investor, 21
 - show_edit_form, 21
- asset.class.cpp, 120
- asset_add
 - investor_ui, 86
- asset_class, 22
 - asset, 21
 - asset_class_id, 22
 - asset_class_name, 22
 - dropDown_assetClass, 22
 - E_R, 22
 - exists, 22
 - get_assetList, 22
- asset_class.class.cpp, 124
- asset_class_id
 - asset_class, 22
- asset_class_name
 - asset_class, 22
- asset_class_names
 - investor_ui, 96
- asset_delete
 - asset, 20
 - investor_ui, 86
- asset_edit
 - investor_ui, 86
- asset_id
 - asset, 21
- asset_save
 - investor_ui, 86
- assetClass_covar
 - follow_up, 47
 - investment_problem, 72
 - investor, 80
 - investor_ui, 96
 - market, 100
 - simulation, 111
- assetClass_mu
 - follow_up, 47
 - investment_problem, 72
 - investor, 80
 - investor_ui, 96
 - market, 100
 - simulation, 111
- assetClass_name
 - follow_up, 47
 - investment_problem, 72
 - investor, 80
 - investor_ui, 96
 - market, 100
 - simulation, 111
- assetClassColors
 - html_helper, 58
- b
 - normdist.h, 229
- benchmark
 - aGOAL, 16
- birth_date
 - follow_up, 47
 - investment_problem, 72
 - investor, 80
 - investor_ui, 96
 - simulation, 111
- bona, 23
 - action_ref, 25
 - class_name, 25
 - d_get, 24
 - defList, 25
 - delete_bona, 24
 - exists, 24
 - f_get, 24
 - i_get, 24
 - list, 24
 - load_fromDB, 24
 - load_fromEnv, 24
 - load_fromRecordSet, 24
 - order_by, 25
 - s_get, 24
 - save, 24
 - show, 24
 - show_edit_form, 25
 - show_tc, 25
 - show_th, 25
 - tbl, 25
 - tbl_name, 25
 - verify, 25
- bona.class.cpp, 125
- bona_cf.class.cpp, 130
- bona_goal.class.cpp, 132
- bootstrap, 11
 - bs_modal, 11
 - bs_navbar_end, 11
 - bs_navbar_end1, 11
 - bs_navbar_end2, 11

- bs_navbar_item, 12
- bs_navbar_start, 12
- bootstrap.namespace.cpp, 133
- bs_modal
 - bootstrap, 11
- bs_navbar_end
 - bootstrap, 11
- bs_navbar_end1
 - bootstrap, 11
- bs_navbar_end2
 - bootstrap, 11
- bs_navbar_item
 - bootstrap, 12
- bs_navbar_start
 - bootstrap, 12
- button_to
 - investor_ui, 86
- c
 - normdist.h, 229
- c_arrow
 - html_helper, 58
- c_content_header
 - html_helper, 58
- c_doc_type
 - html_helper, 58
- c_photos_phdb
 - html_helper, 58
- c_photos_phdb_dir
 - html_helper, 59
- c_right_menu_names
 - html_helper, 59
- c_right_menu_urls
 - html_helper, 59
- c_salt
 - html_helper, 59
- CURR_LEN
 - efp.cpp, 145
- calc_V_high
 - investment_problem, 65
- calc_alpha
 - investment_problem, 64
- calc_risk
 - investment_problem, 64
- cash_flow, 26
 - action_ref, 28
 - cash_flow, 27
 - cash_flow, 27
 - class_name, 28
 - defList, 28
 - delete_bona, 27
 - exists, 27
 - list, 27
 - load_fromDB, 27
 - load_fromEnv, 27
 - order_by, 28
 - save, 27
 - show, 27
 - show_edit_form, 27
 - tbl, 28
 - tbl_name, 28
- cash_flow_add
 - investor_ui, 86
- cash_flow_delete
 - investor_ui, 86
- cash_flow_edit
 - investor_ui, 86
- cash_flow_list
 - investor_ui, 86
- cash_flow_save
 - investor_ui, 86
- cgi
 - efp.cpp, 148
- class_name
 - bona, 25
 - cash_flow, 28
 - goal, 52
- cls_currency, 28
 - conversion_factor, 29
 - currency_id, 29
 - currency_name, 29
 - dropDown_currency, 29
 - entity, 29
 - set_curr, 29
 - to_curr, 29
- cls_frequency, 29
 - frequency_id, 30
 - get_freq_label, 30
 - multiplier, 30
 - set_freq, 30
- col_type
 - column, 32
- collapse_close
 - html_helper, 54
- collapse_open
 - html_helper, 54
- color
 - aGOAL, 16
- color_followup
 - aGOAL, 17
- column, 30
 - col_type, 32
 - column_name, 32
 - dValue, 33
 - data_align, 32
 - drop_down_default, 32
 - drop_down_keyField, 32
 - drop_down_keyFieldQuote, 32
 - drop_down_showField, 32
 - drop_down_table, 33
 - fValue, 33
 - get_dValue, 31
 - get_fValue, 31
 - get_iValue, 31
 - get_sValue, 31
 - iValue, 33
 - input_type, 33

