Rodoc

Generated by Doxygen 1.9.7

Chapter 1

Rodoc

This is the main page for documentation of the program Rodoc.

The code is separated mainly into two parts:

1. Core
2. Graphics

Core is for working with the family tree and also for parsing files.

Graphics is just the graphical interface of the application.

This application uses `Jsoncpp library`, `Qt graphics library` with version 6.

Database is a group of `json` files and directories. It follows this predefined structure (# is for a comment):

```
[rootDir]          # Chosen by the user.
  - Persons.json   # All persons.
  - Config.json    # Configuration and templates.
  - Events.json    # Events.
  - Relations.json # Relations
  - Files.json     # Files metadata.
  - Media.json     # Media metadata.
  - Notes.json     # Notes metadata.
  - Media          # Directory for media.
    - [files]
  - Notes          # Directory for notes.
    - [file]
  - Files          # Directory for other files.
    - [files]
```

Also `Files`, `Notes` and `Media` are internally implemented the same way. The main difference is for the end user.

Chapter 2

icons

Převzaté z `link`.

Hlavní stránka je `tady`.

Chapter 3

Events

Event entity allows to describe a particular event and associate it with the following pieces of information:

- **Date** (e.g., *20. 7. 2023*)
- **Place** (e.g., *Prague, Czech Republic*)
- **List of an arbitrary number of persons** acting as participants

In order to make the concept of events more generic and not limit ourselves to certain hard-wired event types only, each event is derived from a particular **event template**.

3.1 Event Templates

Each template has the following characteristics:

- **Title** (e.g., *Wedding*)
- **Description** (e.g., *A social event mainly between soon to be wife and husband.*)
- Flag activating a **date** (e.g., *enabled*)
- Flag activating a **place** (e.g., *disabled*)
- Arbitrary number of **roles** allowing to distinguish individual **participating persons** (e.g., *groom*, *bride*, ...)
- Flag allowing **additional participating persons** without any roles assigned

3.2 Predefined Templates

Event templates must be created first before they can be used. While it is possible to introduce specific user-defined templates, several widely used templates were already predefined and are available for import:

- **Wedding**
- **Divorce**
- **Funeral**
- **Graduation**

Chapter 4

Files, Notes & Media

Each person can have an unlimited number of **files**, **media**, and **notes**. To organize files easily, every **person** has a separate virtual directory for each file type. When a file is already added to the database, it can be **shared** among other persons without making **duplicates**.

4.1 Virtual Directories

Every virtual directory can be organized individually, just like in every file manager.

- **Renaming** a file or a directory. When renaming a file, it will be renamed for all persons, but the names of directories are not shared
- **Deleting** a file or a directory
- **Creating** a new directory
- **Adding** new files

Keep in mind that all the files need to be in a directory, thus there is always the **root** directory that cannot be deleted, but it can be renamed.

4.2 Adding Files

We have three options for adding files:

1. Choosing a file that is **already in the database**
2. Making a **copy** of an already existing file from a computer
3. Creating a completely **new and empty file**

To keep the project internally consistent and clean, files are **permanently removed** upon saving the project if it happens that no **person** has access to them.

Chapter 5

Welcome to rodoc

This is a brief introduction to this application for organizing family trees. Each **family tree** is saved and organized in a **project**.

5.1 Project

Each **project** is stored in a specific directory on a filesystem, assuming that the entire subtree of the project root directory should not be used for different purposes.

Various kinds of entities can be used throughout the project to express all the intended information related not just to persons and their relations.

5.2 Persons

Each family tree primarily contains a list of **persons**, each having basic information about themselves, such as a **name**, **surname**, etc. Each person can also have multiple user-defined **tags**.

5.3 Relations

To bind two particular **persons** together, we use the concept of a **relation** to record their real-world interconnection.

5.4 Events

Persons may also attend various types of events, such as weddings, allowing these to be associated with a **place**, **date**, as well as allowing the persons to participate in specific roles.

5.5 Files

Finally, each **person** can have an unlimited number of physical files of arbitrary types, logically divided into three groups of **files**, **media**, and **notes**.

Chapter 6

Layout

The **main window** has its **menu** and two separate **panels**.

6.1 Menu

The menu is for using the app functions.

- Changing the **settings**
- Saving and opening **projects**
- Editing **templates**
- Making **exports**
- **Importing** data

6.2 Panels

The left panel displays the information regarding the active person or so-called **proband**, while the right panel serves for the **family tree** visualization.

6.2.1 Left Panel

The left panel contains the name of the **proband** at the top, and next to that options to create a new person or delete the current one. Under that, there are several tabs:

- **General** serves for changing the default **proband** information and **tags**
- Tabs **relations**, **events**, **files**, **media**, and **notes** permits organizing the same-named entities

6.2.2 Right Panel

The right panel is also internally organized into three tabs where every view is interactive.

1. **Project overview** is a list of all **entities**, which can be filtered.
2. **Close family** is a diagram of all persons in relation to the **proband**. These persons are separated into three generations.
3. **Tree view** is a diagram of the **family tree** showing the so-called *hourglass* shape diagram. The number of generations shown can be configured, it is also possible to zoom in and out the view.

Chapter 7

Person

Each **person** has a set of values that can be edited:

- **Titles** before and after a full name (e.g., *M.Sc.* or *Ph.D.*)
- **Name** (e.g., *John*)
- **Surname** (e.g., *Swanson*)
- **Birthname** (e.g., *Smith*)
- **Gender** (e.g., *Male*)
- **Date of birth** (e.g., *23. 2. 2000*)
- **Place of birth** (e.g., *Prague*)
- Flag for an **alive** person, otherwise a **date** and **place of death**

7.1 Dates

Every **date** value can actually be:

- **Unknown** as a special value for missing information
- Just a **single date** (e.g., *23. 2. 2000*)
- **Range** of two dates (e.g., *From: 23. 2. 2000 to: 25. 2. 2000*)

Each date can also be associated with an optional **note** allowing to hold any information intended.

7.2 Tags

Tag is a pair of a **name** and its **value**. Each person can be associated with an arbitrary number of such tags. However, they need to be unique. Whenever a new tag name is used for the first time, it becomes globally available within the entire project.

Chapter 8

Relations

Relation entity allows to bind two particular persons together to represent their real-world relationship. Each relation has these two pieces of information:

- **First person** (e.g., *John Swanson*)
- **Second person** (e.g., *Roy Swanson*)

In order not to be restrained by a hard-wired list of permitted relations, the concept of a **relation template** was introduced.

8.1 Relation Templates

Each template has the following characteristics:

- **Title** (e.g., *Fatherhood*)
- **Description** (e.g., *Relation between a father and a child*)
- Name of the **first person** (e.g., *father*)
- Name of the **second person** (e.g., *child*)
- **Generation difference** (e.g., *-1*)
- **Trait** (e.g., *Fatherhood*)

8.1.1 Generation Difference

Generation difference describes which person is older and which one is younger. Any number can be possibly provided, but only **-1**, **0**, and **1** usually make sense.

- **-1** means that the first person is older,
- **0** means that both persons are in the same generation
- **1** means that the first person is younger.

8.1.2 Trait

Trait determines how a particular **relation** is intended to be interpretted. The given list of possibilities shadows the real world.

- **No special trait** (e.g., *Uncle-Nephew*)
- **Motherhood** (e.g., *Mother-Daughter*)
- **Fatherhood** (e.g., *Father-Son*)
- **Partnership** (e.g., *Marriage*)
- **Sibling** (e.g., *Brother-Sister*)

8.1.3 Predefined Templates

Relation templates must be created first before they can be used. While it is possible to introduce specific user-defined templates, several widely used templates were already predefined and are available for import:

- **Fatherhood**
- **Motherhood**
- **Adoptive motherhood**
- **Adoptive fatherhood**
- **Partnership**
- **Marriage**
- **Brotherhood**
- **Sisterhood**
- **Siblings**

8.2 Promotion

When a **person** has several **relations** of the same and supported **trait** (e.g., *Father*, *Mother*, *Partner*, ...), one of them can be declared as **promoted**, i.e., considered as the main one.

8.3 Suggestions

After a new **relation** is created, several **suggestions** for additional plausible relations that can be inferred are automatically offered in order to smooth the entire process.

- **Partnership relation** -> share children between both of them
- **Sibling relation** -> share parents
- **Parent relation** -> children are bound as siblings, partners as parents, and vice versa

Chapter 9

Namespace Index

9.1 Namespace List

Here is a list of all namespaces with brief descriptions:

Chapter 10

Hierarchical Index

10.1 Class Hierarchy

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

Chapter 11

Class Index

11.1 Class List

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

Chapter 12

File Index

12.1 File List

Here is a list of all files with brief descriptions:

Chapter 13

Namespace Documentation

13.1 addfile Namespace Reference

Namespace for strings shown in add file dialog.

Variables

- const std::string NONE = ¨None¨

    String for no extension.

13.1.1 Detailed Description

Namespace for strings shown in add file dialog.

13.1.2 Variable Documentation

13.1.2.1 NONE

```
const std::string addfile::NONE = ¨None¨
```

String for no extension.

13.2 button Namespace Reference

Namespace for general texts on buttons.

Variables

- const std::string ADD_NEW = ¨Add new¨

  Text for adding new element.
- const std::string CANCEL = ¨Cancel¨

  Text for cancelling.
- const std::string REMOVE = ¨Remove¨

  text for removing element.

13.2.1 Detailed Description

Namespace for general texts on buttons.

13.2.2 Variable Documentation

13.2.2.1 ADD_NEW

```
const std::string button::ADD_NEW = ¨Add new¨
```

Text for adding new element.

13.2.2.2 CANCEL

```
const std::string button::CANCEL = ¨Cancel¨
```

Text for cancelling.

13.2.2.3 REMOVE

```
const std::string button::REMOVE = ¨Remove¨
```

text for removing element.

13.3 error Namespace Reference

Namespace for errors and exceptions messages.

Variables

- const std::string UNKNOWN_EXCEP_STR = ¨Unknown exception catch.¨

  String for catching unknown exception.
- const std::string EXCEPTION_STR = ¨Exception: ¨

  String for adding before exception description.
- const std::string CORRUPTED_DATABASE = ¨Given database is corrupted: ¨

  String shown when database is corrupted.
- const std::string EMPTY_BOXES = ¨All boxes must not be empty.¨

  String for showing that some boxes were left empty.
- const std::string NO_CHOSEN_FILE = ¨No file was chosen.¨

  No file chosen and empty box.
- const std::string NO_ROLE = ¨Given role 'no role' cannot be used. It would be considered as no special role.¨

  When person tries to add ¨no role¨ role, which is allocated for no roles.
- const std::string FORBIDDEN_CHARS = ¨Text cannot contain forbidden characters. &lt; &gt; \¨¨

  When in any box where inserted forbidden characters.

13.3.1 Detailed Description

Namespace for errors and exceptions messages.

13.3.2 Variable Documentation

13.3.2.1 CORRUPTED_DATABASE

```
const std::string error::CORRUPTED_DATABASE = ¨Given database is corrupted:  ¨
```

String shown when database is corrupted.

13.3.2.2 EMPTY_BOXES

```
const std::string error::EMPTY_BOXES = ¨All boxes must not be empty.¨
```

String for showing that some boxes were left empty.

13.3.2.3 EXCEPTION_STR

```
const std::string error::EXCEPTION_STR = ¨Exception:  ¨
```

String for adding before exception description.

13.3.2.4 FORBIDDEN_CHARS

```
const std::string error::FORBIDDEN_CHARS = ¨Text cannot contain forbidden characters.  &lt;
&gt; \¨¨
```

When in any box where inserted forbidden characters.

13.3.2.5 NO_CHOSEN_FILE

```
const std::string error::NO_CHOSEN_FILE = ¨No file was chosen.¨
```

No file chosen and empty box.

13.3.2.6 NO_ROLE

```
const std::string error::NO_ROLE = ¨Given role 'no role' cannot be used.  It would be considered
as no special role.¨
```

When person tries to add ¨no role¨ role, which is allocated for no roles.

13.3.2.7 UNKNOWN_EXCEP_STR

```
const std::string error::UNKNOWN_EXCEP_STR = ¨Unknown exception catch.¨
```

String for catching unknown exception.

13.4 events Namespace Reference

Namespace for strings for default events.

Variables

- const std::string WEDDING = ¨Wedding¨

  String for wedding.
- const std::string DESC_WEDDING = ¨A social event mainly between soon to be wife and husband.¨

  String for description of wedding.
- const std::string GROOM = ¨Groom¨

  String for groom.
- const std::string BRIDE = ¨Bride¨

  String for bride.
- const std::string BEST_MAN = ¨Best man¨

  String for best man.
- const std::string MAID_OF_HONOR = ¨Maid of Honor¨

  String for maid of honor.
- const std::string DIVORCE = ¨Divorce¨

  String for divorce.
- const std::string DESC_DIVORCE = ¨Divorce is the process of terminating a marriage or marital union.¨

  String for divorce description.
- const std::string FORMER_HUSBAND = ¨Former husband¨

  String for former husband.
- const std::string FORMER_WIFE = ¨Former wife¨

  String for former wife.
- const std::string FUNERAL = ¨Funeral¨

  String for funeral.

- const std::string DESC_FUNERAL = ¨A funeral is a ceremony connected with the final disposition of a corpse, such as a burial or cremation, with the attendant observances.¨

  String for funeral description.
- const std::string DEAD_PERSON = ¨Dead person¨

  String for dead person.
- const std::string GRADUATION = ¨Graduation¨

  String for graduation.
- const std::string DESC_GRADUATION = ¨Graduation is the awarding of a diploma to a student by an educational institution.¨

  String for graduation description.
- const std::string GRADUATE = ¨Graduate¨

  String for graduate.
- const std::string NO_ROLE = ¨no role¨

  String for no role in event.

13.4.1 Detailed Description

Namespace for strings for default events.

13.4.2 Variable Documentation

13.4.2.1 BEST_MAN

```
const std::string events::BEST_MAN = ¨Best man¨
```

String for best man.

13.4.2.2 BRIDE

```
const std::string events::BRIDE = ¨Bride¨
```

String for bride.

13.4.2.3 DEAD_PERSON

```
const std::string events::DEAD_PERSON = ¨Dead person¨
```

String for dead person.

13.4.2.4 DESC_DIVORCE

```
const std::string events::DESC_DIVORCE = ¨Divorce is the process of terminating a marriage or
marital union.¨
```

String for divorce description.

13.4.2.5 DESC_FUNERAL

```
const std::string events::DESC_FUNERAL = ¨A funeral is a ceremony connected with the final
disposition of a corpse, such as a burial or cremation, with the attendant observances.¨
```

String for funeral description.

13.4.2.6 DESC_GRADUATION

```
const std::string events::DESC_GRADUATION = ¨Graduation is the awarding of a diploma to a
student by an educational institution.¨
```

String for graduation description.

13.4.2.7 DESC_WEDDING

```
const std::string events::DESC_WEDDING = ¨A social event mainly between soon to be wife and
husband.¨
```

String for description of wedding.

13.4.2.8 DIVORCE

```
const std::string events::DIVORCE = ¨Divorce¨
```

String for divorce.

13.4.2.9 FORMER_HUSBAND

```
const std::string events::FORMER_HUSBAND = ¨Former husband¨
```

String for former husband.

13.4.2.10 FORMER_WIFE

```
const std::string events::FORMER_WIFE = ¨Former wife¨
```

String for former wife.

13.4.2.11 FUNERAL

```
const std::string events::FUNERAL = ¨Funeral¨
```

String for funeral.

13.4.2.12 GRADUATE

```
const std::string events::GRADUATE = ¨Graduate¨
```

String for graduate.

13.4.2.13 GRADUATION

```
const std::string events::GRADUATION = ¨Graduation¨
```

String for graduation.

13.4.2.14 GROOM

```
const std::string events::GROOM = ¨Groom¨
```

String for groom.

13.4.2.15 MAID_OF_HONOR

```
const std::string events::MAID_OF_HONOR = ¨Maid of Honor¨
```

String for maid of honor.

13.4.2.16 NO_ROLE

```
const std::string events::NO_ROLE = ¨no role¨
```

String for no role in event.

13.4.2.17 WEDDING

```
const std::string events::WEDDING = ¨Wedding¨
```

String for wedding.

13.5 gender Namespace Reference

Namespace for constants with gender enumeration.

Functions

- std::string str (Gender g)

Variables

- constexpr Gender AllGenders [ ] = {Other, Male, Female}

  All genders in a list.

13.5.1 Detailed Description

Namespace for constants with gender enumeration.

13.5.2 Function Documentation

13.5.2.1 str()

```
std::string gender::str (
             Gender g )
```

Write the gender as a string.

Parameters

g	Given Gender.

Returns

Newly constructed string representing the gender.

13.5.3 Variable Documentation

13.5.3.1 AllGenders

```
constexpr Gender gender::AllGenders[ ] = {Other, Male, Female}  [constexpr]
```

All genders in a list.

13.6 html Namespace Reference

Namespace for html tags and everything with html.

Variables

- const std::string RESOURCES = ¨resources¨

  Where will resources be saved.
- const std::string PREFIX_CLOSE = ¨/close-¨

  Prefix for close family diagram.
- const std::string PREFIX_FAMILY_TREE = ¨/tree-¨

Prefix for family tree diagram.

- const std::string CUSTOM_CSS = ¨styles.css¨

  Custom CSS file name.
- const std::string GENERAL_INFO = ¨General information¨

  General information.
- const std::string BIRTH_DATE = ¨Birth date: ¨

  Birth date.
- const std::string BIRTH_PLACE = ¨Birth place: ¨

  Birth place.
- const std::string DEATH_DATE = ¨Date of death: ¨

  Date of death.
- const std::string DEATH_PLACE = ¨Place of death: ¨

  Place of death.
- const std::string GENDER = ¨Gender: ¨

  Gender.
- const std::string CUSTOM_TAGS = ¨Custom tags:¨

  Custom tags.
- const std::string DESCRIPTION = ¨Description:¨

  Description.
- const std::string NOTE = ¨Note:¨

  Note.
- const std::string DATE = ¨Date: ¨

  Date.
- const std::string PLACE = ¨Place: ¨

  Place.
- const std::string PERSONS = ¨Persons:¨

  Persons.
- const std::string ALL_PERSONS = ¨All persons¨

  All persons.
- const std::string EVENTS = ¨Events¨

  Events.
- const std::string FAMILY_TREE = ¨Family tree¨

  Family tree.
- const std::string FAMILY_TREE_DIAGRAM = ¨Family tree diagram¨

  Family tree diagram.
- const std::string RELATIONS = ¨Relations¨

  Relations.
- const std::string CLOSE_FAMILY = ¨Close family¨

  Close family.
- const std::string CLOSE_FAMILY_DIAGRAM = ¨Close family diagram¨

  Close family diagram.

13.6.1 Detailed Description

Namespace for html tags and everything with html.

13.6.2 Variable Documentation

13.6.2.1 ALL_PERSONS

```
const std::string html::ALL_PERSONS = ¨All persons¨
```

All persons.

13.6.2.2 BIRTH_DATE

```
const std::string html::BIRTH_DATE = ¨Birth date:  ¨
```

Birth date.

13.6.2.3 BIRTH_PLACE

```
const std::string html::BIRTH_PLACE = ¨Birth place:  ¨
```

Birth place.

13.6.2.4 CLOSE_FAMILY

```
const std::string html::CLOSE_FAMILY = ¨Close family¨
```

Close family.

13.6.2.5 CLOSE_FAMILY_DIAGRAM

```
const std::string html::CLOSE_FAMILY_DIAGRAM = ¨Close family diagram¨
```

Close family diagram.

13.6.2.6 CUSTOM_CSS

```
const std::string html::CUSTOM_CSS = ¨styles.css¨
```

Custom CSS file name.

13.6.2.7 CUSTOM_TAGS

```
const std::string html::CUSTOM_TAGS = ¨Custom tags:¨
```

Custom tags.

13.6.2.8 DATE

```
const std::string html::DATE = ¨Date: ¨
```

Date.

13.6.2.9 DEATH_DATE

```
const std::string html::DEATH_DATE = ¨Date of death: ¨
```

Date of death.

13.6.2.10 DEATH_PLACE

```
const std::string html::DEATH_PLACE = ¨Place of death: ¨
```

Place of death.

13.6.2.11 DESCRIPTION

```
const std::string html::DESCRIPTION = ¨Description:¨
```

Description.