- label, 33
- placeholder, 33
- post_code, 33
- quote, 31
- sValue, 33
- set_value, 32
- show, 32
- show_inputField, 32
- show_with_previous, 33
- column.class.cpp, 135
- column_name
 - column, 32
- con
 - db_helper, 38
- config, 34
 - css_dir, 35
 - db_host, 35
 - db_name, 35
 - db_pwd, 35
 - db_user, 35
 - font_combination, 35
 - img_dir, 35
 - language, 35
 - layout, 35
 - merge_ER_vol, 35
 - parse_wrapper, 35
 - riskFunction, 36
 - scripts_dir, 36
 - show_aside, 36
 - show_social, 36
 - simul_show_table, 36
 - soft_dir, 36
 - soft_file, 36
 - soft_name, 36
 - soft_server_url, 36
 - soft_url, 36
 - soft_version, 37
 - tbl_prefix, 37
 - total_portf_monthNbrs, 37
 - wrapper_name, 37
 - wrapper_url, 37
 - www_dir, 37
- config.class.cpp, 137
 - LOCAL, 137
- conversion_factor
 - cls_currency, 29
- cookie_policy
 - html_helper, 54
- cout_box_color
 - html_helper, 55
- covar
 - follow_up, 47
 - investment_problem, 72
 - investor, 81
 - investor_ui, 96
 - market, 100
 - simulation, 111
- css_dir
 - config, 35
- curr_format
 - global_functions.h, 155
- currFormat
 - global_functions.h, 155
- currency
 - aGOAL, 17
 - asset, 21
 - follow_up, 48
 - investment_problem, 72
 - investor, 81
 - investor_ui, 96
 - simulation, 111
- currency.class.cpp, 138
- currency_id
 - cls_currency, 29
- currency_name
 - cls_currency, 29
- d
 - normdist.h, 229
- d_get
 - bona, 24
- DEFAULT_SCALE
 - efp.cpp, 145
- dValue
 - column, 33
- data_align
 - column, 32
- date_format_dateStr
 - global_functions.h, 155
- date_format_tm
 - global_functions.h, 155
- dateStr2Age
 - follow_up, 42
 - investment_problem, 65
 - investor, 78
 - investor_ui, 86
 - simulation, 105
- dateStr2Mnbr
 - global_functions.h, 155
- dateStr2tm
 - global_functions.h, 155
- db
 - db_helper, 38
 - efp.cpp, 148
- db_helper, 37
 - ~db_helper, 38
 - con, 38
 - db, 38
 - db_helper, 38
 - db_helper, 38
 - driver, 38
 - getFloat, 38
 - getInt, 38
 - pstmt, 38
 - res, 39
 - res2, 39
 - runSQL, 38

- stmt, 39
- db_helper.class.cpp, 139
- db_host
 - config, 35
- db_name
 - config, 35
- db_pwd
 - config, 35
- db_user
 - config, 35
- defList
 - bona, 25
 - cash_flow, 28
 - goal, 52
- delete_bona
 - bona, 24
 - cash_flow, 27
 - goal, 51
- description
 - aGOAL, 17
 - asset, 21
 - follow_up, 48
 - investment_problem, 72
 - portfolio, 102
 - simulation, 112
- desirability
 - follow_up, 48
 - investment_problem, 73
 - investor, 81
 - investor_ui, 97
 - simulation, 112
- driver
 - db_helper, 38
- drop_down_default
 - column, 32
- drop_down_keyField
 - column, 32
- drop_down_keyFieldQuote
 - column, 32
- drop_down_showField
 - column, 32
- drop_down_table
 - column, 33
- dropDown_assetClass
 - asset_class, 22
- dropDown_currency
 - cls_currency, 29
- E_R
 - asset_class, 22
- ER
 - follow_up, 48
 - investment_problem, 73
 - investor, 81
 - investor_ui, 97
 - market, 100
 - simulation, 112
- efp.cpp, 142
 - ALPHA_LOWER_LIMIT, 144
 - ALPHA_PLOT_HIGH, 144
 - ALPHA_PLOT_LOW, 144
 - ALPHA_UPPER_LIMIT, 144
 - CURR_LEN, 145
 - cgi, 148
 - DEFAULT_SCALE, 145
 - db, 148
 - error_message, 148
 - goal_type_i2s, 149
 - goal_type_s2i, 149
 - hh, 149
 - INDEX, 145
 - INPUT_WIDTH_FLOAT, 145
 - INPUT_WIDTH_INT, 145
 - INPUT_WIDTH_TEXT, 145
 - MAX_AGE, 145
 - MAX_ALPHA, 145
 - MAX_SCALE, 146
 - MIN_ALPHA, 146
 - MIN_SCALE, 146
 - main, 148
 - NBR_ASSET_CLASSES, 146
 - NBR_ITERATIONS, 146
 - NBR_RIGHT_MENU, 146
 - NBR_SCALE, 147
 - NBR_SUB_TABS, 147
 - NBR_TABS, 147
 - oAsset, 149
 - oAssetClass, 149
 - oCF, 149
 - oConfig, 149
 - oCurrency, 150
 - oGoal, 150
 - oInvestor, 150
 - PRECISION, 147
 - RETIREMENT_AGE, 147
 - SECONDS_IN_MONTH, 147
 - std_message, 150
- entity
 - cls_currency, 29
- erfinv
 - normdist.h, 228
- error_message
 - efp.cpp, 148
- exists
 - asset, 20
 - asset_class, 22
 - bona, 24
 - cash_flow, 27
 - follow_up, 42
 - goal, 51
 - investment_problem, 65
 - investor, 78
 - investor_ui, 87
 - simulation, 106
- expand_cf
 - investment_problem, 65
- experience

- follow_up, 48
 - investment_problem, 73
 - investor, 81
 - investor_ui, 97
 - simulation, 112
- f_get
 - bona, 24
- fValue
 - column, 33
- fast_erfinv
 - normdist.h, 228
- feedback
 - investor_ui, 87
- first_name
 - follow_up, 48
 - investment_problem, 73
 - investor, 81
 - investor_ui, 97
 - simulation, 112
- flSum
 - global_functions.h, 155
- follow_up, 39
 - add_to_db, 42
 - age, 42
 - age2monthNbr, 42
 - assetClass_covar, 47
 - assetClass_mu, 47
 - assetClass_name, 47
 - birth_date, 47
 - covar, 47
 - currency, 48
 - dateStr2Age, 42
 - description, 48
 - desirability, 48
 - ER, 48
 - exists, 42
 - experience, 48
 - first_name, 48
 - follow_up, 41
 - follow_up, 41
 - get_description, 42
 - get_from_db, 42
 - get_full_name, 42
 - get_max_month_for_lower_goals, 42
 - get_nbr_goals, 43
 - get_nbrMonths2simulate, 43
 - get_portf_from_db, 43
 - goalZ, 48
 - investor_id, 48
 - javaGraph, 43
 - knowledge, 48
 - last_name, 48
 - load_ER, 43
 - load_covar, 43
 - load_from_db, 43
 - load_preferences, 44
 - market_return, 48
 - max_exposure, 48
 - months2simulate, 44
 - nbrMonths, 49
 - pLogR, 49
 - pMu, 49
 - pSigma, 49
 - parse_dashboard, 44
 - parse_javaGraph, 44
 - password, 49
 - plot_market_evol, 44
 - portfolio_id, 49
 - portfolios, 49
 - prepare_javaVars_colors, 44
 - reload_ER, 45
 - reload_covar, 45
 - save, 45
 - save_ER, 45
 - save_covar, 45
 - save_preferences, 45
 - set_followup_color_post, 46
 - set_followup_color_prae, 46
 - set_max_exposure, 46
 - set_mu, 46
 - set_scale, 46
 - set_sigma, 46
 - show_simulation, 46
 - simulate, 46
 - simulate_till_age, 49
 - solve, 47
 - user_name, 49
 - weights, 49
- follow_up.class.cpp, 153
- font_combination
 - config, 35
- footer
 - html_helper, 55
- frequency
 - aGOAL, 17
- frequency.class.cpp, 153
- frequency_id
 - cls_frequency, 30
- from_date
 - aGOAL, 17
- get_assetList
 - asset_class, 22
- get_box_color_class
 - html_helper, 55
- get_dValue
 - column, 31
- get_description
 - follow_up, 42
 - investment_problem, 65
 - portfolio, 101
 - simulation, 106
- get_fValue
 - column, 31
- get_freq_label
 - cls_frequency, 30
- get_from_db

- follow_up, 42
- investment_problem, 65
- investor, 78
- investor_ui, 87
- simulation, 106
- get_full_name
 - follow_up, 42
 - investment_problem, 65
 - investor, 78
 - investor_ui, 87
 - simulation, 106
- get_glyphicon
 - html_helper, 55
- get_iValue
 - column, 31
- get_max_month_for_lower_goals
 - follow_up, 42
 - investment_problem, 65
 - simulation, 106
- get_nbr_goals
 - follow_up, 43
 - investment_problem, 66
 - simulation, 106
- get_nbrMonths2simulate
 - follow_up, 43
 - simulation, 106
- get_next_step
 - html_helper, 55
- get_next_sub_step
 - html_helper, 55
- get_optimal_portfolio
 - investment_problem, 66
- get_portf_from_db
 - follow_up, 43
 - investment_problem, 66
 - portfolio, 101
 - simulation, 106
- get_portfolio_riskiest
 - investment_problem, 66
- get_portfolio_safest
 - investment_problem, 66
- get_prev_step
 - html_helper, 55
- get_prev_sub_step
 - html_helper, 55
- get_sValue
 - column, 31
- get_step_nbr
 - html_helper, 55
- get_string
 - html_helper, 56
- get_sub_step_nbr
 - html_helper, 56
- get_tm
 - global_functions.h, 156
- getFloat
 - db_helper, 38
- getInt
 - db_helper, 38
- global_functions.h, 154
 - add0, 155
 - curr_format, 155
 - currFormat, 155
 - date_format_dateStr, 155
 - date_format_tm, 155
 - dateStr2Mnbr, 155
 - dateStr2tm, 155
 - flSum, 155
 - get_tm, 156
 - is_valid_dateStr, 156
 - mNbr2dateStr, 156
 - months2yms, 156
 - Pi, 157
 - round2str, 156
 - thousand_separator, 156
 - tm2DateStr, 156
 - tm2Mnbr, 156
- goal, 50
 - action_ref, 52
 - class_name, 52
 - defList, 52
 - delete_bona, 51
 - exists, 51
 - goal, 51
 - list, 51
 - load_fromDB, 51
 - load_fromEnv, 51
 - order_by, 52
 - save, 51
 - show, 51
 - show_edit_form, 51
 - tbl, 52
 - tbl_name, 52
- goal_add
 - investor_ui, 87
- goal_delete
 - investor_ui, 87
- goal_edit
 - investor_ui, 87
- goal_help_form
 - investor_ui, 87
- goal_help_save
 - investor_ui, 87
- goal_id
 - aGOAL, 17
- goal_list
 - investor_ui, 88
- goal_save
 - investor_ui, 88
- goal_seek_means
 - investment_problem, 66
- goal_seek_tmp_means
 - investment_problem, 67
- goal_type
 - aGOAL, 17
- goal_type_i2s