13.6.2.12 EVENTS

```
const std::string html::EVENTS = ¨Events¨
```

Events.

13.6.2.13 FAMILY_TREE

```
const std::string html::FAMILY_TREE = ¨Family tree¨
```

Family tree.

13.6.2.14 FAMILY_TREE_DIAGRAM

```
const std::string html::FAMILY_TREE_DIAGRAM = ¨Family tree diagram¨
```

Family tree diagram.

13.6.2.15 GENDER

```
const std::string html::GENDER = ¨Gender: ¨
```

Gender.

13.6.2.16 GENERAL_INFO

```
const std::string html::GENERAL_INFO = ¨General information¨
```

General information.

13.6.2.17 NOTE

```
const std::string html::NOTE = ¨Note:¨
```

Note.

13.6.2.18 PERSONS

```
const std::string html::PERSONS = ¨Persons:¨
```

Persons.

13.6.2.19 PLACE

```
const std::string html::PLACE = ¨Place:  ¨
```

Place.

13.6.2.20 PREFIX_CLOSE

```
const std::string html::PREFIX_CLOSE = ¨/close-¨
```

Prefix for close family diagram.

13.6.2.21 PREFIX_FAMILY_TREE

```
const std::string html::PREFIX_FAMILY_TREE = ¨/tree-¨
```

Prefix for family tree diagram.

13.6.2.22 RELATIONS

```
const std::string html::RELATIONS = ¨Relations¨
```

Relations.

13.6.2.23 RESOURCES

```
const std::string html::RESOURCES = ¨resources¨
```

Where will resources be saved.

13.7 jsonlabel Namespace Reference

Namespace for all JSON labels.

Variables

- const std::string TITLE = ¨title¨

  JSON Label for title.
- const std::string ID = ¨id¨

  JSON Label for id.
- const std::string DESCRIPTION = ¨description¨

  JSON Label for description.
- const std::string CONTAINS_DATE = ¨contains date¨

  JSON Label for if the event contains a date or not.
- const std::string MORE_PEOPLE = ¨more people¨

  JSON Label for if the event contains more people.
- const std::string SPECIAL_ROLES = ¨special roles¨

  JSON Label for if the event has special roles.
- const std::string NAME = ¨name¨

  JSON Label for name.
- const std::string FIRST_NAME = ¨first name¨

  JSON Label for first name.
- const std::string SECOND_NAME = ¨second name¨

  JSON Label for second name.
- const std::string TRAIT = ¨trait¨

  JSON Label for trait.
- const std::string GEN_DIFF = ¨generation difference¨

  JSON Label for generation difference.
- const std::string REL_TEMPL = ¨relation templates¨

  JSON Label for relations templates.
- const std::string EVE_TEMPL = ¨event templates¨

  JSON Label for event templates.
- const std::string TYPE = ¨type¨

  JSON Label for type.
- const std::string DATE = ¨date¨

  JSON Label for date.
- const std::string LAST_DATE = ¨last date¨

  JSON Label for last date.
- const std::string TEXT = ¨text¨

  JSON Label for text.
- const std::string TAGS = ¨tags¨

  JSON Label for tags.
- const std::string TAG = ¨tag¨

  JSON Label for single tag.
- const std::string RELATIONS = ¨relations¨

  JSON Label for relations.
- const std::string EVENTS = ¨events¨

  JSON Label for events.
- const std::string SURNAME = ¨surname¨

  JSON Label for surname.

- const std::string MAIDEN_NAME = ¨maiden name¨

  JSON Label for maiden name.
- const std::string VALUE = ¨value¨

  JSON Label for value.
- const std::string BIRTH = ¨birth¨

  JSON Label for birth.
- const std::string DEATH = ¨death¨

  JSON Label for death.
- const std::string VIRTUAL_DRIVE = ¨virtual drive¨

  JSON Label for virtual drive.
- const std::string MEDIA = ¨media¨

  JSON Label for media.
- const std::string NOTES = ¨notes¨

  JSON Label for notes.
- const std::string FILES = ¨files¨

  JSON Label for files.
- const std::string PERSONS = ¨persons¨

  JSON Label for persons.
- const std::string TEMPLATE = ¨template¨

  JSON Label for for template.
- const std::string TEMPLATES = ¨templates¨

  JSON Label for for templates.
- const std::string FIRST_PERSON = ¨first person¨

  JSON Label for first person.
- const std::string SECOND_PERSON = ¨second person¨

  JSON Label for second person.
- const std::string PLACE = ¨place¨

  JSON Label for place.
- const std::string ROLE = ¨role¨

  JSON Label for role.
- const std::string PERSON = ¨person¨

  JSON Label for person.
- const std::string GENDER = ¨gender¨

  JSON Label for gender.
- const std::string LIVES = ¨lives¨

  JSON Label for if the person lives or not.
- const std::string FRONT_TITLE = ¨title in front¨

  JSON Label for title in front of a name.
- const std::string AFTER_TITLE = ¨title after¨

  JSON Label for title after the name.
- const std::string FATHER = ¨father¨

  JSON Label for father.
- const std::string MOTHER = ¨mother¨

  JSON Label for mother.
- const std::string PARTNER = ¨partner¨

  JSON Label for partner.
- const std::string MOVEABLE_TABS = ¨moveable tabs¨

  JSON Label for setting moveable tabs.
- const std::string TAB_POSITION = ¨tab position¨

  JSON Label for tab position.
- const std::string GLOBAL_MAIN_PERSON = ¨global main person¨

JSON Label for global main person.

- const std::string CONTAINS_PLACE = ¨contains place¨

  JSON Label if an event template contains a place.
- const std::string SHOW_GENERAL_TAB = ¨show general tab¨

  JSON Label for app settings if general tab should be visible.
- const std::string SHOW_EVENT_TAB = ¨show event tab¨

  JSON Label for app settings if event tab should be visible.
- const std::string SHOW_FILE_TAB = ¨show file tab¨

  JSON Label for app settings if file tab should be visible.
- const std::string SHOW_MEDIA_TAB = ¨show media tab¨

  JSON Label for app settings if media tab should be visible.
- const std::string SHOW_NOTE_TAB = ¨show note tab¨

  JSON Label for app settings if note tab should be visible.
- const std::string SHOW_RELATION_TAB = ¨show relation tab¨

  JSON Label for app settings if relation tab should be visible.
- const std::string APP_SETTINGS = ¨app settings¨

  JSON Label for app settings.
- const std::string DRIVES = ¨drives¨

  JSON Label for drives.
- const std::string EXPORT = ¨export¨

  JSON Label for export.
- const std::string RED = ¨red¨

  JSON Label for red.
- const std::string BLUE = ¨blue¨

  JSON Label for blue.
- const std::string GREEN = ¨green¨

  JSON Label for green.
- const std::string FONT_FAMILY = ¨font family¨

  JSON Label for font family.
- const std::string FONT_SIZE = ¨font size¨

  JSON Label for font size.
- const std::string HIGHLIGHTED_COLOR = ¨highlighted color¨

  JSON Label highlighted color.
- const std::string PROBANT_COLOR = ¨proband color¨

  JSON Label for proband color.
- const std::string PROMOTED_COLOR = ¨promoted color¨

  JSON Label promoted color.
- const std::string STANDARD_COLOR = ¨standard color¨

  JSON Label standard color.
- const std::string X_SIZE = ¨x size¨

  JSON Label for size in the x axis.
- const std::string Y_SIZE = ¨y size¨

  JSON Label for size in the y axis.
- const std::string GEN_SIZE_UP = ¨number of generations up¨

  JSON Label for how many generations are being shown upwards.
- const std::string GEN_SIZE_DOWN = ¨number of generations down¨

  JSON Label for how many generations are being shown downwards.
- const std::string DIR = ¨dir¨

  JSON Label for directory.
- const std::string ROOT = ¨root¨

  JSON Label for root.

- const std::string HELP = ¨help¨

  JSON Label for help.
- const std::string SPLITTER_POSITION_ONE = ¨splitter position one¨

  JSON Label for splitter position one.
- const std::string SPLITTER_POSITION_TWO = ¨splitter position two¨

  JSON Label for splitter position two.
- const std::string TEXT_COLOR = ¨text color¨

  JSON Label for the color of text.
- const std::string LINE_COLOR = ¨line color¨

  JSON Label for the line color.
- const std::string LINE_WIDTH = ¨line width¨

  JSON Label for line width.
- const std::string BORDER_RADIUS = ¨radius¨

  JSON Label border radius.

13.7.1 Detailed Description

Namespace for all JSON labels.

13.7.2 Variable Documentation

13.7.2.1 AFTER_TITLE

```
const std::string jsonlabel::AFTER_TITLE = ¨title after¨
```

JSON Label for title after the name.

13.7.2.2 APP_SETTINGS

```
const std::string jsonlabel::APP_SETTINGS = ¨app settings¨
```

JSON Label for app settings.

13.7.2.3 BIRTH

```
const std::string jsonlabel::BIRTH = ¨birth¨
```

JSON Label for birth.

13.7.2.4 BLUE

```
const std::string jsonlabel::BLUE = ¨blue¨
```

JSON Label for blue.

13.7.2.5 BORDER_RADIUS

```
const std::string jsonlabel::BORDER_RADIUS = ¨radius¨
```

JSON Label border radius.

13.7.2.6 CONTAINS_DATE

```
const std::string jsonlabel::CONTAINS_DATE = ¨contains date¨
```

JSON Label for if the event contains a date or not.

13.7.2.7 CONTAINS_PLACE

```
const std::string jsonlabel::CONTAINS_PLACE = ¨contains place¨
```

JSON Label if an event template contains a place.

13.7.2.8 DATE

```
const std::string jsonlabel::DATE = ¨date¨
```

JSON Label for date.

13.7.2.9 DEATH

```
const std::string jsonlabel::DEATH = ¨death¨
```

JSON Label for death.

13.7.2.10 DESCRIPTION

```
const std::string jsonlabel::DESCRIPTION = ¨description¨
```

JSON Label for description.

13.7.2.11 DIR

```
const std::string jsonlabel::DIR = ¨dir¨
```

JSON Label for directory.

13.7.2.12 DRIVES

```
const std::string jsonlabel::DRIVES = ¨drives¨
```

JSON Label for drives.

13.7.2.13 EVE_TEMPL

```
const std::string jsonlabel::EVE_TEMPL = ¨event templates¨
```

JSON Label for event templates.

13.7.2.14 EVENTS

```
const std::string jsonlabel::EVENTS = ¨events¨
```

JSON Label for events.

13.7.2.15 EXPORT

```
const std::string jsonlabel::EXPORT = ¨export¨
```

JSON Label for export.

13.7.2.16 FATHER

```
const std::string jsonlabel::FATHER = ¨father¨
```

JSON Label for father.

13.7.2.17 FILES

```
const std::string jsonlabel::FILES = ¨files¨
```

JSON Label for files.

13.7.2.18 FIRST_NAME

```
const std::string jsonlabel::FIRST_NAME = ¨first name¨
```

JSON Label for first name.

13.7.2.19 FIRST_PERSON

```
const std::string jsonlabel::FIRST_PERSON = ¨first person¨
```

JSON Label for first person.

13.7.2.20 FONT_FAMILY

```
const std::string jsonlabel::FONT_FAMILY = ¨font family¨
```

JSON Label for font family.

13.7.2.21 FONT_SIZE

```
const std::string jsonlabel::FONT_SIZE = ¨font size¨
```

JSON Label for font size.

13.7.2.22 FRONT_TITLE

```
const std::string jsonlabel::FRONT_TITLE = ¨title in front¨
```

JSON Label for title in front of a name.

13.7.2.23 GEN_DIFF

```
const std::string jsonlabel::GEN_DIFF = ¨generation difference¨
```

JSON Label for generation difference.

13.7.2.24 GEN_SIZE_DOWN

```
const std::string jsonlabel::GEN_SIZE_DOWN = ¨number of generations down¨
```

JSON Label for how many generations are being shown downwards.

13.7.2.25 GEN_SIZE_UP

```
const std::string jsonlabel::GEN_SIZE_UP = ¨number of generations up¨
```

JSON Label for how many generations are being shown upwards.

13.7.2.26 GENDER

```
const std::string jsonlabel::GENDER = ¨gender¨
```

JSON Label for gender.

13.7.2.27 GLOBAL_MAIN_PERSON

```
const std::string jsonlabel::GLOBAL_MAIN_PERSON = ¨global main person¨
```

JSON Label for global main person.

13.7.2.28 GREEN

```
const std::string jsonlabel::GREEN = ¨green¨
```

JSON Label for green.

13.7.2.29 HELP

```
const std::string jsonlabel::HELP = ¨help¨
```

JSON Label for help.

13.7.2.30 HIGHLIGHTED_COLOR

```
const std::string jsonlabel::HIGHLIGHTED_COLOR = ¨highlighted color¨
```

JSON Label highlighted color.

13.7.2.31 ID

```
const std::string jsonlabel::ID = ¨id¨
```

JSON Label for id.

13.7.2.32 LAST_DATE

```
const std::string jsonlabel::LAST_DATE = ¨last date¨
```

JSON Label for last date.

13.7.2.33 LINE_COLOR

```
const std::string jsonlabel::LINE_COLOR = ¨line color¨
```

JSON Label for the line color.

13.7.2.34 LINE_WIDTH

```
const std::string jsonlabel::LINE_WIDTH = ¨line width¨
```

JSON Label for line width.

13.7.2.35 LIVES

```
const std::string jsonlabel::LIVES = ¨lives¨
```

JSON Label for if the person lives or not.

13.7.2.36 MAIDEN_NAME

```
const std::string jsonlabel::MAIDEN_NAME = ¨maiden name¨
```

JSON Label for maiden name.

13.7.2.37 MEDIA

```
const std::string jsonlabel::MEDIA = ¨media¨
```

JSON Label for media.

13.7.2.38 MORE_PEOPLE

```
const std::string jsonlabel::MORE_PEOPLE = ¨more people¨
```

JSON Label for if the event contains more people.

13.7.2.39 MOTHER

```
const std::string jsonlabel::MOTHER = ¨mother¨
```

JSON Label for mother.

13.7.2.40 MOVEABLE_TABS

```
const std::string jsonlabel::MOVEABLE_TABS = ¨moveable tabs¨
```

JSON Label for setting moveable tabs.

13.7.2.41 NAME

```
const std::string jsonlabel::NAME = ¨name¨
```

JSON Label for name.

13.7.2.42 NOTES

```
const std::string jsonlabel::NOTES = ¨notes¨
```

JSON Label for notes.

13.7.2.43 PARTNER

```
const std::string jsonlabel::PARTNER = ¨partner¨
```

JSON Label for partner.

13.7.2.44 PERSON

```
const std::string jsonlabel::PERSON = ¨person¨
```

JSON Label for person.

13.7.2.45 PERSONS

```
const std::string jsonlabel::PERSONS = ¨persons¨
```

JSON Label for persons.

13.7.2.46 PLACE

```
const std::string jsonlabel::PLACE = ¨place¨
```

JSON Label for place.

13.7.2.47 PROBANT_COLOR

```
const std::string jsonlabel::PROBANT_COLOR = ¨proband color¨
```

JSON Label for proband color.

13.7.2.48 PROMOTED_COLOR

```
const std::string jsonlabel::PROMOTED_COLOR = ¨promoted color¨
```

JSON Label promoted color.

13.7.2.49 RED

```
const std::string jsonlabel::RED = ¨red¨
```

JSON Label for red.

13.7.2.50 REL_TEMPL

```
const std::string jsonlabel::REL_TEMPL = ¨relation templates¨
```

JSON Label for relations templates.

13.7.2.51 RELATIONS

```
const std::string jsonlabel::RELATIONS = ¨relations¨
```

JSON Label for relations.

13.7.2.52 ROLE

```
const std::string jsonlabel::ROLE = ¨role¨
```

JSON Label for role.

13.7.2.53 ROOT

```
const std::string jsonlabel::ROOT = ¨root¨
```

JSON Label for root.

13.7.2.54 SECOND_NAME

```
const std::string jsonlabel::SECOND_NAME = ¨second name¨
```

JSON Label for second name.

13.7.2.55 SECOND_PERSON

```
const std::string jsonlabel::SECOND_PERSON = ¨second person¨
```

JSON Label for second person.

13.7.2.56 SHOW_EVENT_TAB

```
const std::string jsonlabel::SHOW_EVENT_TAB = ¨show event tab¨
```

JSON Label for app settings if event tab should be visible.

13.7.2.57 SHOW_FILE_TAB

```
const std::string jsonlabel::SHOW_FILE_TAB = ¨show file tab¨
```

JSON Label for app settings if file tab should be visible.

13.7.2.58 SHOW_GENERAL_TAB

```
const std::string jsonlabel::SHOW_GENERAL_TAB = ¨show general tab¨
```

JSON Label for app settings if general tab should be visible.

13.7.2.59 SHOW_MEDIA_TAB

```
const std::string jsonlabel::SHOW_MEDIA_TAB = ¨show media tab¨
```

JSON Label for app settings if media tab should be visible.

13.7.2.60 SHOW_NOTE_TAB

```
const std::string jsonlabel::SHOW_NOTE_TAB = ¨show note tab¨
```

JSON Label for app settings if note tab should be visible.

13.7.2.61 SHOW_RELATION_TAB

```
const std::string jsonlabel::SHOW_RELATION_TAB = ¨show relation tab¨
```

JSON Label for app settings if relation tab should be visible.

13.7.2.62 SPECIAL_ROLES

```
const std::string jsonlabel::SPECIAL_ROLES = ¨special roles¨
```

JSON Label for if the event has special roles.

13.7.2.63 SPLITTER_POSITION_ONE

```
const std::string jsonlabel::SPLITTER_POSITION_ONE = ¨splitter position one¨
```

JSON Label for splitter position one.

13.7.2.64 SPLITTER_POSITION_TWO

```
const std::string jsonlabel::SPLITTER_POSITION_TWO = ¨splitter position two¨
```

JSON Label for splitter position two.

13.7.2.65 STANDARD_COLOR

```
const std::string jsonlabel::STANDARD_COLOR = ¨standard color¨
```

JSON Label standard color.

13.7.2.66 SURNAME

```
const std::string jsonlabel::SURNAME = ¨surname¨
```

JSON Label for surname.

13.7.2.67 TAB_POSITION

```
const std::string jsonlabel::TAB_POSITION = ¨tab position¨
```

JSON Label for tab position.

13.7.2.68 TAG

```
const std::string jsonlabel::TAG = ¨tag¨
```

JSON Label for single tag.

13.7.2.69 TAGS

```
const std::string jsonlabel::TAGS = ¨tags¨
```

JSON Label for tags.

13.7.2.70 TEMPLATE

```
const std::string jsonlabel::TEMPLATE = ¨template¨
```

JSON Label for for template.

13.7.2.71 TEMPLATES

```
const std::string jsonlabel::TEMPLATES = ¨templates¨
```

JSON Label for for templates.

13.7.2.72 TEXT

```
const std::string jsonlabel::TEXT = ¨text¨
```

JSON Label for text.

13.7.2.73 TEXT_COLOR

```
const std::string jsonlabel::TEXT_COLOR = ¨text color¨
```

JSON Label for the color of text.

13.7.2.74 TITLE

```
const std::string jsonlabel::TITLE = ¨title¨
```

JSON Label for title.

13.7.2.75 TRAIT

```
const std::string jsonlabel::TRAIT = ¨trait¨
```

JSON Label for trait.

13.7.2.76 TYPE

```
const std::string jsonlabel::TYPE = ¨type¨
```

JSON Label for type.

13.7.2.77 VALUE

```
const std::string jsonlabel::VALUE = ¨value¨
```

JSON Label for value.

13.7.2.78 VIRTUAL_DRIVE

```
const std::string jsonlabel::VIRTUAL_DRIVE = ¨virtual drive¨
```

JSON Label for virtual drive.

13.7.2.79 X_SIZE

```
const std::string jsonlabel::X_SIZE = ¨x size¨
```

JSON Label for size in the x axis.

13.7.2.80 Y_SIZE

```
const std::string jsonlabel::Y_SIZE = ¨y size¨
```

JSON Label for size in the y axis.

13.8 parser Namespace Reference

Namespace for all strings representing files and directories in the database.