- efp.cpp, 149
- goal_type_s2i
 - efp.cpp, 149
- goalZ
 - follow_up, 48
 - investment_problem, 73
 - simulation, 112
- HIGH
 - normdist.h, 227
- header
 - html_helper, 56
- header_menu_bar
 - html_helper, 56
- hh
 - efp.cpp, 149
 - investor_ui, 97
- home_screen
 - investor_ui, 88
- html_helper, 52
 - aside, 54
 - assetClassColors, 58
 - c_arrow, 58
 - c_content_header, 58
 - c_doc_type, 58
 - c_photos_phdb, 58
 - c_photos_phdb_dir, 59
 - c_right_menu_names, 59
 - c_right_menu_urls, 59
 - c_salt, 59
 - collapse_close, 54
 - collapse_open, 54
 - cookie_policy, 54
 - cout_box_color, 55
 - footer, 55
 - get_box_color_class, 55
 - get_glyphicon, 55
 - get_next_step, 55
 - get_next_sub_step, 55
 - get_prev_step, 55
 - get_prev_sub_step, 55
 - get_step_nbr, 55
 - get_string, 56
 - get_sub_step_nbr, 56
 - header, 56
 - header_menu_bar, 56
 - html_helper, 54
 - html_helper, 54
 - navbar_box, 56
 - navbar_end, 56
 - navbar_item, 56
 - navbar_start, 57
 - next_step, 57
 - parse_breadcrumb, 57
 - parse_date_time, 57
 - parse_title, 57
 - privacy_policy, 57
 - STEPDEF, 54
 - STRMAP, 54
 - scale125i2s, 59
 - seriesColorsGoalPlot, 59
 - show_args, 57
 - show_home, 57
 - show_social, 58
 - show_toolBar, 58
 - step_nbrs, 59
 - steps, 59
 - sub_step_exists, 58
 - sub_step_nbrs, 59
 - sub_steps, 59
 - terms_of_use, 58
- html_helper.class.cpp, 160
- i_get
 - bona, 24
- INDEX
 - efp.cpp, 145
- INPUT_WIDTH_FLOAT
 - efp.cpp, 145
- INPUT_WIDTH_INT
 - efp.cpp, 145
- INPUT_WIDTH_TEXT
 - efp.cpp, 145
- iValue
 - column, 33
- img_dir
 - config, 35
- init_personalize
 - t_personalize_Eng_UK.h, 238
- input_type
 - column, 33
- investment_problem, 60
 - add_to_db, 64
 - age, 64
 - age2monthNbr, 64
 - allocate_to_unallocated_goal, 64
 - assetClass_covar, 72
 - assetClass_mu, 72
 - assetClass_name, 72
 - birth_date, 72
 - calc_V_high, 65
 - calc_alpha, 64
 - calc_risk, 64
 - covar, 72
 - currency, 72
 - dateStr2Age, 65
 - description, 72
 - desirability, 73
 - ER, 73
 - exists, 65
 - expand_cf, 65
 - experience, 73
 - first_name, 73
 - get_description, 65
 - get_from_db, 65
 - get_full_name, 65
 - get_max_month_for_lower_goals, 65
 - get_nbr_goals, 66

get_optimal_portfolio, 66
get_portf_from_db, 66
get_portfolio_riskiest, 66
get_portfolio_safest, 66
goal_seek_means, 66
goal_seek_tmp_means, 67
goalZ, 73
investment_problem, 63
investment_problem, 63
investor_id, 73
javaGraph, 67
knowledge, 73
last_name, 73
load_ER, 67
load_covar, 67
load_from_db, 67
load_preferences, 67
max_exposure, 73
means, 73
means_block, 73
means_left, 74
means_tmp, 74
means_used, 74
months2simulate, 68
nbr_assets, 74
nbr_cfs, 74
nbr_goals, 74
nbr_portfolios, 74
pLogR, 74
pMu, 75
pSigma, 75
parse_javaGraph, 68
password, 74
portfolio_id, 75
portfolio_suitable, 75
portfolios, 75
prepare_javaVars_colors, 68
priorityMax, 75
priorityMin, 75
reload_ER, 68
reload_covar, 68
save, 69
save_ER, 69
save_covar, 69
save_preferences, 69
set_assets, 69
set_cfs, 69
set_followup_color_post, 69
set_followup_color_prae, 70
set_goal_color, 70
set_goal_remarks, 70
set_goals, 70
set_max_exposure, 70
set_means_goal, 70
set_mu, 71
set_portfolios, 71
set_priorityLimits, 71
set_scale, 71
set_sigma, 71
set_unallocated_goal, 71
simulate_till_age, 75
solve, 71
user_name, 75
weights, 75
investment_problem.class.cpp, 175
investor, 76
 add_to_db, 77
 age, 77
 age2monthNbr, 78
 aGOAL, 17
 asset, 21
 assetClass_covar, 80
 assetClass_mu, 80
 assetClass_name, 80
 birth_date, 80
 covar, 81
 currency, 81
 dateStr2Age, 78
 desirability, 81
 ER, 81
 exists, 78
 experience, 81
 first_name, 81
 get_from_db, 78
 get_full_name, 78
 investor, 77
 investor_id, 81
 knowledge, 81
 last_name, 81
 load_ER, 78
 load_covar, 78
 load_from_db, 78
 load_preferences, 79
 max_exposure, 81
 months2simulate, 79
 password, 81
 reload_ER, 79
 reload_covar, 79
 save, 79
 save_ER, 80
 save_covar, 79
 save_preferences, 80
 set_max_exposure, 80
 set_scale, 80
 simulate_till_age, 81
 user_name, 82
investor.class.cpp, 190
investor_id
 follow_up, 48
 investment_problem, 73
 investor, 81
 investor_ui, 97
 simulation, 112
investor_ui, 82
 account_form, 85
 account_save, 85

active_action, 96
 add_to_db, 85
 age, 85
 age2monthNbr, 85
 asset_add, 86
 asset_class_names, 96
 asset_delete, 86
 asset_edit, 86
 asset_save, 86
 assetClass_covar, 96
 assetClass_mu, 96
 assetClass_name, 96
 birth_date, 96
 button_to, 86
 cash_flow_add, 86
 cash_flow_delete, 86
 cash_flow_edit, 86
 cash_flow_list, 86
 cash_flow_save, 86
 covar, 96
 currency, 96
 dateStr2Age, 86
 desirability, 97
 ER, 97
 exists, 87
 experience, 97
 feedback, 87
 first_name, 97
 get_from_db, 87
 get_full_name, 87
 goal_add, 87
 goal_delete, 87
 goal_edit, 87
 goal_help_form, 87
 goal_help_save, 87
 goal_list, 88
 goal_save, 88
 hh, 97
 home_screen, 88
 investor_id, 97
 investor_ui, 85
 investor_ui, 85
 is_logged_in, 88
 knowledge, 97
 last_name, 97
 load_ER, 88
 load_covar, 88
 load_from_db, 88
 load_preferences, 89
 login, 89
 login_exec, 89
 login_form, 89
 logout, 89
 max_exposure, 97
 months2simulate, 89
 parse_bm, 89
 parse_form_close_personalize, 89
 parse_form_open_personalize, 89
 parse_goal_box, 90
 parse_savings_plan, 90
 parse_simulation_duration_form, 90
 parse_simulation_insight, 90
 parse_user_info, 90
 password, 97
 personalize_ER_table, 90
 personalize_ERvol_table, 90
 personalize_covar_table, 90
 personalize_form, 90
 personalize_pref_table, 91
 personalize_reload, 91
 personalize_save, 91
 personalize_screen, 91
 personalize_set_ER, 91
 personalize_set_corr, 91
 personalize_set_general, 91
 personalize_set_prefs, 91
 personalize_set_vol, 91
 personalize_vol_table, 92
 register_exec, 92
 register_form, 92
 reload_ER, 92
 reload_covar, 92
 save, 92
 save_ER, 93
 save_covar, 93
 save_preferences, 93
 set_birth_date, 93
 set_currency, 93
 set_first_name, 93
 set_investor_id, 93
 set_investor_id_fromEnv, 93
 set_last_name, 93
 set_max_exposure, 93
 set_password, 94
 set_scale, 94
 set_simulate_till_age, 94
 set_user_name, 94
 show_account_form_body, 94
 show_disclaimer, 94
 show_follow_up, 94
 show_goal_help_form_body, 95
 show_home_screen_form, 95
 show_inventory_form, 95
 show_inventory_form_body, 95
 show_login_form_body, 95
 show_register_form_body, 95
 show_simulation, 95
 show_total_portfolio, 95
 simulate_till_age, 97
 update_last_activity_at, 96
 user_name, 97
 investor_ui.class.cpp, 194
 is_logged_in
 investor_ui, 88
 is_valid_dateStr
 global_functions.h, 156

- javaGraph
 - follow_up, 43
 - investment_problem, 67
 - simulation, 107
- knowledge
 - follow_up, 48
 - investment_problem, 73
 - investor, 81
 - investor_ui, 97
 - simulation, 112
- LOCAL
 - config.class.cpp, 137
- LOW
 - normdist.h, 227
- label
 - column, 33
- language
 - config, 35
- last_name
 - follow_up, 48
 - investment_problem, 73
 - investor, 81
 - investor_ui, 97
 - simulation, 112
- layout
 - config, 35
- list
 - bona, 24
 - cash_flow, 27
 - goal, 51
- list_assets
 - asset, 20
- load
 - asset, 20
- load_ER
 - follow_up, 43
 - investment_problem, 67
 - investor, 78
 - investor_ui, 88
 - market, 99
 - simulation, 107
- load_covar
 - follow_up, 43
 - investment_problem, 67
 - investor, 78
 - investor_ui, 88
 - market, 99
 - simulation, 107
- load_from_db
 - follow_up, 43
 - investment_problem, 67
 - investor, 78
 - investor_ui, 88
 - simulation, 107
- load_fromDB
 - bona, 24
 - cash_flow, 27
 - goal, 51
- load_fromEnv
 - bona, 24
 - cash_flow, 27
 - goal, 51
- load_fromRecordSet
 - bona, 24
- load_preferences
 - follow_up, 44
 - investment_problem, 67
 - investor, 79
 - investor_ui, 89
 - simulation, 107
- login
 - investor_ui, 89
- login_exec
 - investor_ui, 89
- login_form
 - investor_ui, 89
- logout
 - investor_ui, 89
- MAX_AGE
 - efp.cpp, 145
- MAX_ALPHA
 - efp.cpp, 145
- MAX_MNTHS_2_SIMULATE
 - efp.cpp, 145
- MAX_SCALE
 - efp.cpp, 146
- MIN_ALPHA
 - efp.cpp, 146
- MIN_MNTHS_2_SIMULATE
 - efp.cpp, 146
- MIN_SCALE
 - efp.cpp, 146
- mNbr2dateStr
 - global_functions.h, 156
- main
 - efp.cpp, 148
- market, 98
 - assetClass_covar, 100
 - assetClass_mu, 100
 - assetClass_name, 100
 - covar, 100
 - ER, 100
 - load_ER, 99
 - load_covar, 99
 - market, 99
 - reload_ER, 99
 - reload_covar, 99
 - save_ER, 100
 - save_covar, 100
- market.class.cpp, 223
- market_return
 - follow_up, 48
 - simulation, 112
- max_exposure
 - follow_up, 48

- investment_problem, 73
- investor, 81
- investor_ui, 97
- simulation, 112
- max_shortfall
 - aGOAL, 17
- means
 - investment_problem, 73
- means_block
 - investment_problem, 73
- means_goal
 - aGOAL, 17
- means_left
 - investment_problem, 74
- means_tmp
 - investment_problem, 74
- means_used
 - investment_problem, 74
- merge_ER_vol
 - config, 35
- months2simulate
 - follow_up, 44
 - investment_problem, 68
 - investor, 79
 - investor_ui, 89
 - simulation, 107
- months2yms
 - global_functions.h, 156
- multiplier
 - cls_frequency, 30
- NBR_ASSET_CLASSES
 - efp.cpp, 146
- NBR_ITERATIONS
 - efp.cpp, 146
- NBR_RIGHT_MENU
 - efp.cpp, 146
- NBR_SCALE
 - efp.cpp, 147
- NBR_SUB_TABS
 - efp.cpp, 147
- NBR_TABS
 - efp.cpp, 147
- navbar_box
 - html_helper, 56
- navbar_end
 - html_helper, 56
- navbar_item
 - html_helper, 56
- navbar_start
 - html_helper, 57
- nbr_assets
 - investment_problem, 74
- nbr_cfs
 - investment_problem, 74
- nbr_goals
 - investment_problem, 74
- nbr_portfolios
 - investment_problem, 74
- nbrMonths
 - follow_up, 49
 - simulation, 113
- next_step
 - html_helper, 57
- normdist.h, 227
 - a, 229
 - b, 229
 - c, 229
 - d, 229
 - erfinv, 228
 - fast_erfinv, 228
 - HIGH, 227
 - LOW, 227
 - norminv, 228
 - phi, 228
- norminv
 - normdist.h, 228
- oAsset
 - efp.cpp, 149
- oAssetClass
 - efp.cpp, 149
- oCF
 - efp.cpp, 149
- oConfig
 - efp.cpp, 149
- oCurrency
 - efp.cpp, 150
- oGoal
 - efp.cpp, 150
- oInvestor
 - efp.cpp, 150
- operator<<
 - aGOAL, 16
 - aGOAL.struct.cpp, 115
- optimal_portf
 - aGOAL, 17
- order_by
 - bona, 25
 - cash_flow, 28
 - goal, 52
- pLogR
 - follow_up, 49
 - investment_problem, 74
 - portfolio, 102
 - simulation, 113
- pMu
 - follow_up, 49
 - investment_problem, 75
 - portfolio, 102
 - simulation, 113
- PRECISION
 - efp.cpp, 147
- pSigma
 - follow_up, 49
 - investment_problem, 75
 - portfolio, 102