Variables

- const std::string FILES_DIR = ¨Files¨

  Directory for all documents and files in general.
- const std::string JSON_CONFIG = ¨Config.json¨

  JSON file-name for configuration and templates.
- const std::string JSON_EVENTS = ¨Events.json¨

  JSON file-name for events.
- const std::string JSON_FILES = ¨Files.json¨

  JSON file-name for files meta-data.
- const std::string JSON_MEDIA = ¨Media.json¨

  JSON file-name for media meta-data.
- const std::string JSON_NOTES = ¨Notes.json¨

  JSON file-name for notes meta-data.
- const std::string JSON_PERSONS = ¨Persons.json¨

  JSON file-name for persons.
- const std::string JSON_RELATIONS = ¨Relations.json¨

  JSON file-name for relations.
- const std::string MEDIA_DIR = ¨Media¨

  Directory for all pictures and other media.
- const std::string NOTES_DIR = ¨Notes¨

  Directory for all notes.
- const std::string ERROR_LOG = ¨.error.log¨

  File for writing error log.

13.8.1 Detailed Description

Namespace for all strings representing files and directories in the database.

13.8.2 Variable Documentation

13.8.2.1 ERROR_LOG

```
const std::string parser::ERROR_LOG = ¨.error.log¨
```

File for writing error log.

13.8.2.2 FILES_DIR

```
const std::string parser::FILES_DIR = ¨Files¨
```

Directory for all documents and files in general.

13.8.2.3 JSON_CONFIG

```
const std::string parser::JSON_CONFIG = ¨Config.json¨
```

JSON file-name for configuration and templates.

13.8.2.4 JSON_EVENTS

```
const std::string parser::JSON_EVENTS = ¨Events.json¨
```

JSON file-name for events.

13.8.2.5 JSON_FILES

```
const std::string parser::JSON_FILES = ¨Files.json¨
```

JSON file-name for files meta-data.

13.8.2.6 JSON_MEDIA

```
const std::string parser::JSON_MEDIA = ¨Media.json¨
```

JSON file-name for media meta-data.

13.8.2.7 JSON_NOTES

```
const std::string parser::JSON_NOTES = ¨Notes.json¨
```

JSON file-name for notes meta-data.

13.8.2.8 JSON_PERSONS

```
const std::string parser::JSON_PERSONS = ¨Persons.json¨
```

JSON file-name for persons.

13.8.2.9 JSON_RELATIONS

```
const std::string parser::JSON_RELATIONS = ¨Relations.json¨
```

JSON file-name for relations.

13.8.2.10 MEDIA_DIR

```
const std::string parser::MEDIA_DIR = ¨Media¨
```

Directory for all pictures and other media.

13.8.2.11 NOTES_DIR

```
const std::string parser::NOTES_DIR = ¨Notes¨
```

Directory for all notes.

13.9 person Namespace Reference

Namespace for strings related to a person.

Variables

- const std::string UNKNOWN_STR = ¨Unknown¨

  String for unknown tags.
- const std::string SURNAME_STR = ¨Surname¨

  String for default surname.
- const std::string NAME_STR = ¨Name¨

  String for default name.
- const std::string ROOT = ¨root¨

  String for default root directories.
- const std::string DATE_FROM = ¨From: ¨

  String for date interval from.
- const std::string DATE_TO = ¨ to: ¨

  String for date interval to.

13.9.1 Detailed Description

Namespace for strings related to a person.

13.9.2 Variable Documentation

13.9.2.1 DATE_FROM

```
const std::string person::DATE_FROM = ¨From:  ¨
```

String for date interval from.

13.9.2.2 DATE_TO

```
const std::string person::DATE_TO = ¨ to:  ¨
```

String for date interval to.

13.9.2.3 NAME_STR

```
const std::string person::NAME_STR = ¨Name¨
```

String for default name.

13.9.2.4 ROOT

```
const std::string person::ROOT = ¨root¨
```

String for default root directories.

13.9.2.5 SURNAME_STR

```
const std::string person::SURNAME_STR = ¨Surname¨
```

String for default surname.

13.9.2.6 UNKNOWN_STR

```
const std::string person::UNKNOWN_STR = ¨Unknown¨
```

String for unknown tags.

13.10 relation Namespace Reference

Namespace for strings for default relations.

Variables

- const std::string FATHERHOOD = ¨Fatherhood¨

  String for Fatherhood.
- const std::string FATHERHOOD_DESC = ¨Relation between father and his child.¨

  String for Fatherhood description.
- const std::string FATHER = ¨Father¨

  String for father.
- const std::string CHILD = ¨Child¨

  String for child.
- const std::string MOTHERHOOD = ¨Motherhood¨

  String for Motherhood.
- const std::string MOTHERHOOD_DESC = ¨Relation between mother and her child.¨

  String for Motherhood description.
- const std::string MOTHER = ¨Mother¨

  String for mother.
- const std::string PARTNERSHIP = ¨Partnership¨

  String for Partnership.
- const std::string PARTNERSHIP_DESC = ¨Relation between two partners.¨

  String for description of partnership.
- const std::string PARTNER = ¨Partner¨

  String for partner.
- const std::string MARRIAGE = ¨Marriage¨

  String for Marriage.
- const std::string MARRIAGE_DESC = ¨Bound by law between two people. Usually a man (husband) and a woman (wife).¨

  String for Marriage description.
- const std::string WIFE = ¨Wife¨

  String for wife.
- const std::string HUSBAND = ¨Husband¨

  String for husband.
- const std::string ADOPTIVE_FATHERHOOD = ¨Adoptive fatherhood¨

  String for title for adoptive fatherhood.
- const std::string DESC_ADOPTIVE_FATHERHOOD = ¨Fatherhood provided by adopting a child.¨

  String for description for adoptive fatherhood.
- const std::string ADOPTIVE_FATHER = ¨Adoptive father¨

  String for adoptive father.
- const std::string ADOPTED_CHILD = ¨Adopted child¨

  String for adopted child.
- const std::string ADOPTIVE_MOTHERHOOD = ¨Adoptive motherhood¨

  String for title for adoptive motherhood.
- const std::string DESC_ADOPTIVE_MOTHERHOOD = ¨Motherhood provided by adopting a child.¨

  String for description for adoptive motherhood.
- const std::string ADOPTIVE_MOTHER = ¨Adoptive mother¨

  String for adoptive father.
- const std::string BROTHERHHOD = ¨Brotherhood¨

  String for brotherhood.
- const std::string DESC_BROTHERHOOD = ¨Relation between two brothers.¨

  String for brotherhood description.
- const std::string BROTHER = ¨Brother¨

  String for brother.

- const std::string SISTERHOOD = ¨Sisterhood¨

  String for sisterhood.
- const std::string DESC_SISTERHOOD = ¨Relation between two sisters.¨

  String for sisterhood description.
- const std::string SISTER = ¨Sister¨

  String for sister.
- const std::string SIBLINGS = ¨Siblings¨

  String for siblings.
- const std::string DESC_SIBLINGS = ¨Relation between brother and sister.¨

  String for siblings description.

13.10.1 Detailed Description

Namespace for strings for default relations.

13.10.2 Variable Documentation

13.10.2.1 ADOPTED_CHILD

```
const std::string relation::ADOPTED_CHILD = ¨Adopted child¨
```

String for adopted child.

13.10.2.2 ADOPTIVE_FATHER

```
const std::string relation::ADOPTIVE_FATHER = ¨Adoptive father¨
```

String for adoptive father.

13.10.2.3 ADOPTIVE_FATHERHOOD

```
const std::string relation::ADOPTIVE_FATHERHOOD = ¨Adoptive fatherhood¨
```

String for title for adoptive fatherhood.

13.10.2.4 ADOPTIVE_MOTHER

```
const std::string relation::ADOPTIVE_MOTHER = ¨Adoptive mother¨
```

String for adoptive father.

13.10.2.5 ADOPTIVE_MOTHERHOOD

```
const std::string relation::ADOPTIVE_MOTHERHOOD = ¨Adoptive motherhood¨
```

String for title for adoptive motherhood.

13.10.2.6 BROTHER

```
const std::string relation::BROTHER = ¨Brother¨
```

String for brother.

13.10.2.7 BROTHERHHOD

```
const std::string relation::BROTHERHHOD = ¨Brotherhood¨
```

String for brotherhood.

13.10.2.8 CHILD

```
const std::string relation::CHILD = ¨Child¨
```

String for child.

13.10.2.9 DESC_ADOPTIVE_FATHERHOOD

```
const std::string relation::DESC_ADOPTIVE_FATHERHOOD = ¨Fatherhood provided by adopting a
child.¨
```

String for description for adoptive fatherhood.

13.10.2.10 DESC_ADOPTIVE_MOTHERHOOD

```
const std::string relation::DESC_ADOPTIVE_MOTHERHOOD = ¨Motherhood provided by adopting a
child.¨
```

String for description for adoptive motherhood.

13.10.2.11 DESC_BROTHERHOOD

```
const std::string relation::DESC_BROTHERHOOD = ¨Relation between two brothers.¨
```

String for brotherhood description.

13.10.2.12 DESC_SIBLINGS

```
const std::string relation::DESC_SIBLINGS = ¨Relation between brother and sister.¨
```

String for siblings description.

13.10.2.13 DESC_SISTERHOOD

```
const std::string relation::DESC_SISTERHOOD = ¨Relation between two sisters.¨
```

String for sisterhood description.

13.10.2.14 FATHER

```
const std::string relation::FATHER = ¨Father¨
```

String for father.

13.10.2.15 FATHERHOOD

```
const std::string relation::FATHERHOOD = ¨Fatherhood¨
```

String for Fatherhood.

13.10.2.16 FATHERHOOD_DESC

```
const std::string relation::FATHERHOOD_DESC = ¨Relation between father and his child.¨
```

String for Fatherhood description.

13.10.2.17 HUSBAND

```
const std::string relation::HUSBAND = ¨Husband¨
```

String for husband.

13.10.2.18 MARRIAGE

```
const std::string relation::MARRIAGE = ¨Marriage¨
```

String for Marriage.

13.10.2.19 MARRIAGE_DESC

```
const std::string relation::MARRIAGE_DESC = ¨Bound by law between two people.  Usually a man
(husband) and a woman (wife).¨
```

String for Marriage description.

13.10.2.20 MOTHER

```
const std::string relation::MOTHER = ¨Mother¨
```

String for mother.

13.10.2.21 MOTHERHOOD

```
const std::string relation::MOTHERHOOD = ¨Motherhood¨
```

String for Motherhood.

13.10.2.22 MOTHERHOOD_DESC

```
const std::string relation::MOTHERHOOD_DESC = ¨Relation between mother and her child.¨
```

String for Motherhood description.

13.10.2.23 PARTNER

```
const std::string relation::PARTNER = ¨Partner¨
```

String for partner.

13.10.2.24 PARTNERSHIP

```
const std::string relation::PARTNERSHIP = ¨Partnership¨
```

String for Partnership.

13.10.2.25 PARTNERSHIP_DESC

```
const std::string relation::PARTNERSHIP_DESC = ¨Relation between two partners.¨
```

String for description of partnership.

13.10.2.26 SIBLINGS

```
const std::string relation::SIBLINGS = ¨Siblings¨
```

String for siblings.

13.10.2.27 SISTER

```
const std::string relation::SISTER = ¨Sister¨
```

String for sister.

13.10.2.28 SISTERHOOD

```
const std::string relation::SISTERHOOD = ¨Sisterhood¨
```

String for sisterhood.

13.10.2.29 WIFE

```
const std::string relation::WIFE = ¨Wife¨
```

String for wife.

13.11 tagchooser Namespace Reference

Namespace for strings shown in tag chooser dialog.

Variables

- const std::string CUSTOM_TAG = ¨+ Custom tag¨

  Last custom tag choice.

13.11.1 Detailed Description

Namespace for strings shown in tag chooser dialog.

13.11.2 Variable Documentation

13.11.2.1 CUSTOM_TAG

```
const std::string tagchooser::CUSTOM_TAG = ¨+ Custom tag¨
```

Last custom tag choice.

13.12 trait Namespace Reference

Namespace for everything with relation traits.

Functions

- std::string str (Trait t)

Variables

- constexpr Trait AllTraits [ ] = {None, Fatherhood, Motherhood, Partnership, Sibling}

  All possible traits.

13.12.1 Detailed Description

Namespace for everything with relation traits.

13.12.2 Function Documentation

13.12.2.1 str()

```
std::string trait::str (
             Trait t )
```

Represent trait as a string.

Parameters

t	Given trait.

Returns

String representing given trait.

13.12.3 Variable Documentation

13.12.3.1 AllTraits

```
constexpr Trait trait::AllTraits[ ] = {None, Fatherhood, Motherhood, Partnership, Sibling}
[constexpr]
```

All possible traits.

13.13 Ui Namespace Reference

Namespace for classes having all its user interface items.

13.13.1 Detailed Description

Namespace for classes having all its user interface items.

Chapter 14

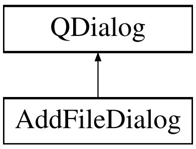
Class Documentation

14.1 AddFileDialog Class Reference

Dialog for adding file.

```
#include <dialogs.h>
```

Inheritance diagram for AddFileDialog:



Public Slots

- void copyFile ()

  Open file chooser for copying file.
- void createOrChooseFile ()

  Create or choose file based on selected tab.
- void openFile (QTreeWidgetItem ∗item)

Public Member Functions

- AddFileDialog (FileType type, const std::vector< std::string > &extensions, VirtualDrive ∗parentDrive, FamilyTree ∗FT, QWidget ∗parent)
- ∼AddFileDialog ()

  Default destructor.

Private Member Functions

- size_t getId (QTreeWidgetItem ∗item)

Private Attributes

- FamilyTree ∗ FT_

  Used family tree.
- VirtualDrive ∗ drive_

  Parent drive.
- FileType type_

  Type of the file.
- Ui::AddFileDialog ∗ ui

  User interface of the Qt framework.

14.1.1 Detailed Description

Dialog for adding file.

14.1.2 Constructor & Destructor Documentation

14.1.2.1 AddFileDialog()

```
AddFileDialog::AddFileDialog (
            FileType type,
            const std::vector< std::string > & extensions,
            VirtualDrive ∗ parentDrive,
            FamilyTree ∗ FT,
            QWidget ∗ parent )
```

Default constructor.

Parameters

type	If the file is general file, media or note.
extensions	Vector of all extensions to show.
parentDrive	Parent drive where the file will be stored.
FT	Given family tree the current app is using.
parent	The Qt Widget parent.

14.1.2.2 ∼AddFileDialog()

```
AddFileDialog::∼AddFileDialog ( )
```

Default destructor.

14.1.3 Member Function Documentation

14.1.3.1 copyFile

```
void AddFileDialog::copyFile ( ) [slot]
```

Open file chooser for copying file.

14.1.3.2 createOrChooseFile

```
void AddFileDialog::createOrChooseFile ( )  [slot]
```

Create or choose file based on selected tab.

14.1.3.3 getId()

```
size_t AddFileDialog::getId (
             QTreeWidgetItem * item )  [private]
```

Get the id from the text of the given id.

Parameters

item	Which item has the id.

Returns

Id from its text.

14.1.3.4 openFile

```
void AddFileDialog::openFile (
             QTreeWidgetItem * item )  [slot]
```

Show file upon clicking it.

Parameters

item	Which item was clicked.

14.1.4 Member Data Documentation

14.1.4.1 drive_

```
VirtualDrive* AddFileDialog::drive_  [private]
```

Parent drive.

14.1.4.2 FT_

```
FamilyTree* AddFileDialog::FT_  [private]
```

Used family tree.

14.1.4.3 type_

```
FileType AddFileDialog::type_ [private]
```

Type of the file.

14.1.4.4 ui

```
Ui::AddFileDialog* AddFileDialog::ui [private]
```

User interface of the Qt framework.

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

- graphics/dialogs.h
- graphics/dialogs.cpp

14.2 AppSettings Struct Reference

Struct for app settings.

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

Public Member Functions

- void readJson (const Json::Value &value)
- void setDefault ()

  Set the default options.

Public Attributes

- std::string fontFamily

  Used font in the graphic boxes.
- int fontSize

  Size of the font used in the graphic boxes.
- int genSizeDown

  Number of generations to show down.
- int genSizeUp

  Number of generations to show up.
- Color highlightedColor

  Color for highlighted box.
- Color lineColor

  Color of the lines.
- int lineWidth

  Width of the line.
- Color probandColor

  Color for proband.
- Color promoted

Color for promoted relation.

- int radius

  Border radius.
- bool showEventTab

  If event tab should be visible.
- bool showFileTab

  If file tab should be visible.
- bool showGeneralTab

  If general tab should be visible.
- bool showMediaTab

  If media tab should be visible.
- bool showNoteTab

  If note tab should be visible.
- bool showRelationTab

  If relation tab should be visible.
- int sizeX

  Size of the box in x axis.
- int sizeY

  Size of the box in y axis.
- int splitterPositionOne

  Splitter position one.
- int splitterPositionTwo

  Splitter position two.
- Color standard

  Standard color.
- TabPosition tabPosition

  Tab position of the information tab.
- Color textColor

  Color of the text.

14.2.1 Detailed Description

Struct for app settings.

14.2.2 Member Function Documentation

14.2.2.1 readJson()

```
void AppSettings::readJson (
             const Json::Value & value )
```

Read and load data from JSON value.

Parameters

value	Given JSON value.

14.2.2.2 setDefault()

```
void AppSettings::setDefault ( )
```

Set the default options.

14.2.3 Member Data Documentation

14.2.3.1 fontFamily

```
std::string AppSettings::fontFamily
```

Used font in the graphic boxes.

14.2.3.2 fontSize

```
int AppSettings::fontSize
```

Size of the font used in the graphic boxes.

14.2.3.3 genSizeDown

```
int AppSettings::genSizeDown
```

Number of generations to show down.

14.2.3.4 genSizeUp

```
int AppSettings::genSizeUp
```

Number of generations to show up.

14.2.3.5 highlightedColor

```
Color AppSettings::highlightedColor
```

Color for highlighted box.

14.2.3.6 lineColor

```
Color AppSettings::lineColor
```

Color of the lines.

14.2.3.7 lineWidth

```
int AppSettings::lineWidth
```

Width of the line.

14.2.3.8 probandColor

```
Color AppSettings::probandColor
```

Color for proband.

14.2.3.9 promoted

```
Color AppSettings::promoted
```

Color for promoted relation.

14.2.3.10 radius

```
int AppSettings::radius
```

Border radius.

14.2.3.11 showEventTab

```
bool AppSettings::showEventTab
```

If event tab should be visible.

14.2.3.12 showFileTab

```
bool AppSettings::showFileTab
```

If file tab should be visible.

14.2.3.13 showGeneralTab

```
bool AppSettings::showGeneralTab
```

If general tab should be visible.

14.2.3.14 showMediaTab

```
bool AppSettings::showMediaTab
```

If media tab should be visible.

14.2.3.15 showNoteTab

```
bool AppSettings::showNoteTab
```

If note tab should be visible.

14.2.3.16 showRelationTab

```
bool AppSettings::showRelationTab
```

If relation tab should be visible.

14.2.3.17 sizeX

```
int AppSettings::sizeX
```

Size of the box in x axis.

14.2.3.18 sizeY

```
int AppSettings::sizeY
```

Size of the box in y axis.

14.2.3.19 splitterPositionOne

```
int AppSettings::splitterPositionOne
```

Splitter position one.

14.2.3.20 splitterPositionTwo

```
int AppSettings::splitterPositionTwo
```

Splitter position two.

14.2.3.21 standard

```
Color AppSettings::standard
```

Standard color.

14.2.3.22 tabPosition

```
TabPosition AppSettings::tabPosition
```

Tab position of the information tab.

14.2.3.23 textColor

```
Color AppSettings::textColor
```

Color of the text.

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

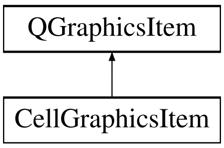
- core/config.h
- core/config.cpp

14.3 CellGraphicsItem Class Reference

Cell around all relations.

```
#include <graphics_items.h>
```

Inheritance diagram for CellGraphicsItem:



Public Member Functions

- CellGraphicsItem (int sizex, int sizey, qreal x, qreal y, QPen ∗pen, int borderRadius)
- QRectF boundingRect () const override
- void paint (QPainter ∗painter, const QStyleOptionGraphicsItem ∗option, QWidget ∗widget) override
- QPainterPath shape () const override

Private Attributes

- int borderRadius_

  Border radius.
- QPen ∗ pen_

  Pointer to the used pen.
- int sizex_

  Size of the cell in the x axis.
- int sizey_

  Size of the cell in the y axis.
- qreal x_

  Top left point x position.
- qreal y_

  Top left point y position.