- simulation, 113
- parse_bm
 - investor_ui, 89
- parse_breadcrumb
 - html_helper, 57
- parse_dashboard
 - follow_up, 44
- parse_date_time
 - html_helper, 57
- parse_form_close_personalize
 - investor_ui, 89
- parse_form_open_personalize
 - investor_ui, 89
- parse_goal_box
 - investor_ui, 90
- parse_javaGraph
 - follow_up, 44
 - investment_problem, 68
 - simulation, 107
- parse_savings_plan
 - investor_ui, 90
- parse_simulation_duration_form
 - investor_ui, 90
- parse_simulation_insight
 - investor_ui, 90
- parse_title
 - html_helper, 57
- parse_user_info
 - investor_ui, 90
- parse_wrapper
 - config, 35
- password
 - follow_up, 49
 - investment_problem, 74
 - investor, 81
 - investor_ui, 97
 - simulation, 113
- personalize_ER_table
 - investor_ui, 90
- personalize_ERvol_table
 - investor_ui, 90
- personalize_covar_table
 - investor_ui, 90
- personalize_form
 - investor_ui, 90
- personalize_pref_table
 - investor_ui, 91
- personalize_reload
 - investor_ui, 91
- personalize_save
 - investor_ui, 91
- personalize_screen
 - investor_ui, 91
- personalize_set_ER
 - investor_ui, 91
- personalize_set_corr
 - investor_ui, 91
- personalize_set_general
 - investor_ui, 91
- personalize_set_prefs
 - investor_ui, 91
- personalize_set_vol
 - investor_ui, 91
- personalize_vol_table
 - investor_ui, 92
- phi
 - normdist.h, 228
- Pi
 - global_functions.h, 157
- placeholder
 - column, 33
- plot_market_evol
 - follow_up, 44
 - simulation, 108
- portfolio, 100
 - description, 102
 - get_description, 101
 - get_portf_from_db, 101
 - pLogR, 102
 - pMu, 102
 - pSigma, 102
 - portfolio_id, 102
 - set_mu, 101
 - set_sigma, 102
 - weights, 102
- portfolio.class.cpp, 231
- portfolio_id
 - follow_up, 49
 - investment_problem, 75
 - portfolio, 102
 - simulation, 113
- portfolio_suitable
 - investment_problem, 75
- portfolios
 - follow_up, 49
 - investment_problem, 75
 - simulation, 113
- post_code
 - column, 33
- prepare_javaVars_colors
 - follow_up, 44
 - investment_problem, 68
 - simulation, 108
- priority
 - aGOAL, 18
- priorityMax
 - investment_problem, 75
- priorityMin
 - investment_problem, 75
- privacy_policy
 - html_helper, 57
- pstmt
 - db_helper, 38
- quote
 - column, 31

RAINY_DAY_MONTH_NBR
 efp.cpp, 147
 RETIREMENT_AGE
 efp.cpp, 147
 ready
 test.js, 240
 realization_age
 aGOAL, 18
 realization_monthNbr
 aGOAL, 18
 realized_shortfall
 aGOAL, 18
 register_exec
 investor_ui, 92
 register_form
 investor_ui, 92
 reload_ER
 follow_up, 45
 investment_problem, 68
 investor, 79
 investor_ui, 92
 market, 99
 simulation, 108
 reload_covar
 follow_up, 45
 investment_problem, 68
 investor, 79
 investor_ui, 92
 market, 99
 simulation, 108
 remarks
 aGOAL, 18
 res
 db_helper, 39
 res2
 db_helper, 39
 riskFunction
 config, 36
 round2str
 global_functions.h, 156
 runSQL
 db_helper, 38

 s_exp
 aGOAL, 18
 s_get
 bona, 24
 s_goal
 aGOAL, 18
 s_high
 aGOAL, 18
 s_low
 aGOAL, 18
 SECONDS_IN_MONTH
 efp.cpp, 147
 STEPDEF
 html_helper, 54
 STRMAP
 html_helper, 54

 sValue
 column, 33
 save
 aGOAL, 15
 asset, 20
 bona, 24
 cash_flow, 27
 follow_up, 45
 goal, 51
 investment_problem, 69
 investor, 79
 investor_ui, 92
 simulation, 109
 save_ER
 follow_up, 45
 investment_problem, 69
 investor, 80
 investor_ui, 93
 market, 100
 simulation, 109
 save_covar
 follow_up, 45
 investment_problem, 69
 investor, 79
 investor_ui, 93
 market, 100
 simulation, 109
 save_preferences
 follow_up, 45
 investment_problem, 69
 investor, 80
 investor_ui, 93
 simulation, 109
 save_results
 aGOAL, 15
 saving_plan
 aGOAL, 18
 scale125i2s
 html_helper, 59
 scripts_dir
 config, 36
 seriesColorsGoalPlot
 html_helper, 59
 set_amount
 aGOAL, 15
 asset, 20
 set_asset_class
 asset, 20
 set_asset_id
 asset, 20
 set_assets
 investment_problem, 69
 set_birth_date
 investor_ui, 93
 set_cfs
 investment_problem, 69
 set_curr
 cls_currency, 29

set_currency
 aGOAL, 15
 asset, 21
 investor_ui, 93

set_description
 aGOAL, 15
 asset, 21

set_first_name
 investor_ui, 93

set_followup_color_post
 follow_up, 46
 investment_problem, 69
 simulation, 109

set_followup_color_prae
 follow_up, 46
 investment_problem, 70
 simulation, 109

set_freq
 cls_frequency, 30

set_frequency
 aGOAL, 15

set_from_date
 aGOAL, 15

set_from_db
 aGOAL, 15

set_goal_color
 investment_problem, 70

set_goal_remarks
 investment_problem, 70

set_goal_type
 aGOAL, 15

set_goals
 investment_problem, 70

set_investor
 asset, 21

set_investor_id
 investor_ui, 93

set_investor_id_fromEnv
 investor_ui, 93

set_last_name
 investor_ui, 93

set_max_exposure
 follow_up, 46
 investment_problem, 70
 investor, 80
 investor_ui, 93
 simulation, 109

set_max_shortfall
 aGOAL, 15

set_means_goal
 investment_problem, 70

set_mu
 follow_up, 46
 investment_problem, 71
 portfolio, 101
 simulation, 110

set_nbrMonths2simulate_from_env
 simulation, 110

set_password
 investor_ui, 94

set_portfolios
 investment_problem, 71

set_priority
 aGOAL, 15

set_priorityLimits
 investment_problem, 71

set_realization_age
 aGOAL, 16

set_scale
 follow_up, 46
 investment_problem, 71
 investor, 80
 investor_ui, 94
 simulation, 110

set_sigma
 follow_up, 46
 investment_problem, 71
 portfolio, 102
 simulation, 110

set_simulate_till_age
 investor_ui, 94

set_till_date
 aGOAL, 16

set_to_pstmt
 aGOAL, 16

set_unallocated_goal
 investment_problem, 71

set_user_name
 investor_ui, 94

set_value
 column, 32

show
 bona, 24
 cash_flow, 27
 column, 32
 goal, 51

show_account_form_body
 investor_ui, 94

show_args
 html_helper, 57

show_aside
 config, 36

show_disclaimer
 investor_ui, 94

show_edit_form
 asset, 21
 bona, 25
 cash_flow, 27
 goal, 51

show_follow_up
 investor_ui, 94

show_goal_help_form_body
 investor_ui, 95

show_graph_till_MNbr
 aGOAL, 16

show_home

- html_helper, 57
- show_home_screen_form
 - investor_ui, 95
- show_inputField
 - column, 32
- show_inventory_form
 - investor_ui, 95
- show_inventory_form_body
 - investor_ui, 95
- show_login_form_body
 - investor_ui, 95
- show_register_form_body
 - investor_ui, 95
- show_simulation
 - follow_up, 46
 - investor_ui, 95
 - simulation, 110
- show_social
 - config, 36
 - html_helper, 58
- show_tc
 - bona, 25
- show_th
 - bona, 25
- show_toolBar
 - html_helper, 58
- show_total_portfolio
 - investor_ui, 95
- show_with_previous
 - column, 33
- simul_show_table
 - config, 36
- simulate
 - follow_up, 46
 - simulation, 110
- simulate_market
 - simulation, 110
- simulate_portfolios
 - simulation, 110
- simulate_till_age
 - follow_up, 49
 - investment_problem, 75
 - investor, 81
 - investor_ui, 97
 - simulation, 113
- simulation, 102
 - add_to_db, 105
 - age, 105
 - age2monthNbr, 105
 - assetClass_covar, 111
 - assetClass_mu, 111
 - assetClass_name, 111
 - birth_date, 111
 - covar, 111
 - currency, 111
 - dateStr2Age, 105
 - description, 112
 - desirability, 112
 - ER, 112
 - exists, 106
 - experience, 112
 - first_name, 112
 - get_description, 106
 - get_from_db, 106
 - get_full_name, 106
 - get_max_month_for_lower_goals, 106
 - get_nbr_goals, 106
 - get_nbrMonths2simulate, 106
 - get_portf_from_db, 106
 - goalZ, 112
 - investor_id, 112
 - javaGraph, 107
 - knowledge, 112
 - last_name, 112
 - load_ER, 107
 - load_covar, 107
 - load_from_db, 107
 - load_preferences, 107
 - market_return, 112
 - max_exposure, 112
 - months2simulate, 107
 - nbrMonths, 113
 - pLogR, 113
 - pMu, 113
 - pSigma, 113
 - parse_javaGraph, 107
 - password, 113
 - plot_market_evol, 108
 - portfolio_id, 113
 - portfolios, 113
 - prepare_javaVars_colors, 108
 - reload_ER, 108
 - reload_covar, 108
 - save, 109
 - save_ER, 109
 - save_covar, 109
 - save_preferences, 109
 - set_followup_color_post, 109
 - set_followup_color_prae, 109
 - set_max_exposure, 109
 - set_mu, 110
 - set_nbrMonths2simulate_from_env, 110
 - set_scale, 110
 - set_sigma, 110
 - show_simulation, 110
 - simulate, 110
 - simulate_market, 110
 - simulate_portfolios, 110
 - simulate_till_age, 113
 - simulation, 105
 - solve, 110
 - user_name, 113
 - weights, 113
- simulation.class.cpp, 233
- simulation_endvalue
 - aGOAL, 18