14.3.1 Detailed Description

Cell around all relations.

14.3.2 Constructor & Destructor Documentation

14.3.2.1 CellGraphicsItem()

```
CellGraphicsItem::CellGraphicsItem (
             int sizex,
             int sizey,
             qreal x,
             qreal y,
             QPen * pen,
             int borderRadius )
```

Default constructor.

Parameters

sizex	Size of the cell in the x axis.
sizey	Size of the cell in the y axis.
x	Top left point x position.
y	Top left point y position.
pen	Which pen is used for lines.
borderRadius	What is the radius of the border.

14.3.3 Member Function Documentation

14.3.3.1 boundingRect()

```
QRectF CellGraphicsItem::boundingRect ( ) const  [override]
```

Get the bounding rectangle of the object.

Returns

Rectangle that is bounding the item.

14.3.3.2 paint()

```
void CellGraphicsItem::paint (
             QPainter * painter,
             const QStyleOptionGraphicsItem * option,
             QWidget * widget )  [override]
```

Paint the object to its painter.

Parameters

painter	Which painter to use to paint the object.
option	Option for the Qt framework (not used).
widget	Widget for the Qt framework (not used).

14.3.3.3 shape()

```
QPainterPath CellGraphicsItem::shape ( ) const  [override]
```

Get the shape of the item.

Returns

Get the shape to paint.

14.3.4 Member Data Documentation

14.3.4.1 borderRadius_

```
int CellGraphicsItem::borderRadius_  [private]
```

Border radius.

14.3.4.2 pen_

```
QPen* CellGraphicsItem::pen_  [private]
```

Pointer to the used pen.

14.3.4.3 sizex_

```
int CellGraphicsItem::sizex_  [private]
```

Size of the cell in the x axis.

14.3.4.4 sizey_

```
int CellGraphicsItem::sizey_  [private]
```

Size of the cell in the y axis.

14.3.4.5 x_

```
qreal CellGraphicsItem::x_  [private]
```

Top left point x position.

14.3.4.6 y_

```
qreal CellGraphicsItem::y_ [private]
```

Top left point y position.

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

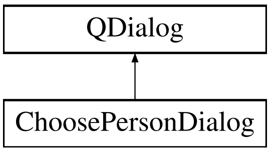
- graphics/graphics_items.h
- graphics/graphics_items.cpp

14.4 ChoosePersonDialog Class Reference

Dialog for choosing person with find window. Parent have to have slot `savePerson()`.

```
#include <dialogs.h>
```

Inheritance diagram for ChoosePersonDialog:



Public Slots

- void clearFilter ()

  Clear filter and show all persons.
- void filterPersons ()

  Filter persons by the text edit.
- void saveSelectedPerson ()

  Save currently selected person.

Signals

- void savePerson (Person ∗person)

  When save can be proceeded.

Public Member Functions

- ChoosePersonDialog (FamilyTree ∗FT, Person ∗person, QWidget ∗parent=nullptr)
- ∼ChoosePersonDialog ()

  Default destructor.

Private Attributes

- FamilyTree ∗ FT_

  Pointer to the family tree.
- Person ∗ person_

  Pointer for.
- Ui::ChoosePersonDialog ∗ ui

  User interface of the Qt framework.

14.4.1 Detailed Description

Dialog for choosing person with find window. Parent have to have slot `savePerson()`.

14.4.2 Constructor & Destructor Documentation

14.4.2.1 ChoosePersonDialog()

```
ChoosePersonDialog::ChoosePersonDialog (
            FamilyTree ∗ FT,
            Person ∗ person,
            QWidget ∗ parent = nullptr )
```

Default constructor.

Parameters

FT	Pointer to the family tree.
person	Pointer to the person to be replaced or nullptr to choose new one.
parent	The Qt Widget parent.

14.4.2.2 ∼ChoosePersonDialog()

```
ChoosePersonDialog::∼ChoosePersonDialog ( )
```

Default destructor.

14.4.3 Member Function Documentation

14.4.3.1 clearFilter

```
void ChoosePersonDialog::clearFilter ( ) [slot]
```

Clear filter and show all persons.

14.4.3.2 filterPersons

```
void ChoosePersonDialog::filterPersons ( )  [slot]
```

Filter persons by the text edit.

14.4.3.3 savePerson

```
void ChoosePersonDialog::savePerson (
             Person * person )  [signal]
```

When save can be proceeded.

14.4.3.4 saveSelectedPerson

```
void ChoosePersonDialog::saveSelectedPerson ( )  [slot]
```

Save currently selected person.

14.4.4 Member Data Documentation

14.4.4.1 FT_

```
FamilyTree* ChoosePersonDialog::FT_  [private]
```

Pointer to the family tree.

14.4.4.2 person_

```
Person* ChoosePersonDialog::person_  [private]
```

Pointer for.

14.4.4.3 ui

```
Ui::ChoosePersonDialog* ChoosePersonDialog::ui  [private]
```

User interface of the Qt framework.

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

- graphics/dialogs.h
- graphics/dialogs.cpp

14.5 Color Struct Reference

Struct for general color.

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

Public Member Functions

- Color (int r=255, int g=255, int b=255)
- void readJson (const Json::Value &value)

  Read it from json value.

Public Attributes

- int red

  Red part of the color.
- int green

  Green part of the color.
- int blue

  Blue part of the color.

14.5.1 Detailed Description

Struct for general color.

14.5.2 Constructor & Destructor Documentation

14.5.2.1 Color()

```
Color::Color (
            int r = 255,
            int g = 255,
            int b = 255 )  [explicit]
```

Default constructor.

Parameters

r	Red.
g	Green.
b	Blue.

14.5.3 Member Function Documentation

14.5.3.1 readJson()

```
void Color::readJson (
             const Json::Value & value )
```

Read it from json value.

14.5.4 Member Data Documentation

14.5.4.1 blue

```
int Color::blue
```

Blue part of the color.

14.5.4.2 green

```
int Color::green
```

Green part of the color.

14.5.4.3 red

```
int Color::red
```

Red part of the color.

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

- core/config.h
- core/config.cpp

14.6 Date Class Reference

Basic date format. Holding three numbers for year, month and day. If some of the numbers are not defined 0 is present.

```
#include <date.h>
```

Public Member Functions

- Date (int y=0, int m=0, int d=0)
- int getDay () const
- int getMonth () const
- int getYear () const
- bool isEmpty () const
- bool isExact () const
- void read (const std::string &date)
- void setDay (int d)
- void setMonth (int m)
- void setYear (int y)
- std::string str () const

Private Attributes

- int day

  Day number.
- int month

  Month number.
- int year

  Year number.

14.6.1 Detailed Description

Basic date format. Holding three numbers for year, month and day. If some of the numbers are not defined 0 is present.

14.6.2 Constructor & Destructor Documentation

14.6.2.1 Date()

```
Date::Date (
             int y = 0,
             int m = 0,
             int d = 0 )  [explicit]
```

Default constructor.

Parameters

d	Day number.
m	Month number.
y	Year number.

14.6.3 Member Function Documentation

14.6.3.1 getDay()

```
int Date::getDay ( ) const
```

Get the day number.

Returns

Day number (0 if it was not set).

14.6.3.2 getMonth()

```
int Date::getMonth ( ) const
```

Get the month number.

Returns

Month number (0 if it was not set).

14.6.3.3 getYear()

```
int Date::getYear ( ) const
```

Get the year number.

Returns

Year number (0 if it was not set).

14.6.3.4 isEmpty()

```
bool Date::isEmpty ( ) const
```

Check if the date is completely not defined. All values are non-defined if they are equal to 0.

Returns

True if all values are 0.

14.6.3.5 isExact()

```
bool Date::isExact ( ) const
```

Check if the date is exact or not. Date is exact when all values are defined (non-zero).

Returns

True if the date is exact.

14.6.3.6 read()

```
void Date::read (
             const std::string & date )
```

Load date from string in specified flexible format: yyyy-mm-dd (if there would be for example 05 it can only be 5). If it does not follow the format there will be corrupted data and won't raise an exception.

Parameters

date	Written date in correct format.

14.6.3.7 setDay()

```
void Date::setDay (
             int d )
```

Set the day number.

Parameters

d	Day number.

14.6.3.8 setMonth()

```
void Date::setMonth (
             int m )
```

Set the month number.

Parameters

m	Month number.

14.6.3.9 setYear()

```
void Date::setYear (
             int y )
```

Set the year number.

Parameters

y	Year number.

14.6.3.10 str()

```
std::string Date::str ( ) const
```

Get string representing the date in format yyyy-MM-DD. If one of the values are set to 0, then it is replaced by the same number of ?

Returns

String in the format.

14.6.4 Member Data Documentation

14.6.4.1 day

```
int Date::day  [private]
```

Day number.

14.6.4.2 month

```
int Date::month  [private]
```

Month number.

14.6.4.3 year

```
int Date::year  [private]
```

Year number.

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

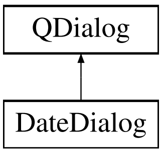
- core/date.h
- core/date.cpp

14.7 DateDialog Class Reference

Dialog for editing custom dates.

```
#include <dialogs.h>
```

Inheritance diagram for DateDialog:



Public Slots

- void saveDate ()

  Slot for saving the date if OK button was pressed.
- void unknownDate (bool isUnknown)

Public Member Functions

- DateDialog (FamilyTree ∗FT, WrappedDate ∗date=nullptr, QWidget ∗parent=nullptr)
- ∼DateDialog ()

  Default destructor.

Private Attributes

- WrappedDate ∗ date_

  Pointer to the used date.
- FamilyTree ∗ FT_

  Pointer to the family tree.
- Ui::DateDialog ∗ ui

  User interface of the Qt framework.

14.7.1 Detailed Description

Dialog for editing custom dates.

14.7.2 Constructor & Destructor Documentation

14.7.2.1 DateDialog()

```
DateDialog::DateDialog (
            FamilyTree ∗ FT,
            WrappedDate ∗ date = nullptr,
            QWidget ∗ parent = nullptr )  [explicit]
```

Default constructor.

Parameters

FT	Pointer to the family tree.
date	Pointer to the used date.
parent	The Qt Widget parent.

14.7.2.2 ∼DateDialog()

```
DateDialog::∼DateDialog ( )
```

Default destructor.

14.7.3 Member Function Documentation

14.7.3.1 saveDate

```
void DateDialog::saveDate ( )  [slot]
```

Slot for saving the date if OK button was pressed.

14.7.3.2 unknownDate

```
void DateDialog::unknownDate (
             bool isUnknown )  [slot]
```

If the unknown check box was changed.

Parameters

isUnknown	If the check box was checked.

14.7.4 Member Data Documentation

14.7.4.1 date_

```
WrappedDate* DateDialog::date_  [private]
```

Pointer to the used date.

14.7.4.2 FT_

```
FamilyTree* DateDialog::FT_  [private]
```

Pointer to the family tree.

14.7.4.3 ui

```
Ui::DateDialog* DateDialog::ui  [private]
```

User interface of the Qt framework.

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

- graphics/dialogs.h
- graphics/dialogs.cpp

14.8 Event Class Reference

Class holding data for a single event.

```
#include <family_tree_items.h>
```

Public Member Functions

- Event (Settings *settings)
- void addPerson (size_t person, const std::string &role=events::NO_ROLE)
- void clearPersons ()

  Clear all persons.
- WrappedDate * date ()
- const WrappedDate & getDate () const
- size_t getId () const
- const std::vector< std::pair< std::string, size_t > > & getPersons () const
- const std::string & getPlace () const
- const Settings & getSettings () const
- size_t getTemplate () const
- const std::string & getText () const
- void printHtml (std::ostream &os, const std::vector< std::pair< std::string, std::string > > &persons)
- void readJson (const Json::Value &value)
- bool removePerson (size_t id)
- WrappedDate & setDate ()
- void setId (size_t id)
- void setPlace (const std::string &place)
- void setTemplate (size_t templ)
- void setText (const std::string &text)
- std::string str () const
- std::vector< size_t > updateToTemplate ()

Private Attributes

- WrappedDate date_

  Date of this event.
- size_t id_

  Id of this event.
- std::vector< std::pair< std::string, size_t > > persons_

  People with their roles.
- std::string place_

  Place of this event.
- Settings * settings_

  Settings used in app.
- size_t template_

  Template for this event.
- std::string text_

  Text (note or value) for this event.

14.8.1 Detailed Description

Class holding data for a single event.

14.8.2 Constructor & Destructor Documentation

14.8.2.1 Event()

```
Event::Event (
            Settings * settings )  [explicit]
```

Default constructor.

Parameters

settings	The settings used in app (to access the template).

14.8.3 Member Function Documentation

14.8.3.1 addPerson()

```
void Event::addPerson (
             size_t person,
             const std::string & role = events::NO_ROLE )
```

Add person to the vector.

Parameters

person	Id of the person.
role	Special role for this person, or empty if he does not have special role.

14.8.3.2 clearPersons()

```
void Event::clearPersons ( )
```

Clear all persons.

14.8.3.3 date()

```
WrappedDate * Event::date ( )
```

Get the pointer to the date.

Returns

Pointer to the wrapped date.

14.8.3.4 getDate()

```
const WrappedDate & Event::getDate ( ) const
```

Get the date of this event.

Returns

Constant reference to the date.

14.8.3.5 getId()

```
size_t Event::getId ( ) const
```

Get the id of this event.

Returns

Id of the event.

14.8.3.6 getPersons()

```
const std::vector< std::pair< std::string, size_t > > & Event::getPersons ( ) const
```

Get the vector of persons.

Returns

Constant reference to the vector of persons.

14.8.3.7 getPlace()

```
const std::string & Event::getPlace ( ) const
```

Get the place of this event.

Returns

Constant reference to the place.

14.8.3.8 getSettings()

```
const Settings & Event::getSettings ( ) const
```

Get the settings.

Returns

Constant reference to the settings.

14.8.3.9 getTemplate()

```
size_t Event::getTemplate ( ) const
```

Get the template for this event.

Returns

Id of the template.

14.8.3.10 getText()

```
const std::string & Event::getText ( ) const
```

Get the text note of the event.

Returns

Constant reference to the text.

14.8.3.11 printHtml()

```
void Event::printHtml (
            std::ostream & os,
            const std::vector< std::pair< std::string, std::string > > & persons )
```

Print event to an output stream in HTML format.

Parameters

os	Given output stream,
persons	List of pairs for role&person names.

14.8.3.12 readJson()

```
void Event::readJson (
            const Json::Value & value )
```

Read the relation from JSON value.

Parameters

value	The JSON value from file.

14.8.3.13 removePerson()

```
bool Event::removePerson (
            size_t id )
```

Remove person from the list of all persons.

Parameters

id	Id of the person.

Returns

True if there is no other person left (thus left out event), otherwise false.

14.8.3.14 setDate()

```
WrappedDate & Event::setDate ( )
```

Set the date for this event.

Returns

Reference to the date.

14.8.3.15 setId()

```
void Event::setId (
             size_t id )
```

Set the id for this event.

Parameters

id	Given id.

14.8.3.16 setPlace()

```
void Event::setPlace (
             const std::string & place )
```

Set the place for this event.

Parameters

place	New place.

14.8.3.17 setTemplate()

```
void Event::setTemplate (
             size_t templ )
```

Set the template of this event.

Parameters

templ	Id of the new template.

14.8.3.18 setText()

```
void Event::setText (
            const std::string & text )
```

Set the text for this event.

Parameters

text	Given new text.

14.8.3.19 str()

```
std::string Event::str ( ) const
```

Get the event representation as a string.

Returns

Newly constructed string representing the event.

14.8.3.20 updateToTemplate()

```
std::vector< size_t > Event::updateToTemplate ( )
```

Update this event to the changed template.

Returns

Vector of all persons which were removed.

14.8.4 Member Data Documentation

14.8.4.1 date_

```
WrappedDate Event::date_ [private]
```

Date of this event.

14.8.4.2 id_

```
size_t Event::id_ [private]
```

Id of this event.

14.8.4.3 persons_

```
std::vector<std::pair<std::string, size_t> > Event::persons_ [private]
```

People with their roles.

14.8.4.4 place_

```
std::string Event::place_ [private]
```

Place of this event.

14.8.4.5 settings_

```
Settings* Event::settings_ [private]
```

Settings used in app.

14.8.4.6 template_

```
size_t Event::template_ [private]
```

Template for this event.

14.8.4.7 text_

```
std::string Event::text_ [private]
```

Text (note or value) for this event.

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

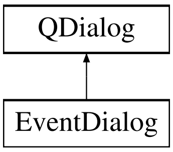
- core/family_tree_items.h
- core/family_tree_items.cpp

14.9 EventDialog Class Reference

Dialog showing single event for editing it.

```
#include <dialogs.h>
```

Inheritance diagram for EventDialog:



Public Slots

- void addEmptyRole ()

  Add empty role to the roles.
- void choosePerson ()

  Open dialog for choosing person.
- void openCustomDate ()

  Open dialog for custom date.
- void refreshDates ()

  Refresh date from dialog.
- void refreshPersons (QTreeWidgetItem ∗item)
- void removeEmptyRole ()

  Remove empty role from the list.
- void removePerson ()

  Remove person from its role.
- void saveEvent ()

  Save the data to the event.
- void savePerson (Person ∗person)
- void showTemplate ()

  Show the template details.

Public Member Functions

- EventDialog (FamilyTree ∗FT, QWidget ∗parent=nullptr, Event ∗event=nullptr)
- ∼EventDialog ()

  Default destructor.

Private Member Functions

- void showEventDetails ()

  Show event details.

Private Attributes

- WrappedDate date_

  Wrapped date of this event.
- Event ∗ event_

  Pointer to the event that is being edited, may be a null pointer.
- FamilyTree ∗ FT_

  Pointer to the family tree.
- Person ∗ main_

  Pointer to the main person used if no other persons are in event.
- Ui::EventDialog ∗ ui

  User interface of the Qt framework.

14.9.1 Detailed Description

Dialog showing single event for editing it.

14.9.2 Constructor & Destructor Documentation

14.9.2.1 EventDialog()

```
EventDialog::EventDialog (
            FamilyTree * FT,
            QWidget * parent = nullptr,
            Event * event = nullptr )  [explicit]
```

Default constructor.

Parameters

FT	Pointer to the family tree.
parent	The Qt Widget parent.
event	Which event is being edited if any.

14.9.2.2 ∼EventDialog()

```
EventDialog::∼EventDialog ( )
```

Default destructor.

14.9.3 Member Function Documentation

14.9.3.1 addEmptyRole

```
void EventDialog::addEmptyRole ( )  [slot]
```

Add empty role to the roles.

14.9.3.2 choosePerson

```
void EventDialog::choosePerson ( )  [slot]
```

Open dialog for choosing person.

14.9.3.3 openCustomDate

```
void EventDialog::openCustomDate ( )  [slot]
```

Open dialog for custom date.

14.9.3.4 refreshDates

```
void EventDialog::refreshDates ( )  [slot]
```

Refresh date from dialog.

14.9.3.5 refreshPersons

```
void EventDialog::refreshPersons (
            QTreeWidgetItem * item )  [slot]
```

Refresh persons comboBox to the current role and its value.

Parameters

item	Which item was clicked on.

14.9.3.6 removeEmptyRole

```
void EventDialog::removeEmptyRole ( )  [slot]
```

Remove empty role from the list.

14.9.3.7 removePerson

```
void EventDialog::removePerson ( )  [slot]
```

Remove person from its role.

14.9.3.8 saveEvent

```
void EventDialog::saveEvent ( )  [slot]
```

Save the data to the event.

14.9.3.9 savePerson

```
void EventDialog::savePerson (
            Person * person )  [slot]
```

Save person from dialog.

Parameters

person	Which person was selected.

14.9.3.10 showEventDetails()

```
void EventDialog::showEventDetails ( )  [private]
```

Show event details.

14.9.3.11 showTemplate

```
void EventDialog::showTemplate ( )  [slot]
```

Show the template details.

14.9.4 Member Data Documentation

14.9.4.1 date_

```
WrappedDate EventDialog::date_  [private]
```

Wrapped date of this event.

14.9.4.2 event_

```
Event* EventDialog::event_  [private]
```

Pointer to the event that is being edited, may be a null pointer.

14.9.4.3 FT_

```
FamilyTree* EventDialog::FT_  [private]
```

Pointer to the family tree.

14.9.4.4 main_

```
Person* EventDialog::main_  [private]
```

Pointer to the main person used if no other persons are in event.

14.9.4.5 ui

```
Ui::EventDialog* EventDialog::ui  [private]
```

User interface of the Qt framework.

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

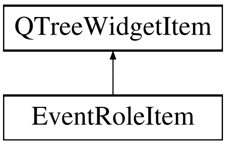
- graphics/dialogs.h
- graphics/dialogs.cpp

14.10 EventRoleItem Class Reference

Tree widget item for persons in events.

```
#include <graphics_items.h>
```

Inheritance diagram for EventRoleItem:



Public Member Functions

- EventRoleItem (const std::string &role, size_t id=0)
- size_t getId ()
- void setPerson (const QString &name, size_t id=0)

Private Attributes

- size_t id_

  Id of the person.

14.10.1 Detailed Description

Tree widget item for persons in events.

14.10.2 Constructor & Destructor Documentation

14.10.2.1 EventRoleItem()

```
EventRoleItem::EventRoleItem (
             const std::string & role,
             size_t id = 0 )   [explicit]
```

Default constructor.

Parameters

role	String representing the role.
id	Id of the person if any.

14.10.3 Member Function Documentation

14.10.3.1 getId()

```
size_t EventRoleItem::getId ( )
```

Get the id of the person.

Returns

Id of the person.

14.10.3.2 setPerson()

```
void EventRoleItem::setPerson (
             const QString & name,
             size_t id = 0 )
```

Set person to the role.

Parameters

name	Name of the person.
id	Id of the person.

14.10.4 Member Data Documentation

14.10.4.1 id_

```
size_t EventRoleItem::id_  [private]
```

Id of the person.

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

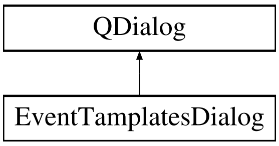
- graphics/graphics_items.h
- graphics/graphics_items.cpp

14.11 EventTamplatesDialog Class Reference

Dialog showing all event templates and possibly editing them or creating new ones.

```
#include <dialogs.h>
```

Inheritance diagram for EventTamplatesDialog:



Public Slots

- void addDefaults ()

  Add default templates.
- void addNewRole ()

  Add new role to the list.
- void addNewTemplate ()

  Show empty boxes to add new template.
- void removeRole ()

  Remove role from the list.
- void removeTemplate ()

  Remove template from the list.
- void saveTemplate ()

  Save template or create a new one.
- void showTemplate (QTreeWidgetItem ∗item)

Public Member Functions

- EventTamplatesDialog (FamilyTree ∗FT, QWidget ∗parent=nullptr)
- ∼EventTamplatesDialog ()

    Default destructor.

Private Member Functions

- void clear ()

    Clear all boxes.
- void refreshListWidget ()

    Refresh list of all relation templates.
- void saveTemplate (EventTemplate ∗event)
- void setEnabledWidgets (bool enabled)

Private Attributes

- bool addingNew_

    When we are adding new template.
- FamilyTree ∗ FT_

    Pointer to the family tree.
- Ui::EventTamplatesDialog ∗ ui

    User interface of the Qt framework.

14.11.1 Detailed Description

Dialog showing all event templates and possibly editing them or creating new ones.

14.11.2 Constructor & Destructor Documentation

14.11.2.1 EventTamplatesDialog()

```
EventTamplatesDialog::EventTamplatesDialog (
            FamilyTree ∗ FT,
            QWidget ∗ parent = nullptr )  [explicit]
```

Default constructor.

Parameters

FT	Pointer to the family tree.
parent	The Qt Widget parent.

14.11.2.2 ∼EventTamplatesDialog()

```
EventTamplatesDialog::∼EventTamplatesDialog ( )
```

Default destructor.

14.11.3 Member Function Documentation

14.11.3.1 addDefaults

```
void EventTamplatesDialog::addDefaults ( )  [slot]
```

Add default templates.

14.11.3.2 addNewRole

```
void EventTamplatesDialog::addNewRole ( )  [slot]
```

Add new role to the list.

14.11.3.3 addNewTemplate

```
void EventTamplatesDialog::addNewTemplate ( )  [slot]
```

Show empty boxes to add new template.

14.11.3.4 clear()

```
void EventTamplatesDialog::clear ( )  [private]
```

Clear all boxes.

14.11.3.5 refreshListWidget()

```
void EventTamplatesDialog::refreshListWidget ( )  [private]
```

Refresh list of all relation templates.

14.11.3.6 removeRole

```
void EventTamplatesDialog::removeRole ( )  [slot]
```

Remove role from the list.

14.11.3.7 removeTemplate

```
void EventTamplatesDialog::removeTemplate ( )  [slot]
```

Remove template from the list.

14.11.3.8 saveTemplate [1/2]

```
void EventTamplatesDialog::saveTemplate ( )  [slot]
```

Save template or create a new one.

14.11.3.9 saveTemplate() [2/2]

```
void EventTamplatesDialog::saveTemplate (
             EventTemplate * event )  [private]
```

Overwrite data of the event template by the input from ui.

Parameters

event	Pointer to which template to overwrite.

14.11.3.10 setEnabledWidgets()

```
void EventTamplatesDialog::setEnabledWidgets (
             bool enabled )  [private]
```

Set enabled for all widgets.

Parameters

enabled	If the widgets should be enabled or not.

14.11.3.11 showTemplate

```
void EventTamplatesDialog::showTemplate (
             QTreeWidgetItem * item )  [slot]
```

Show template details.

Parameters

item	Which item is clicked.

14.11.4 Member Data Documentation

14.11.4.1 addingNew_

```
bool EventTamplatesDialog::addingNew_  [private]
```

When we are adding new template.

14.11.4.2 FT_

```
FamilyTree* EventTamplatesDialog::FT_  [private]
```

Pointer to the family tree.

14.11.4.3 ui

```
Ui::EventTamplatesDialog* EventTamplatesDialog::ui  [private]
```

User interface of the Qt framework.

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

- graphics/dialogs.h
- graphics/dialogs.cpp

14.12 EventTemplate Class Reference

Template for events. It specifies whether the event includes a date, a place or if more persons are involved and if they have some special roles.

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

Public Member Functions

- EventTemplate ()

  Default constructor.
- bool containsDate () const
- bool containsPlace () const
- const std::string & getDescription () const
- size_t getId () const
- const std::vector< std::string > & getRoles () const
- const std::string & getTitle () const
- bool hasMorePeopleInvolved () const
- bool hasRoles () const
- void readJson (const Json::Value &value)
- void setContainsDate (bool hasDate)
- void setContainsPlace (bool hasPlace)
- std::string & setDescription ()
- void setId (size_t id)
- void setMorePeopleInvolved (bool morePeople)
- void setRoles (std::vector< std::string > &roles)
- std::string & setTitle ()
- std::string str () const

Private Attributes

- bool containsDate_

  If the event contains a date or not.
- bool containsPlace_

  If the event contains a place or not.
- std::string description_

  Description of the event.
- size_t id_

  Special ID of this template. May change during the execution.
- bool morePeopleInvolved_

  If more people are involved. Otherwise only one is involved.
- std::vector< std::string > roles_

  Vector of special role names.
- std::string title_

  Title of the event.

14.12.1 Detailed Description

Template for events. It specifies whether the event includes a date, a place or if more persons are involved and if they have some special roles.

14.12.2 Constructor & Destructor Documentation

14.12.2.1 EventTemplate()

```
EventTemplate::EventTemplate ( )
```

Default constructor.

14.12.3 Member Function Documentation

14.12.3.1 containsDate()

```
bool EventTemplate::containsDate ( ) const
```

If the event contains a date or not.

Returns

True if it contains a date, false otherwise.

14.12.3.2 containsPlace()

```
bool EventTemplate::containsPlace ( ) const
```

If the event contains a place or not.

Returns

True if the event contains a place, false otherwise.

14.12.3.3 getDescription()

```
const std::string & EventTemplate::getDescription ( ) const
```

Get the description of the event.

Returns

Constant reference to the description.

14.12.3.4 getId()

```
size_t EventTemplate::getId ( ) const
```

Get the ID of this event template.

Returns

ID of this event template.

14.12.3.5 getRoles()

```
const std::vector< std::string > & EventTemplate::getRoles ( ) const
```

Get the vector of special roles.

Returns

Constant reference to the vector of role names.

14.12.3.6 getTitle()

```
const std::string & EventTemplate::getTitle ( ) const
```

Get the title of this event.

Returns

Constant reference to the title.

14.12.3.7 hasMorePeopleInvolved()

```
bool EventTemplate::hasMorePeopleInvolved ( ) const
```

If there is more people involved in this event.

Returns

True if more people are involved.

14.12.3.8 hasRoles()

```
bool EventTemplate::hasRoles ( ) const
```

If the event has special roles.

Returns

True if special roles were given.

14.12.3.9 readJson()

```
void EventTemplate::readJson (
             const Json::Value & value )
```

Read the data from JSON.

Parameters

value	Given JSON value.

14.12.3.10 setContainsDate()

```
void EventTemplate::setContainsDate (
             bool hasDate )
```

Set if the template contains date or not.

Parameters

hasDate	If the template contains a date.

14.12.3.11 setContainsPlace()

```
void EventTemplate::setContainsPlace (
             bool hasPlace )
```

Set if the template contains place or not.

Parameters

hasPlace	If the template contains a place.

14.12.3.12 setDescription()

```
std::string & EventTemplate::setDescription ( )
```

Set the description of the event by giving the reference to the string.

Returns

Reference to the description.

14.12.3.13 setId()

```
void EventTemplate::setId (
             size_t id )
```

Set the ID of this event template.

Parameters

id	New ID.

14.12.3.14 setMorePeopleInvolved()

```
void EventTemplate::setMorePeopleInvolved (
            bool morePeople )
```

Set if more people are involved in this event.

Parameters

morePeople	True if more people are involved.

14.12.3.15 setRoles()

```
void EventTemplate::setRoles (
            std::vector< std::string > & roles )
```

Set the roles.

Parameters

roles	Vector of all role names.

14.12.3.16 setTitle()

```
std::string & EventTemplate::setTitle ( )
```

Set the title of the event by giving the reference to the string.

Returns

Reference to the title.

14.12.3.17 str()

```
std::string EventTemplate::str ( ) const
```

Represent event template as a string.

Returns

Newly constructed string representing the template.

14.12.4 Member Data Documentation

14.12.4.1 containsDate_

```
bool EventTemplate::containsDate_ [private]
```

If the event contains a date or not.

14.12.4.2 containsPlace_

```
bool EventTemplate::containsPlace_ [private]
```

If the event contains a place or not.

14.12.4.3 description_

```
std::string EventTemplate::description_ [private]
```

Description of the event.

14.12.4.4 id_

```
size_t EventTemplate::id_ [private]
```

Special ID of this template. May change during the execution.

14.12.4.5 morePeopleInvolved_

```
bool EventTemplate::morePeopleInvolved_ [private]
```

If more people are involved. Otherwise only one is involved.

14.12.4.6 roles_

```
std::vector<std::string> EventTemplate::roles_ [private]
```

Vector of special role names.

14.12.4.7 title_

```
std::string EventTemplate::title_ [private]
```

Title of the event.

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

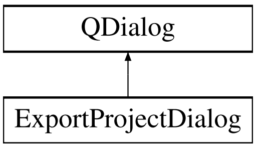
- core/config.h
- core/config.cpp

14.13 ExportProjectDialog Class Reference

Dialog for exporting project.

`#include <dialogs.h>`

Inheritance diagram for ExportProjectDialog:



Public Slots

- void checkPerson ()

  Let the person check box be always checked.
- void exportProject ()

  Export project upon clicking OK.

Public Member Functions

- ExportProjectDialog (FamilyTree ∗FT, QWidget ∗parent=nullptr)
- ∼ExportProjectDialog ()

  Default destructor.

Private Attributes

- FamilyTree ∗ FT_

  Pointer to the family tree.
- Ui::ExportProjectDialog ∗ ui

  Used Qt user interface.

14.13.1 Detailed Description

Dialog for exporting project.

14.13.2 Constructor & Destructor Documentation

14.13.2.1 ExportProjectDialog()

```
ExportProjectDialog::ExportProjectDialog (
            FamilyTree * FT,
            QWidget * parent = nullptr )   [explicit]
```

Default constructor.

Parameters

FT	Pointer to the used Family tree.
parent	The Qt Widget parent.

14.13.2.2 ∼ExportProjectDialog()

```
ExportProjectDialog::∼ExportProjectDialog ( )
```

Default destructor.

14.13.3 Member Function Documentation

14.13.3.1 checkPerson

```
void ExportProjectDialog::checkPerson ( )  [slot]
```

Let the person check box be always checked.

14.13.3.2 exportProject

```
void ExportProjectDialog::exportProject ( )  [slot]
```

Export project upon clicking OK.

14.13.4 Member Data Documentation

14.13.4.1 FT_

```
FamilyTree* ExportProjectDialog::FT_  [private]
```

Pointer to the family tree.

14.13.4.2 ui

```
Ui::ExportProjectDialog* ExportProjectDialog::ui  [private]
```

Used Qt user interface.

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

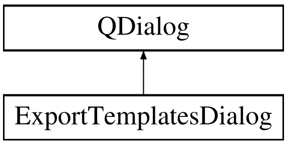
- graphics/dialogs.h
- graphics/dialogs.cpp

14.14 ExportTemplatesDialog Class Reference

Dialog for exporting some of the event and relation templates.

```
#include <dialogs.h>
```

Inheritance diagram for ExportTemplatesDialog:



Public Slots

- void exportTemplates ()

  When OK button was pressed load all selected templates and export them.

Public Member Functions

- ExportTemplatesDialog (FamilyTree ∗FT, QWidget ∗parent=nullptr)
- ∼ExportTemplatesDialog ()

  Default destructor.

Private Attributes

- FamilyTree ∗ FT_

  Pointer to the family tree.
- Ui::ExportTemplatesDialog ∗ ui

  Used Qt user interface.

14.14.1 Detailed Description

Dialog for exporting some of the event and relation templates.

14.14.2 Constructor & Destructor Documentation

14.14.2.1 ExportTemplatesDialog()

```
ExportTemplatesDialog::ExportTemplatesDialog (
            FamilyTree ∗ FT,
            QWidget ∗ parent = nullptr )   [explicit]
```

Default constructor.

Parameters

FT	Pointer to the family tree.
parent	The Qt Widget parent.

14.14.2.2 ∼ExportTemplatesDialog()

```
ExportTemplatesDialog::∼ExportTemplatesDialog ( )
```

Default destructor.

14.14.3 Member Function Documentation

14.14.3.1 exportTemplates

```
void ExportTemplatesDialog::exportTemplates ( )  [slot]
```

When OK button was pressed load all selected templates and export them.

14.14.4 Member Data Documentation

14.14.4.1 FT_

```
FamilyTree∗ ExportTemplatesDialog::FT_  [private]
```

Pointer to the family tree.

14.14.4.2 ui

```
Ui::ExportTemplatesDialog∗ ExportTemplatesDialog::ui  [private]
```

Used Qt user interface.

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

- graphics/dialogs.h
- graphics/dialogs.cpp

14.15 FamilyTree Class Reference

Main class for holding all data of a family tree. Also it acts as a bridge to the core of the application.

```
#include <family_tree.h>
```

Public Member Functions

- FamilyTree ()

  Default constructor.
- Event ∗ addEvent ()
- Event ∗ addEvent (size_t index)
- File ∗ addFile (const std::string &name, FileType type=GENERAL_FILE)
- File ∗ addFile (const std::string &name, size_t index, FileType type=GENERAL_FILE)
- Person ∗ addPerson ()
- Person ∗ addPerson (size_t index)
- Relation ∗ addRelation ()
- Relation ∗ addRelation (size_t index)
- void append (FamilyTree &other)
- std::pair< bool, std::string > checkFileConsistence ()
- void clear ()

  Clear all data and set it for a new project.
- void clearProjectPaths ()

  Clear all project paths except for this one.
- size_t copyFile (const std::string &filePath, FileType type=GENERAL_FILE)
- void createDatabase (const std::string &dirPath)
- size_t createEmptyFile (const std::string &name, FileType type=GENERAL_FILE)
- void enableHelpOnStartup (bool help)
- void exportProject (const std::string &filePath, bool includingEvents, bool includingRelations)
- std::string createResourcesDir (const std::string &target)
- void exportTemplates (const std::vector< size_t > &eventTemplates, const std::vector< size_t > &relTemplates, const std::string &filename)
- std::optional< Event ∗ > getEvent (size_t id)
- const std::map< size_t, std::unique_ptr< Event > > & getEvents () const
- std::optional< File ∗ > getFile (size_t id, FileType type=GENERAL_FILE)
- void getFileAbsolutePath (std::string &absolutePath, size_t id, FileType type=GENERAL_FILE)
- const std::map< size_t, std::unique_ptr< File > > & getFiles (FileType type=GENERAL_FILE)
- std::vector< std::string > getOrphanFiles ()
- Person ∗ getMainPerson ()
- const std::map< size_t, std::unique_ptr< Person > > & getPersons ()
- std::optional< Person ∗ > getPerson (size_t id)
- const std::vector< std::string > & getProjectPaths () const
- std::optional< Relation ∗ > getRelation (size_t id)
- const std::map< size_t, std::unique_ptr< Relation > > & getRelations ()
- std::vector< RelationSuggestion > getRelationSuggestions (size_t relId)
- Settings ∗ getSettings ()
- void getUrlFile (size_t id, std::string &fileUrl, FileType type=GENERAL_FILE)
- bool importProject (const std::string &filePath)
- bool importRootDir (const std::string &dirPath)
- std::pair< bool, size_t > importTemplates (const std::string &filePath)
- bool isDirectorySet ()
- bool isSaved () const
- void log (const std::string &error)
- std::pair< bool, bool > openDatabase (const std::string &dirPath, std::string &errorMessage, std::string &backupFile)
- bool openDatabase (std::string &errorMessage)
- bool openSingleFileDatabase (const std::string &filePath)
- void printHtml (const std::string &dirPath)
- void printMainPersonHtml (const std::string &outputFile)
- void promoteRelation (size_t id)

- void removeBackup ()

  Remove backup file if there is any.
- void removeEvent (size_t id)
- void removeEventTemplate (size_t id)
- size_t removeOrphanFiles ()
- void removePerson ()

  Remove main person from the tree. Also delete all of his appearances.
- void removeRelation (size_t id)
- void removeRelationTemplate (size_t id)
- void renameFile (size_t id, const std::string &newFilename, FileType type)
- bool restoreBackup (const std::string &backupFile, std::string &errorMessage)
- void setMainPerson (Person ∗p)
- void setMainPerson (size_t id)
- void setRelation (size_t relId, size_t pers1Id, size_t pers2Id, size_t templId)
- void setUnsaved ()

  Set unsaved state.
- bool showHelpOnStartup ()
- bool storeDatabase ()
- size_t templatesBasedOnEventTemplate (size_t templateId)
- size_t templatesBasedOnRelationTemplate (size_t templateId)
- void updateEventsWithTemplate (size_t templ)
- bool writeConfig (const std::string &newPath)

Private Member Functions

- bool checkFileTypeConsistence (std::ostream &os, FileType type)
- void getGeneralOrphaFiles (std::vector< std::string > &files, const std::map< size_t, std::unique_ptr< File > > &container, FileType type)
- void getParentSuggestions (std::vector< RelationSuggestion > &suggestions, const Person ∗parent, const Person ∗child, size_t id, Trait trait, const std::set< size_t > &forbiddenPersonParent, const std::set< size_t > &forbiddenPersonChild)
- void getPartnersSuggestions (std::vector< RelationSuggestion > &suggestions, const Person ∗person, const Person ∗second, size_t id, const std::set< size_t > &forbiddenPersons)
- void getSiblingsSuggestions (std::vector< RelationSuggestion > &suggestions, const Person ∗person, const Person ∗second, size_t id, const std::set< size_t > &forbiddenPersons)
- bool loadDatabaseFile (const std::string &file, std::function< void(const Json::Value &)> reader, std::string &errorMessage)
- void printCss (const std::string &dirPath)
- void printFiles (std::ostream &os, const std::map< size_t, std::unique_ptr< File > > &container, const std↩ ::string &label)
- void printEvents (std::ostream &os)
- void printHtmlPerson (std::ostream &os, bool links, Person ∗p)
- void printPeople (std::ostream &os)
- void printRelations (std::ostream &os)
- void readJsonEvents (const Json::Value &value)
- void readJsonFiles (const Json::Value &value, std::map< size_t, std::unique_ptr< File > > &container, const std::string &label, size_t &index)
- void readJsonPeople (const Json::Value &value)
- void readJsonRelations (const Json::Value &value)
- size_t removeGeneralOrphanFile (std::map< size_t, std::unique_ptr< File > > &container, FileType type, const std::string &dir)
- void storeFileDatabase (FileType type)

Private Attributes

- std::map< size_t, std::unique_ptr< Event > > allEvents_

  All the events in the tree.
- std::map< size_t, std::unique_ptr< File > > allFiles_

  All files in app.
- std::map< size_t, std::unique_ptr< File > > allMedia_

  All media in app.
- std::map< size_t, std::unique_ptr< File > > allNotes_

  All notes in app.
- std::map< size_t, std::unique_ptr< Person > > allPersons_

  Map of all persons in family tree.
- std::map< size_t, std::unique_ptr< Relation > > allRelations_

  All relations in the tree.
- size_t event_index_

  Last free index for event. Always start from 1.
- size_t file_index_

  Last free id for file. Always start from 1.
- Person * mainPerson_

  Main person showing as the centre of the tree.
- size_t media_index_

  Last free id for media. Always start from 1.
- size_t note_index_

  Last free id for note. Always start from 1.
- bool openHelp

  If help window should be shown.
- Parser parser_

  Parser for working with locally stored database.
- size_t person_index_

  Last free id for person. Always start from 1.
- std::vector< std::string > projectPaths

  Paths to existing projects.
- size_t relation_index_

  Last free index for relation.
- bool savedChanges_

  If all changes were saved or not.
- Settings settings_

  Settings of the app.