simulation_string
 aGOAL, 19

soft_dir
 config, 36

soft_file
 config, 36

soft_name
 config, 36

soft_server_url
 config, 36

soft_url
 config, 36

soft_version
 config, 37

solve
 follow_up, 47
 investment_problem, 71
 simulation, 110

std_message
 efp.cpp, 150

step_nbrs
 html_helper, 59

steps
 html_helper, 59

stmt
 db_helper, 39

sub_step_exists
 html_helper, 58

sub_step_nbrs
 html_helper, 59

sub_steps
 html_helper, 59

success
 aGOAL, 19

t_Asset_Class
 texts_Eng_UK.h, 243

t_Desirability
 t_personalize_Eng_UK.h, 238

t_Experience
 t_personalize_Eng_UK.h, 238

t_Knowledge
 t_personalize_Eng_UK.h, 238

t_Login
 texts_Eng_UK.h, 247

t_Max_Exp
 t_personalize_Eng_UK.h, 238

t_Register
 texts_Eng_UK.h, 247

t_after_title
 texts_Eng_UK.h, 242

t_and
 texts_Eng_UK.h, 242

t_aside
 texts_Eng_UK.h, 243

t_author
 texts_Eng_UK.h, 243

t_b_currency
 texts_Eng_UK.h, 243

t_b_date
 texts_Eng_UK.h, 243

t_before_list
 texts_Eng_UK.h, 243

t_breadcrumb
 texts_Eng_UK.h, 243

t_cancel
 texts_Eng_UK.h, 244

t_clear_form
 texts_Eng_UK.h, 244

t_collapse
 texts_Eng_UK.h, 244

t_currency
 texts_Eng_UK.h, 244

t_dashboard
 texts_Eng_UK.h, 244

t_delete
 texts_Eng_UK.h, 244

t_disclaimer
 t_disclaimer_Eng_UK.h, 236

t_disclaimer_Eng_UK.h, 236
 t_disclaimer, 236

t_edit
 texts_Eng_UK.h, 244

t_errMsg
 texts_Eng_UK.h, 244

t_extra_info
 texts_Eng_UK.h, 244

t_extraInfo
 texts_Eng_UK.h, 244

t_feedback
 texts_Eng_UK.h, 246

t_feedback_post
 texts_Eng_UK.h, 246

t_hide_extra_info
 texts_Eng_UK.h, 246

t_info_bdate
 t_personalize_Eng_UK.h, 238

t_info_curr
 t_personalize_Eng_UK.h, 238

t_info_simtill
 t_personalize_Eng_UK.h, 238

t_month
 texts_Eng_UK.h, 247

t_months
 texts_Eng_UK.h, 247

t_next_step
 texts_Eng_UK.h, 247

t_personalize
 t_personalize_Eng_UK.h, 238

t_personalize1
 t_personalize_Eng_UK.h, 238

t_personalize_Eng_UK.h, 237
 init_personalize, 238
 t_Desirability, 238
 t_Experience, 238
 t_Knowledge, 238
 t_Max_Exp, 238

- t_info_bdate, 238
- t_info_curr, 238
- t_info_simtill, 238
- t_personalize, 238
- t_personalize1, 238
- t_personalize_concepts, 238
- t_personalize_scoring, 239
- t_reload, 239
- t_personalize_concepts
 - t_personalize_Eng_UK.h, 238
- t_personalize_scoring
 - t_personalize_Eng_UK.h, 239
- t_plotlabels_evol
 - texts_Eng_UK.h, 247
- t_prev_step
 - texts_Eng_UK.h, 247
- t_regFrm
 - texts_Eng_UK.h, 247
- t_reload
 - t_personalize_Eng_UK.h, 239
- t_reset
 - texts_Eng_UK.h, 248
- t_save
 - texts_Eng_UK.h, 248
- t_scale125i2s
 - texts_Eng_UK.h, 248
- t_show_extra_info
 - texts_Eng_UK.h, 248
- t_showhide_extra_info
 - texts_Eng_UK.h, 248
- t_sim_till
 - texts_Eng_UK.h, 248
- t_simulation_duration
 - texts_Eng_UK.h, 248
- t_title
 - texts_Eng_UK.h, 248
- t_titles
 - texts_Eng_UK.h, 248
- t_warning
 - texts_Eng_UK.h, 249
- t_year
 - texts_Eng_UK.h, 249
- t_years
 - texts_Eng_UK.h, 249
- target_amount
 - aGOAL, 19
- tbl
 - bona, 25
 - cash_flow, 28
 - goal, 52
- tbl_name
 - bona, 25
 - cash_flow, 28
 - goal, 52
- tbl_prefix
 - config, 37
- terms_of_use
 - html_helper, 58
- test.js, 240
 - ready, 240
- texts_Eng_UK.h, 241
 - t_Asset_Class, 243
 - t_Login, 247
 - t_Register, 247
 - t_after_title, 242
 - t_and, 242
 - t_aside, 243
 - t_author, 243
 - t_b_currency, 243
 - t_b_date, 243
 - t_before_list, 243
 - t_breadcrumb, 243
 - t_cancel, 244
 - t_clear_form, 244
 - t_collapse, 244
 - t_currency, 244
 - t_dashboard, 244
 - t_delete, 244
 - t_edit, 244
 - t_errMsg, 244
 - t_extra_info, 244
 - t_extraInfo, 244
 - t_feedback, 246
 - t_feedback_post, 246
 - t_hide_extra_info, 246
 - t_month, 247
 - t_months, 247
 - t_next_step, 247
 - t_plotlabels_evol, 247
 - t_prev_step, 247
 - t_regFrm, 247
 - t_reset, 248
 - t_save, 248
 - t_scale125i2s, 248
 - t_show_extra_info, 248
 - t_showhide_extra_info, 248
 - t_sim_till, 248
 - t_simulation_duration, 248
 - t_title, 248
 - t_titles, 248
 - t_warning, 249
 - t_year, 249
 - t_years, 249
- thousand_separator
 - global_functions.h, 156
- till_date
 - aGOAL, 19
- tm2DateStr
 - global_functions.h, 156
- tm2Mnbr
 - global_functions.h, 156
- to_curr
 - cls_currency, 29
- total_portf_monthNbrs
 - config, 37
- update_last_activity_at

- investor_ui, [96](#)
- user_name
 - follow_up, [49](#)
 - investment_problem, [75](#)
 - investor, [82](#)
 - investor_ui, [97](#)
 - simulation, [113](#)
- verify
 - bona, [25](#)
- weights
 - follow_up, [49](#)
 - investment_problem, [75](#)
 - portfolio, [102](#)
 - simulation, [113](#)
- wrapper_name
 - config, [37](#)
- wrapper_url
 - config, [37](#)
- www_dir
 - config, [37](#)