14.15.1 Detailed Description

Main class for holding all data of a family tree. Also it acts as a bridge to the core of the application.

14.15.2 Constructor & Destructor Documentation

14.15.2.1 FamilyTree()

```
FamilyTree::FamilyTree ( )
```

Default constructor.

14.15.3 Member Function Documentation

14.15.3.1 addEvent() [1/2]

```
Event * FamilyTree::addEvent ( )
```

Add new event to the family tree.

Returns

Pointer to the new event.

14.15.3.2 addEvent() [2/2]

```
Event * FamilyTree::addEvent (
              size_t index )
```

Add new event to the family tree with known index.

Parameters

index	Known index.

Returns

Pointer to the new event.

14.15.3.3 addFile() [1/2]

```
File * FamilyTree::addFile (
              const std::string & name,
              FileType type = GENERAL_FILE )
```

Add file to the database.

Parameters

name	Given name of the file.
type	If the file is general file, media or note.

Returns

Pointer to the new file.

14.15.3.4 addFile() [2/2]

```
File * FamilyTree::addFile (
              const std::string & name,
              size_t index,
              FileType type = GENERAL_FILE )
```

Add file to the database with known index.

Parameters

name	Given name of the file.
index	Known index.
type	If the file is general file, media or note.

Returns

Pointer to the new file.

14.15.3.5 addPerson() [1/2]

```
Person * FamilyTree::addPerson ( )
```

Add person to the database.

Returns

Pointer to that new person.

14.15.3.6 addPerson() [2/2]

```
Person * FamilyTree::addPerson (
             size_t index )
```

Add person to the database with known index.

Parameters

index	Known index.

Returns

Pointer to that new person.

14.15.3.7 addRelation() [1/2]

```
Relation * FamilyTree::addRelation ( )
```

Add relation to the family tree.

Returns

Pointer to the new relation.

14.15.3.8 addRelation() [2/2]

```
Relation * FamilyTree::addRelation (
             size_t index )
```

Add relation to the family tree with known index.

Parameters

index	Known index.

Returns

Pointer to the new relation.

14.15.3.9 append()

```
void FamilyTree::append (
             FamilyTree & other )
```

Append the whole project by another one. Note that the second family tree will be corrupted. Because all the indexes must be changed. Though the second tree won't be saved.

Parameters

other	Second Family tree.

14.15.3.10 checkFileConsistence()

```
std::pair< bool, std::string > FamilyTree::checkFileConsistence ( )
```

Check the consistence of stored file.

Returns

Bool if it is consistent and written details of all wrong files.

14.15.3.11 checkFileTypeConsistence()

```
bool FamilyTree::checkFileTypeConsistence (
             std::ostream & os,
             FileType type ) [private]
```

Check file consistence of a single type.

Parameters

os	Where to write error.
type	Which type to look for.

14.15.3.12 clear()

```
void FamilyTree::clear ( )
```

Clear all data and set it for a new project.

14.15.3.13 clearProjectPaths()

```
void FamilyTree::clearProjectPaths ( )
```

Clear all project paths except for this one.

14.15.3.14 copyFile()

```
size_t FamilyTree::copyFile (
             const std::string & filePath,
             FileType type = GENERAL_FILE )
```

Copy an existing file.

Parameters

filePath	Path to the original file on the disk.
type	What type of file it is.

Returns

Id of the new file.

14.15.3.15 createDatabase()

```
void FamilyTree::createDatabase (
             const std::string & dirPath )
```

Create a new empty database.

Parameters

dirPath	Path to the root directory.

14.15.3.16 createEmptyFile()

```
size_t FamilyTree::createEmptyFile (
             const std::string & name,
             FileType type = GENERAL_FILE )
```

Create an empty file.

Parameters

name	Name of the file. Not the real name on disk.
type	What type of file it is.

Returns

Id of the new file.

14.15.3.17 createResourcesDir()

```
std::string FamilyTree::createResourcesDir (
            const std::string & target )
```

Create resource file for diagrams for HTML export.

Parameters

target	Where the resource dir should be (either a path to the directory or a file).

Returns

String representing the path to the resources directory.

14.15.3.18 enableHelpOnStartup()

```
void FamilyTree::enableHelpOnStartup (
            bool help )
```

Switch if the help window should be opened at start up, also write it to configuration.

Parameters

help	If it should be opened or not.

14.15.3.19 exportProject()

```
void FamilyTree::exportProject (
            const std::string & filePath,
            bool includingEvents,
            bool includingRelations )
```

Export this project in one single JSON file.

Parameters

filePath	Path to the export file.
includingEvents	If the export should include events.
includingRelations	If the export should include relations.

14.15.3.20 exportTemplates()

```
void FamilyTree::exportTemplates (
            const std::vector< size_t > & eventTemplates,
            const std::vector< size_t > & relTemplates,
            const std::string & filename )
```

Export given templates to an output file in JSON format.

Parameters

eventTemplates	Vector of event templates.
relTemplates	Vector of relation templates.
filename	Path to the output file.

14.15.3.21 getEvent()

```
std::optional< Event * > FamilyTree::getEvent (
            size_t id )
```

Get pointer to the event.

Parameters

id	Id of the event.

Returns

Get optionally pointer to the event or empty.

14.15.3.22 getEvents()

```
const std::map< size_t, std::unique_ptr< Event > > & FamilyTree::getEvents ( ) const
```

Get constant reference to the container of all events.

Returns

Constant reference to the map of all events.

14.15.3.23 getFile()

```
std::optional< File * > FamilyTree::getFile (
            size_t id,
            FileType type = GENERAL_FILE )
```

Get pointer to given file.

Parameters

id	Id of the file.
type	What type of file to get.

Returns

Optionally pointer to the file or empty.

14.15.3.24 getFileAbsolutePath()

```
void FamilyTree::getFileAbsolutePath (
            std::string & absolutePath,
            size_t id,
            FileType type = GENERAL_FILE )
```

Get the absolute path to the given file.

Parameters

absolutePath	Where will be the absolute paths stored.
id	Id of the file.
type	Which type is used.

14.15.3.25 getFiles()

```
const std::map< size_t, std::unique_ptr< File > > & FamilyTree::getFiles (
            FileType type = GENERAL_FILE )
```

Get constant reference to the vector of all files.

Parameters

type	What type of files the vector has.

Returns

Constant reference to the map of files with given type.

14.15.3.26 getGeneralOrphaFiles()

```
void FamilyTree::getGeneralOrphaFiles (
            std::vector< std::string > & files,
            const std::map< size_t, std::unique_ptr< File > > & container,
            FileType type ) [private]
```

Get all files that would be removed.

Parameters

files	Where to store these files.
container	Which map to use.
type	Which type we are using.

14.15.3.27 getMainPerson()

```
Person * FamilyTree::getMainPerson ( )
```

Get the main person.

Returns

Pointer to the main person.

14.15.3.28 getOrphanFiles()

```
std::vector< std::string > FamilyTree::getOrphanFiles ( )
```

Get all the orphan files.

Returns

Vector of all files, that will be permanently deleted.

14.15.3.29 getParentSuggestions()

```
void FamilyTree::getParentSuggestions (
            std::vector< RelationSuggestion > & suggestions,
            const Person * parent,
            const Person * child,
            size_t id,
            Trait trait,
            const std::set< size_t > & forbiddenPersonParent,
            const std::set< size_t > & forbiddenPersonChild )  [private]
```

Append relation suggestions by a parent relations.

Parameters

suggestions	Container of all suggestions.
parent	Which person is parent.
child	Which person is child.
id	Id of this relation.
trait	Trait of this relation.
forbiddenPersonParent	Which persons are forbidden to bind to with a parent.
forbiddenPersonChild	Which persons are forbidden to bind to with a child.

14.15.3.30 getPartnersSuggestions()

```
void FamilyTree::getPartnersSuggestions (
            std::vector< RelationSuggestion > & suggestions,
            const Person * person,
            const Person * second,
            size_t id,
            const std::set< size_t > & forbiddenPersons )  [private]
```

Append relation suggestions by partners relations.

Parameters

suggestions	Container of all suggestions.
person	Which person is the main one.
second	Which person is the second one.
id	Id of this relation.
forbiddenPersons	Which persons are forbidden to bind to a given person.

14.15.3.31 getPerson()

```
std::optional< Person * > FamilyTree::getPerson (
            size_t id )
```

Get pointer to the person by its id.

Parameters

id	Id of the person.

Returns

Optionally pointer to the person with its id or empty.

14.15.3.32 getPersons()

```
const std::map< size_t, std::unique_ptr< Person > > & FamilyTree::getPersons ( )
```

Get the container of persons.

Returns

Constant reference to the container.

14.15.3.33 getProjectPaths()

```
const std::vector< std::string > & FamilyTree::getProjectPaths ( ) const
```

Get the constant reference to the paths to the roots of projects.

Returns

Constant reference to the vector of all strings representing root directories.

14.15.3.34 getRelation()

```
std::optional< Relation * > FamilyTree::getRelation (
            size_t id )
```

Get pointer to the relation.

Parameters

id	Id of the relation.

Returns

Optionally pointer to the relation or empty.

14.15.3.35 getRelations()

```
const std::map< size_t, std::unique_ptr< Relation > > & FamilyTree::getRelations ( )
```

Get the constant reference to the map of all relations.

Returns

Constant reference to the map of all relations.

14.15.3.36 getRelationSuggestions()

```
std::vector< RelationSuggestion > FamilyTree::getRelationSuggestions (
            size_t relId )
```

Get relation suggestions to the given relation.

Parameters

rel↩ Id	Id of the given relation.

Returns

vector of all suggestions.

14.15.3.37 getSettings()

```
Settings * FamilyTree::getSettings ( )
```

Get pointer to the settings.

Returns

Pointer to the settings.

14.15.3.38 getSiblingsSuggestions()

```
void FamilyTree::getSiblingsSuggestions (
            std::vector< RelationSuggestion > & suggestions,
            const Person * person,
            const Person * second,
            size_t id,
            const std::set< size_t > & forbiddenPersons )  [private]
```

Append relation suggestions by siblings relations.

Parameters

suggestions	Container of all suggestions.
person	Which person is the main one.
second	Which person is the second one.
id	Id of this relation. – This will be changed
forbiddenPersons	Which persons are forbidden to bind to a given person.

14.15.3.39 getUrlFile()

```
void FamilyTree::getUrlFile (
            size_t id,
            std::string & fileUrl,
            FileType type = GENERAL_FILE )
```

Get string representing URL to local file.

Parameters

id	Index of the given file.
fileUrl	URL of the file.
type	If the file is general file, media or note.

14.15.3.40 importProject()

```
bool FamilyTree::importProject (
            const std::string & filePath )
```

Import project.

Parameters

filePath	Path to the file.

Returns

If the import was successful or not.

14.15.3.41 importRootDir()

```
bool FamilyTree::importRootDir (
            const std::string & dirPath )
```

Import project from root directory.

Parameters

dirPath	Path to the root directory.

Returns

If the import was successful or not.

14.15.3.42 importTemplates()

```
std::pair< bool, size_t > FamilyTree::importTemplates (
            const std::string & filePath )
```

Import templates.

Parameters

filePath	Path to the file.

Returns

If the import was successful or not and number of imported templates.

14.15.3.43 isDirectorySet()

```
bool FamilyTree::isDirectorySet ( )
```

If the root directory for the database was already set.

Returns

True if the directory was set.

14.15.3.44 isSaved()

```
bool FamilyTree::isSaved ( ) const
```

If all changes are saved.

Returns

True if all changes are saved, otherwise false.

14.15.3.45 loadDatabaseFile()

```
bool FamilyTree::loadDatabaseFile (
            const std::string & file,
            std::function< void(const Json::Value &)> reader,
            std::string & errorMessage )  [private]
```

Load one single file from the database.

Parameters

file	Which file in database is being loaded.
reader	Which function read the data and load them.
errorMessage	Where will the error message stored.

Returns

If the parsing was successful or not.

14.15.3.46 log()

```
void FamilyTree::log (
            const std::string & error )
```

Log error code to the parser.

Parameters

error	What is the text of the error.

14.15.3.47 openDatabase() [1/2]

```
std::pair< bool, bool > FamilyTree::openDatabase (
            const std::string & dirPath,
            std::string & errorMessage,
            std::string & backupFile )
```

Open a database from its root directory.

Parameters

dirPath	Path to the root directory.
errorMessage	To show what was the potential error.
backupFile	Which file has a backup.

Returns

First if the parsing was successful. Second if there exists backup.

14.15.3.48 openDatabase() [2/2]

```
bool FamilyTree::openDatabase (
            std::string & errorMessage )
```

Read the database without setting dir path.

Parameters

errorMessage	If error occurred show it there.

Returns

If the parsing of the database was successful or not.

14.15.3.49 openSingleFileDatabase()

```
bool FamilyTree::openSingleFileDatabase (
            const std::string & filePath )
```

Open database from a single file.

Parameters

filePath	Path to the file.

Returns

If the parsing was successful or not.

14.15.3.50 printCss()

```
void FamilyTree::printCss (
            const std::string & dirPath )  [private]
```

Print custom CSS.

Parameters

dirPath	Path to the directory for the CSS.

14.15.3.51 printEvents()

```
void FamilyTree::printEvents (
            std::ostream & os )  [private]
```

Print all events.

Parameters

os	Given output stream.

14.15.3.52 printFiles()

```
void FamilyTree::printFiles (
            std::ostream & os,
            const std::map< size_t, std::unique_ptr< File > > & container,
            const std::string & label )  [private]
```

Print all files from vector container.

Parameters

os	Given stream.
container	Given container of files.
label	What is the label in JSON file.

14.15.3.53 printHtml()

```
void FamilyTree::printHtml (
            const std::string & dirPath )
```

Print the whole project to an output directory.

Parameters

dirPath	Path to the directory.

14.15.3.54 printHtmlPerson()

```
void FamilyTree::printHtmlPerson (
            std::ostream & os,
            bool links,
            Person * p )  [private]
```

Print single person to an output in html format.

Parameters

os	Which output stream to use.
links	True if they should be links to other persons.
p	Which person is to be printed.

14.15.3.55 printMainPersonHtml()

```
void FamilyTree::printMainPersonHtml (
            const std::string & outputFile )
```

Print main person in HTML format.

Parameters

outputFile	Output file.

14.15.3.56 printPeople()

```
void FamilyTree::printPeople (
            std::ostream & os )  [private]
```

Print all the people into a stream in a semi-structured JSON format.

Parameters

os	Given stream.

14.15.3.57 printRelations()

```
void FamilyTree::printRelations (
            std::ostream & os )  [private]
```

Print all relations.

Parameters

os	Given output stream.

14.15.3.58 promoteRelation()

```
void FamilyTree::promoteRelation (
            size_t id )
```

Promote a relationship to be the persons main. If possible it will be applied to both persons.

Parameters

id	Id of the relation that is to be promoted.

14.15.3.59 readJsonEvents()

```
void FamilyTree::readJsonEvents (
            const Json::Value & value )  [private]
```

Read and load all events from its JSON value.

Parameters

value	Loaded value from the source file.

14.15.3.60 readJsonFiles()

```
void FamilyTree::readJsonFiles (
            const Json::Value & value,
            std::map< size_t, std::unique_ptr< File > > & container,
            const std::string & label,
            size_t & index )  [private]
```

Read and load meta-data about files from JSON value.

Parameters

value	Loaded value from the source file.
container	Which vector to use for storing files.
label	Which label was used in the JSON file.
index	Indexing of given container.

14.15.3.61 readJsonPeople()

```
void FamilyTree::readJsonPeople (
            const Json::Value & value )  [private]
```

Read and load people from given JSON value.

Parameters

value	Loaded value from the file.

14.15.3.62 readJsonRelations()

```
void FamilyTree::readJsonRelations (
            const Json::Value & value )  [private]
```

Read and load all relations from its JSON value.

Parameters

value	Loaded value from the source file.

14.15.3.63 removeBackup()

```
void FamilyTree::removeBackup ( )
```

Remove backup file if there is any.

14.15.3.64 removeEvent()

```
void FamilyTree::removeEvent (
            size_t id )
```

Remove event from family tree.

Parameters

id	Id of the event.

14.15.3.65 removeEventTemplate()

```
void FamilyTree::removeEventTemplate (
            size_t id )
```

Remove event template and all events based on this template.

Parameters

id	Id of the template.

14.15.3.66 removeGeneralOrphanFile()

```
size_t FamilyTree::removeGeneralOrphanFile (
            std::map< size_t, std::unique_ptr< File > > & container,
            FileType type,
            const std::string & dir )  [private]
```

Remove general orphan file.

Parameters

container	Which container holding files is to be used.
type	Which type will be used.
dir	Which directory in database to use.

Returns

Number of remove files.

14.15.3.67 removeOrphanFiles()

```
size_t FamilyTree::removeOrphanFiles ( )
```

Remove files that no one uses.

Returns

Number of deleted files.

14.15.3.68 removePerson()

```
void FamilyTree::removePerson ( )
```

Remove main person from the tree. Also delete all of his appearances.

14.15.3.69 removeRelation()

```
void FamilyTree::removeRelation (
             size_t id )
```

Remove relation from the family tree.

Parameters

id	Id of the relation which will be removed.

14.15.3.70 removeRelationTemplate()

```
void FamilyTree::removeRelationTemplate (
             size_t id )
```

Remove relation template and all relations based on this template.

Parameters

id	Id of the template.

14.15.3.71 renameFile()

```
void FamilyTree::renameFile (
             size_t id,
             const std::string & newFilename,
             FileType type )
```

Rename given file.

Parameters

id	Id of the file.
newFilename	New file name for the file.
type	What type is the file.

14.15.3.72 restoreBackup()

```
bool FamilyTree::restoreBackup (
             const std::string & backupFile,
             std::string & errorMessage )
```

Restore backup file from the database. Also load the project again.

Parameters

backupFile	Which is the backup file.
errorMessage	If error occurred show it there.

Returns

If the parsing of restored database was successful or not.

14.15.3.73 setMainPerson() [1/2]

```
void FamilyTree::setMainPerson (
             Person * p )
```

Set the main person.

Parameters

p	New main person.

14.15.3.74 setMainPerson() [2/2]

```
void FamilyTree::setMainPerson (
             size_t id )
```

Set the main person.

Parameters

id	Id of the person.

14.15.3.75 setRelation()

```
void FamilyTree::setRelation (
             size_t relId,
             size_t pers1Id,
             size_t pers2Id,
             size_t templId )
```

Set the data about one relation.

Parameters

relId	Id of the relation to be edited.
pers1↩ Id	Id of the first person.
pers2↩ Id	Id of the second person.
templ↩ Id	Id of the used template.

14.15.3.76 setUnsaved()

```
void FamilyTree::setUnsaved ( )
```

Set unsaved state.

14.15.3.77 showHelpOnStartup()

```
bool FamilyTree::showHelpOnStartup ( )
```

If help dialog should be shown.

Returns

True if it should be shown.

14.15.3.78 storeDatabase()

```
bool FamilyTree::storeDatabase ( )
```

Store the whole database to its files.

Returns

If the storing was successful.

14.15.3.79 storeFileDatabase()

```
void FamilyTree::storeFileDatabase (
            FileType type )  [private]
```

Store file database when putting file to database. Also remove all orphan files.

Parameters

type	What type of files are saved.

14.15.3.80 templatesBasedOnEventTemplate()

```
size_t FamilyTree::templatesBasedOnEventTemplate (
             size_t templateId )
```

Get the number of events based on this event template.

Parameters

template↩ Id	Id of the event template.

Returns

Number of events using this event template.

14.15.3.81 templatesBasedOnRelationTemplate()

```
size_t FamilyTree::templatesBasedOnRelationTemplate (
             size_t templateId )
```

Get the number of relations based on this relation template.

Parameters

template↩ Id	Id of the relation template.

Returns

Number of relations using this relation template.

14.15.3.82 updateEventsWithTemplate()

```
void FamilyTree::updateEventsWithTemplate (
             size_t templ )
```

Update events based on the changed template.

Parameters

templ	Id of the changed template.

14.15.3.83 writeConfig()

```
bool FamilyTree::writeConfig (
             const std::string & newPath )
```

Write to the configuration file and before add new one if it is not present.

Parameters

newPath	Path to the new project.

Returns

True if it was added. False if it was already present.

14.15.4 Member Data Documentation

14.15.4.1 allEvents_

```
std::map<size_t, std::unique_ptr<Event> > FamilyTree::allEvents_  [private]
```

All the events in the tree.

14.15.4.2 allFiles_

```
std::map<size_t, std::unique_ptr<File> > FamilyTree::allFiles_  [private]
```

All files in app.

14.15.4.3 allMedia_

```
std::map<size_t, std::unique_ptr<File> > FamilyTree::allMedia_  [private]
```

All media in app.

14.15.4.4 allNotes_

```
std::map<size_t, std::unique_ptr<File> > FamilyTree::allNotes_  [private]
```

All notes in app.

14.15.4.5 allPersons_

```
std::map<size_t, std::unique_ptr<Person> > FamilyTree::allPersons_  [private]
```

Map of all persons in family tree.

14.15.4.6 allRelations_

```
std::map<size_t, std::unique_ptr<Relation> > FamilyTree::allRelations_  [private]
```

All relations in the tree.

14.15.4.7 event_index_

```
size_t FamilyTree::event_index_  [private]
```

Last free index for event. Always start from 1.

14.15.4.8 file_index_

```
size_t FamilyTree::file_index_  [private]
```

Last free id for file. Always start from 1.

14.15.4.9 mainPerson_

```
Person* FamilyTree::mainPerson_  [private]
```

Main person showing as the centre of the tree.

14.15.4.10 media_index_

```
size_t FamilyTree::media_index_  [private]
```

Last free id for media. Always start from 1.

14.15.4.11 note_index_

```
size_t FamilyTree::note_index_  [private]
```

Last free id for note. Always start from 1.

14.15.4.12 openHelp

```
bool FamilyTree::openHelp  [private]
```

If help window should be shown.

14.15.4.13 parser_

```
Parser FamilyTree::parser_  [private]
```

Parser for working with locally stored database.

14.15.4.14 person_index_

```
size_t FamilyTree::person_index_  [private]
```

Last free id for person. Always start from 1.

14.15.4.15 projectPaths

```
std::vector<std::string> FamilyTree::projectPaths  [private]
```

Paths to existing projects.

14.15.4.16 relation_index_

```
size_t FamilyTree::relation_index_  [private]
```

Last free index for relation.

14.15.4.17 savedChanges_

```
bool FamilyTree::savedChanges_  [private]
```

If all changes were saved or not.

14.15.4.18 settings_

```
Settings FamilyTree::settings_  [private]
```

Settings of the app.

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

- core/family_tree.h
- core/family_tree.cpp

14.16 File Class Reference

Meta-data about one file (media or note). Its real path is [id]-[filename].

```
#include <family_tree_items.h>
```

Public Member Functions

- File ()

    Default constructor.
- const std::string & getFilename () const
- size_t getId () const
- std::string getRealName () const
- void readJson (const Json::Value &value)
- void setFilename (const std::string &fileName)
- void setId (size_t id)

Private Attributes

- std::string filename_

    File name.
- size_t id_

    Id of the file.

14.16.1 Detailed Description

Meta-data about one file (media or note). Its real path is [id]-[filename].

14.16.2 Constructor & Destructor Documentation

14.16.2.1 File()

```
File::File ( )
```

Default constructor.

14.16.3 Member Function Documentation

14.16.3.1 getFilename()

```
const std::string & File::getFilename ( ) const
```

Get the file name.

Returns

Constant reference to the file name.

14.16.3.2 getId()

```
size_t File::getId ( ) const
```

Get the id of the file.

Returns

Id of the file.

14.16.3.3 getRealName()

```
std::string File::getRealName ( ) const
```

Get the real name of the file. Which is [id]-[filename]

Returns

The real name of the file.

14.16.3.4 readJson()

```
void File::readJson (
            const Json::Value & value )
```

Read the file meta-data from JSON value.

Parameters

value	The JSON value of the file.

14.16.3.5 setFilename()

```
void File::setFilename (
            const std::string & fileName )
```

Set the file name.

Parameters

fileName	Given name of the file.

14.16.3.6 setId()

```
void File::setId (
            size_t id )
```

Set the id of the file.

Parameters

id	Given id.

14.16.4 Member Data Documentation

14.16.4.1 filename_

```
std::string File::filename_ [private]
```

File name.

14.16.4.2 id_

```
size_t File::id_ [private]
```

Id of the file.

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

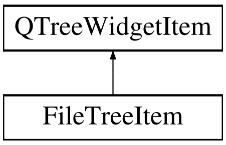
- core/family_tree_items.h
- core/family_tree_items.cpp

14.17 FileTreeItem Class Reference

TreeWidget Item for general file.

```
#include <graphics_items.h>
```

Inheritance diagram for FileTreeItem:



Public Member Functions

- FileTreeItem (VirtualDrive ∗drive, FamilyTree ∗FT, const QIcon &icon, FileType type)
- FileTreeItem (size_t id, FamilyTree ∗FT, const QIcon &icon, FileType type)
- void addFile (size_t id, const QIcon &icon)
- void addSubDrive (const QString &folder)
- VirtualDrive ∗ getDrive ()
- size_t getId ()
- bool isDirectory ()
- bool removeFolder ()
- bool rename (const QString &name)

Private Member Functions

- void removeFile (size_t id)

Private Attributes

- VirtualDrive ∗ drive_

  Drive presented in this folder.
- size_t fileId_

  Id of the shown file.
- FamilyTree ∗ FT_

  Pointer to the family tree.
- bool isDir_

  If the file is directory or not.
- FileType type_

  Type of the file.

14.17.1 Detailed Description

TreeWidget Item for general file.

14.17.2 Constructor & Destructor Documentation

14.17.2.1 FileTreeItem() [1/2]

```
FileTreeItem::FileTreeItem (
             VirtualDrive ∗ drive,
             FamilyTree ∗ FT,
             const QIcon & icon,
             FileType type )
```

Construct this drive and all sub-drives.

Parameters

drive	Which drive is presented in this item.
FT	Pointer to the family tree.
icon	Which icon to show beside the text.
type	Type of the file to show.

14.17.2.2 FileTreeItem() [2/2]

```
FileTreeItem::FileTreeItem (
             size_t id,
             FamilyTree ∗ FT,
             const QIcon & icon,
             FileType type )
```

Constructor for a new file.

Parameters

id	Id of the file.
FT	Pointer to the family tree.
icon	Which icon is shown besides files.
type	What type of files are shown.

14.17.3 Member Function Documentation

14.17.3.1 addFile()

```
void FileTreeItem::addFile (
             size_t id,
             const QIcon & icon )
```

Add file to this folder.

Parameters

id	Id of the file.
icon	Which icon to show besides the text.

14.17.3.2 addSubDrive()

```
void FileTreeItem::addSubDrive (
             const QString & folder )
```

Add sub drive to this drive.

Parameters

folder	The name of the sub-folder.

14.17.3.3 getDrive()

```
VirtualDrive * FileTreeItem::getDrive ( )
```

Get the pointer to the drive.

Returns

Pointer to the drive.

14.17.3.4 getId()

```
size_t FileTreeItem::getId ( )
```

Get the id of the file.

Returns

File id.

14.17.3.5 isDirectory()

```
bool FileTreeItem::isDirectory ( )
```

If te file is directory or not.

Returns

True if it is directory, false otherwise.

14.17.3.6 removeFile()

```
void FileTreeItem::removeFile (
             size_t id )  [private]
```

Remove file in this drive.

Parameters

id	Id of the file.

14.17.3.7 removeFolder()

```
bool FileTreeItem::removeFolder ( )
```

Remove recursively all directories and files.

Returns

True if it is the root directory, false otherwise.

14.17.3.8 rename()

```
bool FileTreeItem::rename (
             const QString & name )
```

When this item is selected for renaming.

Parameters

name	Name for this folder.

Returns

True if this item was file, false otherwise.

14.17.4 Member Data Documentation

14.17.4.1 drive_

```
VirtualDrive* FileTreeItem::drive_  [private]
```

Drive presented in this folder.

14.17.4.2 fileId_

```
size_t FileTreeItem::fileId_  [private]
```

Id of the shown file.

14.17.4.3 FT_

```
FamilyTree* FileTreeItem::FT_  [private]
```

Pointer to the family tree.

14.17.4.4 isDir_

```
bool FileTreeItem::isDir_  [private]
```

If the file is directory or not.

14.17.4.5 type_

```
FileType FileTreeItem::type_  [private]
```

Type of the file.

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

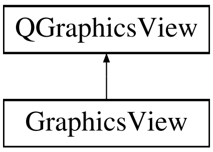
- graphics/graphics_items.h
- graphics/graphics_items.cpp

14.18 GraphicsView Class Reference

Class for creating QGraphicsView with using wheel for zoom in and out. And grabing by mouse moves the view.

```
#include <graphics_items.h>
```

Inheritance diagram for GraphicsView:



Public Member Functions

- GraphicsView (QGraphicsScene ∗scene, QWidget ∗parent=nullptr)

Protected Member Functions

- void wheelEvent (QWheelEvent ∗event) override

14.18.1 Detailed Description

Class for creating QGraphicsView with using wheel for zoom in and out. And grabing by mouse moves the view.

14.18.2 Constructor & Destructor Documentation

14.18.2.1 GraphicsView()

```
GraphicsView::GraphicsView (
             QGraphicsScene ∗ scene,
             QWidget ∗ parent = nullptr )   [explicit]
```

Default constructor.

Parameters

scene	Which scene it is using.
parent	Which widget will be showing the view.

14.18.3 Member Function Documentation

14.18.3.1 wheelEvent()

```
void GraphicsView::wheelEvent (
             QWheelEvent ∗ event )   [override], [protected]
```

Override the wheel event for scrolling.

Parameters

event	When scrolling occurrred.

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

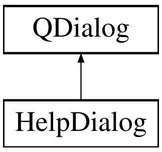
- graphics/graphics_items.h
- graphics/graphics_items.cpp

14.19 HelpDialog Class Reference

Dialog with text edit to show help or other texts.

`#include <dialogs.h>`

Inheritance diagram for HelpDialog:



Public Slots

- void openHelp (bool help)

Public Member Functions

- HelpDialog (FamilyTree ∗FT, QWidget ∗parent=nullptr)
- ∼HelpDialog ()

  Default destructor.

Private Attributes

- FamilyTree ∗ FT_

  Pointer to the family tree.
- Ui::HelpDialog ∗ ui

  User interface of the Qt framework.

14.19.1 Detailed Description

Dialog with text edit to show help or other texts.

14.19.2 Constructor & Destructor Documentation

14.19.2.1 HelpDialog()

```
HelpDialog::HelpDialog (
            FamilyTree * FT,
            QWidget * parent = nullptr )  [explicit]
```

Default constructor.

Parameters

FT	Pointer to the family tree.
parent	The Qt Widget parent.

14.19.2.2 ∼HelpDialog()

```
HelpDialog::∼HelpDialog ( )
```

Default destructor.

14.19.3 Member Function Documentation

14.19.3.1 openHelp

```
void HelpDialog::openHelp (
             bool help )  [slot]
```

If help dialog should be opened at start-up.

Parameters

help	Forwarding of check box in UI. True for not showing.

14.19.4 Member Data Documentation

14.19.4.1 FT_

```
FamilyTree* HelpDialog::FT_  [private]
```

Pointer to the family tree.

14.19.4.2 ui

```
Ui::HelpDialog* HelpDialog::ui  [private]
```

User interface of the Qt framework.

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

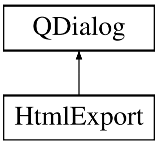
- graphics/dialogs.h
- graphics/dialogs.cpp

14.20 HtmlExport Class Reference

Dialog for exporting HTML for one person and all persons.

`#include <dialogs.h>`

Inheritance diagram for HtmlExport:



Public Slots

- void allAboveGenerations (bool all)
- void allBelowGenerations (bool all)
- void exportAllHtml ()

  Export all persons into html.
- void exportMainHtml ()

  Export only main person to the html.

Public Member Functions

- HtmlExport (FamilyTree ∗FT, bool all, int up, int down, QWidget ∗parent=nullptr)
- ∼HtmlExport ()

  Default destructor.

Private Attributes

- FamilyTree ∗ FT_

  Pointer to the family tree.
- Ui::HtmlExport ∗ ui

  Used Qt user interface.

14.20.1 Detailed Description

Dialog for exporting HTML for one person and all persons.

14.20.2 Constructor & Destructor Documentation

14.20.2.1 HtmlExport()

```
HtmlExport::HtmlExport (
            FamilyTree * FT,
            bool all,
            int up,
            int down,
            QWidget * parent = nullptr )
```

Default constructor.

Parameters

FT	Pointer to the used Family tree.
all	If the HTML export should include all persons or just the main person.
up	How many generations upwards are being exported as of now.
down	How many generations downwards are being exported as of now.
parent	The Qt Widget parent.

14.20.2.2 ∼HtmlExport()

```
HtmlExport::∼HtmlExport ( )
```

Default destructor.

14.20.3 Member Function Documentation

14.20.3.1 allAboveGenerations

```
void HtmlExport::allAboveGenerations (
            bool all )  [slot]
```

If all generations upward option was chosen or not.

Parameters

all	True if it was checked.

14.20.3.2 allBelowGenerations

```
void HtmlExport::allBelowGenerations (
            bool all )  [slot]
```

If all generations downward option was chosen or not.

Parameters

all	True if it was checked.

14.20.3.3 exportAllHtml

```
void HtmlExport::exportAllHtml ( )  [slot]
```

Export all persons into html.

14.20.3.4 exportMainHtml

```
void HtmlExport::exportMainHtml ( ) [slot]
```

Export only main person to the html.

14.20.4 Member Data Documentation

14.20.4.1 FT_

```
FamilyTree* HtmlExport::FT_ [private]
```

Pointer to the family tree.

14.20.4.2 ui

```
Ui::HtmlExport* HtmlExport::ui [private]
```

Used Qt user interface.

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

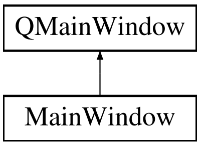
- graphics/dialogs.h
- graphics/dialogs.cpp

14.21 MainWindow Class Reference

Class representing main window using Qt library.

```
#include <mainwindow.h>
```

Inheritance diagram for MainWindow:



Public Slots

- void addNewEvent ()

  Show dialog for adding event.
- void addNewFile ()

  Show dialog for adding file.
- void addNewFileFolder ()

  Add new folder to the file tree.
- void addNewMedia ()

  Show dialog for adding media.
- void addNewMediaFolder ()

  Add new folder to the media tree.
- void addNewNote ()

  Show dialog for adding note.
- void addNewNoteFolder ()

  Add new folder to the note tree.
- void addNewPerson ()

  Add new person to the family tree.
- void addNewRelation ()

  Add new relation to the person by opening dialog window.
- void allAboveGenerations (bool selected)
- void allBelowGenerations (bool selected)
- void changeGenerationsDown (int value)
- void changeGenerationsUp (int value)
- void checkStoreRefreshPersonsInfo ()

  If any value in persons info tab was changed, check its value, store it and refresh ui elements.
- void clearProjectFilter ()

  Clear the project filter.
- void clearProjectPaths ()

  Clear project paths.
- void disableFileFolderPrompt ()

  Hide new folder edit for files and cancel button.
- void disableMediaFolderPrompt ()

  Hide new folder edit for media and cancel button.
- void disableNoteFolderPrompt ()

  Hide new folder edit for notes and cancel button.
- void editEvent (QTreeWidgetItem ∗item)
- void editRelation (QTreeWidgetItem ∗item)
- void editTag (QTreeWidgetItem ∗item=nullptr)
- void enableFileFolderPrompt ()

  Show new folder edit for files and cancel button.
- void enableMediaFolderPrompt ()

  Show new folder edit for media and cancel button.
- void enableNoteFolderPrompt ()

  Show new folder edit for notes and cancel button.
- void exportCloseFamilyDiagramDialog ()

  Open dialog for exporting close family diagram.
- void exportHtml ()

  Export whole HTML project to an output directory by opening a dialog.
- void exportMainHtml ()

  Export only main person html by opening a dialog.

- void exportProject ()

  Open dialog for exporting project.
- void exportTemplates ()

  Open dialog for exporting templates.
- void exportTreeDiagramDialog ()

  Open dialog for exporting family tree diagram.
- void filterProjectItems ()

  Filter only matching items in project.
- void findPerson ()

  Focus find in project view.
- void fitTreeInView ()

  Fit the tree view inside.
- void importProject ()

  Choose file to import from other project.
- void importRootDir ()

  Choose root directory of other project to import from.
- void importTemplates ()

  Choose file to import templates.
- void interractWithProjectItem (QTreeWidgetItem ∗item)
- void openBirthDateDialog ()

  Open dialog for custom Birth Date.
- void openDeathDateDialog ()

  Open dialog for custom Death Date.
- void openEventTemplatesDialog ()

  Open event templates dialog.
- void openExistingProject ()

  Open existing root directory of a project.
- void openGraphicSettings ()

  Open dialog for graphic settings.
- void openNewFamilyTreeProject ()

  Open new empty project.
- void openRelationTemplatesDialog ()

  Open relation templates dialog.
- void openSelectedFile (QTreeWidgetItem ∗item)
- void openSelectedMedia (QTreeWidgetItem ∗item)
- void openSelectedNote (QTreeWidgetItem ∗item)
- void promoteRelation ()

  Promote current relation.
- void refreshDates ()

  Refresh all dates in persons info tab.
- void refreshEventTab ()

  Refresh list with events.
- void refreshGraphics ()

  Refresh all graphics.
- void refreshProjectView ()

  Refresh the whole project that is shown.
- void refreshRelationTab ()

  Refresh relations lists.
- void refreshTabsWithFiles ()

  Refresh all files tabs.
- void refreshTagsList ()

Refresh shown tags.

- void removeEvent ()

  Remove event from family tree.
- void removeFileFolder ()

  Remove file folder.
- void removeMediaFolder ()

  Remove media folder.
- void removeNoteFolder ()

  Remove note folder.
- void removePerson ()

  Remove main person.
- void removeRelation ()

  Remove current relation.
- void removeTag ()

  Delete current tag selected on the person.
- void renameFile ()

  Rename file or folder with files.
- void renameMedia ()

  Rename media or folder with media.
- void renameNote ()

  Rename note or folder with notes.
- bool saveDatabase ()
- void saveGraphicsSetting (QColor probandColor, QColor promotedColor, QColor standardColor, QColor highlightedColor, int sizeX, int sizeY, QFont font, QColor lineColor, QColor textColor, int lineWidth, int rounding)
- void selectOlderGenerationOnly ()

  De-select all other relation boxes other than older.
- void selectSameGenerationOnly ()

  De-select all other relation boxes other than same.
- void selectYoungerGenerationOnly ()

  De-select all other relation boxes other than younger.
- void setEnabledDeathAtributes (bool disabled)
- void setEventTabVisible ()

  Hide or show the event tab.
- void setFileTabVisible ()

  Hide or show the file tab.
- void setGeneralTabVisible ()

  Hide or show the general tab.
- void setMainPerson (size_t personId)
- void setMediaTabVisible ()

  Hide or show the media tab.
- void setNoteTabVisible ()

  Hide or show the note tab.
- void setRelationTabVisible ()

  Hide or show the relation tab.
- void setTabEast ()

  Set the setting of the tab to east.
- void setTabNorth ()

  Set the setting of the tab to north.
- void setTabSouth ()

  Set the setting of the tab to south.

- void setTabWest ()

  Set the setting of the tab to west.
- void showHelp ()

  Show generic help.
- void showRenameFile ()

  Show prompt for renaming file.
- void showRenameMedia ()

  Show prompt for renaming media.
- void showRenameNote ()

  Show prompt for renaming note.
- void splitterChanged ()

  When the splitter changed.
- void switchNewTreeProject ()

  Save tree and create new empty tree to show.
- void zoomIn ()

  Zoom in on the family tree view.
- void zoomOut ()

  Zoom out of the family tree view.

Public Member Functions

- MainWindow (QWidget ∗parent=nullptr)
- ∼MainWindow ()

  Default destructor, which deletes all dynamic attributes.
- bool exportCloseFamilyDiagram (double scale, const std::string &filename=EMPTY_STRING)
- bool exportTreeDiagram (double scale, const std::string &filename=EMPTY_STRING)
- std::pair< int, int > setNumberOfGenerations (int up, int down)

Static Public Member Functions

- static void openExternalFile (const std::string &fileURL)

Protected Member Functions

- void closeEvent (QCloseEvent ∗event)

Private Member Functions

- void addGeneralFile (FileType type, const std::vector< std::string > &extensions, FileTreeItem ∗item)
- void addGeneralFolder (FileType type)
- void clearUi ()

  Clear all UI elements.
- void connectAllSlots ()

  Connect all slots to their signals.
- void editEvent (size_t id)
- void editRelation (size_t id)
- void enableGeneralFolderPrompt (bool visible, FileType type)
- void enableGeneralFolderRenamePrompt (FileType type)
- bool exportGeneralDiagram (const QString &filename, QGraphicsScene ∗gscene, double scale=1)

- void drawCloseContainer (const std::vector< Relation * > &container, qreal sizeX, qreal sizeY, qreal topLeftX, qreal topLeftY, qreal lineX1, qreal lineY1, qreal lineX2, qreal lineY2, size_t partner, size_t mother, size_t father, int gap, bool movex, bool movey)
- void drawCloseFamily ()
- void drawFamilyTree ()
- void drawGeneralGenerations (Person *p, qreal width, qreal midx, qreal midy, int horizontalGap, int verticalGap, size_t barrier, const std::vector< size_t > &container, bool up, std::function< std::pair< size_t, size_t >(Person *, size_t, size_t)> numberFunc, std::function< void(Person *, qreal, qreal, qreal, int, int, size_t)> drawFunc)
- void drawOlderGenerations (Person *p, qreal width, qreal midx, qreal midy, int horizontalGap, int verticalGap, size_t barrier)
- void drawPartner (qreal canvasMidX, qreal canvasMidY, int horizontalGap)
- void drawYoungerGenerations (Person *p, qreal width, qreal midx, qreal midy, int horizontalGap, int verticalGap, size_t barrier)
- std::pair< size_t, size_t > generalGenerationNumber (Person *p, size_t soFar, size_t barrier, const std::vector< size_t > &container, bool up, std::function< std::pair< size_t, size_t >(Person *, size_t, size_t)> func)
- void loadSettings ()

  Load the setting from configuration.
- std::pair< size_t, size_t > olderGenerationNumber (Person *p, int soFar, size_t barrier)
- void openCustomDateDialog (WrappedDate *date)
- void initializeProjectSubView (ProjectItem *parent, Person *person)
- void initializeProjectView ()

  Initialize project view when loading new family tree.
- void openGeneralFile (QTreeWidgetItem *item, FileType type)
- void openGeneralProject (const std::string &dirPath)
- void refreshPersonsInfoTab ()

  Refresh basic information about main person.
- void refreshProjectEventsView (ProjectItem *item, Person *person)
- void refreshProjectGeneralFilesView (const QIcon &containerIcon, const QString &containerName, const QIcon &fileIcon, FileType type, ProjectItem *item, Person *person)
- void refreshProjectFilesView (ProjectItem *item, Person *person)
- void refreshProjectMediaView (ProjectItem *item, Person *person)
- void refreshProjectNoteView (ProjectItem *item, Person *person)
- void refreshProjectRelationsView (ProjectItem *item, Person *person)
- void refreshUi ()

  Refresh the whole UI (tabs, project and graphics).
- void removeGeneralFolder (FileType type)
- void renameGeneralFileItem (FileType type)
- void savePersonsInfo ()

  Save all basic information about main person.
- void setUiElements ()

  Set all additional static widgets of the user interface.
- void showGeneralFiles (QTreeWidget *widget, VirtualDrive *root, FileType type)
- std::pair< size_t, size_t > youngerGenerationNumber (Person *p, size_t soFar, size_t barrier)

Private Attributes

- int borderRadius_

  Radius of the boxes.
- QFont font_

  Used font for view.
- FamilyTree FT

Family tree.
- int genSizeDown_

  How many generations to show in tree view down.
- int genSizeUp_

  How many generations to show in tree view up.
- QColor highlightedColor_

  Color for highlighted boxes.
- QPen linePen_

  Pen used for drawing lines.
- QColor probandColor_

  Color of proband.
- QGraphicsScene ∗ scene

  Scene shown on the graphical view.
- int sizeX_

  X size of the person box.
- int sizeY_

  Y size of the person box.
- QColor promotedColor_

  Color for promoted father, mother and partner.
- QColor standardColor_

  Color for other relations.
- QPen textPen_

  Pen used for texts.
- QGraphicsScene ∗ treeScene

  Tree view scene;.
- GraphicsView ∗ treeView

  Tree view graphics view.
- Ui::MainWindow ∗ ui

  Qt UI framework.

14.21.1 Detailed Description

Class representing main window using Qt library.

14.21.2 Constructor & Destructor Documentation

14.21.2.1 MainWindow()

```
MainWindow::MainWindow (
              QWidget * parent = nullptr )   [explicit]
```

Default constructor.

Parameters

parent	Qt parent.

14.21.2.2 ∼MainWindow()

```
MainWindow::∼MainWindow ( )
```

Default destructor, which deletes all dynamic attributes.

14.21.3 Member Function Documentation

14.21.3.1 addGeneralFile()

```
void MainWindow::addGeneralFile (
            FileType type,
            const std::vector< std::string > & extensions,
            FileTreeItem * item )  [private]
```

Open add file dialog for a person. This is as general as it gets.

Parameters

type	Type of the file, that will be added.
extensions	Which extensions will be shown for creating empty file.
item	Which item is selected to put file.

14.21.3.2 addGeneralFolder()

```
void MainWindow::addGeneralFolder (
            FileType type )  [private]
```

Add folder to its type tree widget.

Parameters

type	What widget it should use.

14.21.3.3 addNewEvent

```
void MainWindow::addNewEvent ( )  [slot]
```

Show dialog for adding event.

14.21.3.4 addNewFile

```
void MainWindow::addNewFile ( )  [slot]
```

Show dialog for adding file.

14.21.3.5 addNewFileFolder

```
void MainWindow::addNewFileFolder ( )  [slot]
```

Add new folder to the file tree.

14.21.3.6 addNewMedia

```
void MainWindow::addNewMedia ( )  [slot]
```

Show dialog for adding media.

14.21.3.7 addNewMediaFolder

```
void MainWindow::addNewMediaFolder ( )  [slot]
```

Add new folder to the media tree.

14.21.3.8 addNewNote

```
void MainWindow::addNewNote ( )  [slot]
```

Show dialog for adding note.

14.21.3.9 addNewNoteFolder

```
void MainWindow::addNewNoteFolder ( )  [slot]
```

Add new folder to the note tree.

14.21.3.10 addNewPerson

```
void MainWindow::addNewPerson ( )  [slot]
```

Add new person to the family tree.

14.21.3.11 addNewRelation

```
void MainWindow::addNewRelation ( )  [slot]
```

Add new relation to the person by opening dialog window.

14.21.3.12 allAboveGenerations

```
void MainWindow::allAboveGenerations (
             bool selected )  [slot]
```

Set to show all generations above.

Parameters

selected	If all generations should be shown.

14.21.3.13 allBelowGenerations

```
void MainWindow::allBelowGenerations (
            bool selected )  [slot]
```

Set to show all generations below.

Parameters

selected	If all generations should be shown.

14.21.3.14 changeGenerationsDown

```
void MainWindow::changeGenerationsDown (
            int value )  [slot]
```

Change the number of shown relations below.

Parameters

value	How many generations to show.

14.21.3.15 changeGenerationsUp

```
void MainWindow::changeGenerationsUp (
            int value )  [slot]
```

Change the number of shown relations above.

Parameters

value	How many generations to show.

14.21.3.16 checkStoreRefreshPersonsInfo

```
void MainWindow::checkStoreRefreshPersonsInfo ( )  [slot]
```

If any value in persons info tab was changed, check its value, store it and refresh ui elements.

14.21.3.17 clearProjectFilter

```
void MainWindow::clearProjectFilter ( )  [slot]
```

Clear the project filter.

14.21.3.18 clearProjectPaths

```
void MainWindow::clearProjectPaths ( )  [slot]
```

Clear project paths.

14.21.3.19 clearUi()

```
void MainWindow::clearUi ( )  [private]
```

Clear all UI elements.

14.21.3.20 closeEvent()

```
void MainWindow::closeEvent (
             QCloseEvent * event )  [protected]
```

When closing the window check if the project is saved or not.

Parameters

event	Called event.

14.21.3.21 connectAllSlots()

```
void MainWindow::connectAllSlots ( )  [private]
```

Connect all slots to their signals.

14.21.3.22 disableFileFolderPrompt

```
void MainWindow::disableFileFolderPrompt ( )  [slot]
```

Hide new folder edit for files and cancel button.

14.21.3.23 disableMediaFolderPrompt

```
void MainWindow::disableMediaFolderPrompt ( )  [slot]
```

Hide new folder edit for media and cancel button.

14.21.3.24 disableNoteFolderPrompt

```
void MainWindow::disableNoteFolderPrompt ( )  [slot]
```

Hide new folder edit for notes and cancel button.

14.21.3.25 drawCloseContainer()

```
void MainWindow::drawCloseContainer (
            const std::vector< Relation * > & container,
            qreal sizeX,
            qreal sizeY,
            qreal topLeftX,
            qreal topLeftY,
            qreal lineX1,
            qreal lineY1,
            qreal lineX2,
            qreal lineY2,
            size_t partner,
            size_t mother,
            size_t father,
            int gap,
            bool movex,
            bool movey )  [private]
```

Draw generalized cell, line and all inside person items in close family view. Like this:

```
<-----sizeX--------->

topLeftX & topLeftY
v
+-------cell--------+   ^
|gap|box|gap|box|gap|  sizeY
+-------------------+   v
          | line to middle box
```

Parameters

container	Container of persons which will be drawn.
sizeX	Size of the cell in the x axis.
sizeY	Size of the cell in the y axis.
topLeftX	Top left point of the cell in the x axis.
topLeftY	Top left point of the cell in the y axis.
lineX1	Starting point of the line in x axis.
lineY1	Starting point of the line in y axis.
lineX2	Ending point of the line in x axis.
lineY2	Ending point of the line in y axis.
partner	Index of partner to use special color.
mother	Index of mother to use special color.
father	Index of father to use special color.
gap	Gap which is between boxes in cell.
movex	Whether to move boxes on the x line.
movey	Whether to move boxes on the y line.

14.21.3.26 drawCloseFamily()

```
void MainWindow::drawCloseFamily ( )  [private]
```

Draw close family to the graphics.

```
   +------cell---------+
   |gap|box|gap|box|gap|
   +-------------------+
+---+<-cell  |
|gap|    +-------+ <-- box
|box|----|proband|
|gap|    +-------+
+---+        |
   +------cell---------+
   |gap|box|gap|box|gap|
   +-------------------+
```

14.21.3.27 drawFamilyTree()

```
void MainWindow::drawFamilyTree ( )  [private]
```

Draw family tree to the corresponding person.

```
  ....            ....
   \/              \/
+--box--+    +--box--+
|father |    |mother |
+-------+    +-------+
        \     /
       +--box--+
       |proband|
       +-------+
        |     |
+--box--+    +--box--+
|child  |    |child  |
+-------+    +-------+
   /|\             |
          ...
```

14.21.3.28 drawGeneralGenerations()

```
void MainWindow::drawGeneralGenerations (
               Person * p,
               qreal width,
               qreal midx,
               qreal midy,
               int horizontalGap,
               int verticalGap,
               size_t barrier,
               const std::vector< size_t > & container,
               bool up,
               std::function< std::pair< size_t, size_t >(Person *, size_t, size_t)> numberFunc,
               std::function< void(Person *, qreal, qreal, qreal, int, int, size_t)> drawFunc )
[private]
```

Draw either top or bottom generation in tree.

Parameters

p	Current person.
width	Width of the place where will be the relations.

Parameters

midx	What is the middle in the x axis of the place.
midy	What is the middle in the y axis of the place.
horizontalGap	Min gap between boxes in the x axis.
verticalGap	Gap between generations.
barrier	How more generations to show.
container	Vector of relations to draw.
up	If we are drawing upwards or downwards.
numberFunc	Function for computing the widths and depths of the trees.
drawFunc	Function to recursively call for drawing.

14.21.3.29 drawOlderGenerations()

```
void MainWindow::drawOlderGenerations (
            Person * p,
            qreal width,
            qreal midx,
            qreal midy,
            int horizontalGap,
            int verticalGap,
            size_t barrier )  [private]
```

Draw one family tree box and recursively make other.

```
+-----------------------+
|<-xfrom  ....          | Box will be in the middle.
|<-yfrom  ....          | Then it will call recursively on the top half the space.
+-----------------------+
<--------space---------->
```

Parameters

p	Which person we are drawing.
width	What is the x size where should the box take place.
midx	What is the middle in the x axis of the place.
midy	What is the middle in the y axis of the place.
horizontalGap	Min gap between boxes in the x axis.
verticalGap	Gap between generations.
barrier	How more generations to show.

14.21.3.30 drawPartner()

```
void MainWindow::drawPartner (
            qreal canvasMidX,
            qreal canvasMidY,
            int horizontalGap )  [private]
```

Draw partner (to the left of the main person) to the family tree if there is one.

Parameters

canvasMidX	Centre-point of the canvas.
canvasMidY	Centre-point of the canvas.
horizontalGap	Gap in between persons boxes.

14.21.3.31 drawYoungerGenerations()

```
void MainWindow::drawYoungerGenerations (
             Person * p,
             qreal width,
             qreal midx,
             qreal midy,
             int horizontalGap,
             int verticalGap,
             size_t barrier )  [private]
```

Draw all younger relations to the tree diagram.

```
+---+     +---+     +---+
|box|     |box|     |box|
+---+     +---+     +---+
  .        /  \       .
.....  +---+  +---+ .....
.   .  |box|  |box| .   .
.....  +---+  +---+ .....
```

Parameters

p	Current person which is above.
width	Width of the place where will be the children drawn.
midx	What is the middle in the x axis of the place for children.
midy	What is the middle in the y axis of the place for children.
horizontalGap	Min gap between boxes in the x axis.
verticalGap	Gap between generations.
barrier	How more generations to show.

14.21.3.32 editEvent [1/2]

```
void MainWindow::editEvent (
             QTreeWidgetItem * item )  [slot]
```

Edit event when clicking on it by opening a dialog window.

Parameters

item	Which item was clicked.

14.21.3.33 editEvent() [2/2]

```
void MainWindow::editEvent (
            size_t id )  [private]
```

Edit event by its id.

Parameters

id	Id of the event.

14.21.3.34 editRelation [1/2]

```
void MainWindow::editRelation (
            QTreeWidgetItem * item )  [slot]
```

Edit relation when clicking on it by opening a dialog window.

Parameters

item	Which item was clicked.

14.21.3.35 editRelation() [2/2]

```
void MainWindow::editRelation (
            size_t id )  [private]
```

Edit relation by its id.

Parameters

id	Id of the relation.

14.21.3.36 editTag

```
void MainWindow::editTag (
            QTreeWidgetItem * item = nullptr )  [slot]
```

Edit tag when clicking on it or creating new one.

Parameters

item	Which item was selected, if any.

14.21.3.37 enableFileFolderPrompt

```
void MainWindow::enableFileFolderPrompt ( )  [slot]
```

Show new folder edit for files and cancel button.

14.21.3.38 enableGeneralFolderPrompt()

```
void MainWindow::enableGeneralFolderPrompt (
             bool visible,
             FileType type )  [private]
```

Show or hide new folder prompt.

Parameters

visible	If the prompt should be visible or not.
type	For which type we are showing/hiding the prompt.

14.21.3.39 enableGeneralFolderRenamePrompt()

```
void MainWindow::enableGeneralFolderRenamePrompt (
             FileType type )  [private]
```

Show rename file prompt.

Parameters

type	Which type is to be shown.

14.21.3.40 enableMediaFolderPrompt

```
void MainWindow::enableMediaFolderPrompt ( )  [slot]
```

Show new folder edit for media and cancel button.

14.21.3.41 enableNoteFolderPrompt

```
void MainWindow::enableNoteFolderPrompt ( )  [slot]
```

Show new folder edit for notes and cancel button.

14.21.3.42 exportCloseFamilyDiagram()

```
bool MainWindow::exportCloseFamilyDiagram (
             double scale,
             const std::string & filename = EMPTY_STRING )
```

Export close family diagram.

Parameters

scale	What is the scale for this picture.
filename	Where the picture should be saved, if it is empty prompt a dialog to choose file.

Returns

If the export was successful or not.

14.21.3.43 exportCloseFamilyDiagramDialog

```
void MainWindow::exportCloseFamilyDiagramDialog ( )  [slot]
```

Open dialog for exporting close family diagram.

14.21.3.44 exportGeneralDiagram()

```
bool MainWindow::exportGeneralDiagram (
             const QString & filename,
             QGraphicsScene * gscene,
             double scale = 1 )  [private]
```

Export general picture of a diagram.

Parameters

filename	What is the output file path.
gscene	Which scene to use.
scale	That is set for this picture.

Returns

If the export was successful or not.

14.21.3.45 exportHtml

```
void MainWindow::exportHtml ( )  [slot]
```

Export whole HTML project to an output directory by opening a dialog.

14.21.3.46 exportMainHtml

```
void MainWindow::exportMainHtml ( )  [slot]
```

Export only main person html by opening a dialog.

14.21.3.47 exportProject

```
void MainWindow::exportProject ( )  [slot]
```

Open dialog for exporting project.

14.21.3.48 exportTemplates

```
void MainWindow::exportTemplates ( )  [slot]
```

Open dialog for exporting templates.

14.21.3.49 exportTreeDiagram()

```
bool MainWindow::exportTreeDiagram (
             double scale,
             const std::string & filename = EMPTY_STRING )
```

Export tree diagram.

Parameters

scale	What is the scale for this picture.
filename	Where the picture should be saved, if it is empty prompt a dialog to choose file.

Returns

If the export was successful or not.

14.21.3.50 exportTreeDiagramDialog

```
void MainWindow::exportTreeDiagramDialog ( )  [slot]
```

Open dialog for exporting family tree diagram.

14.21.3.51 filterProjectItems

```
void MainWindow::filterProjectItems ( )  [slot]
```

Filter only matching items in project.

14.21.3.52 findPerson

```
void MainWindow::findPerson ( )  [slot]
```

Focus find in project view.

14.21.3.53 fitTreeInView

```
void MainWindow::fitTreeInView ( )  [slot]
```

Fit the tree view inside.

14.21.3.54 generalGenerationNumber()

```
std::pair< size_t, size_t > MainWindow::generalGenerationNumber (
            Person * p,
            size_t soFar,
            size_t barrier,
            const std::vector< size_t > & container,
            bool up,
            std::function< std::pair< size_t, size_t >(Person *, size_t, size_t)> func )
[private]
```

Compute recursively the maximum number of younger or older generations for the given person.

Parameters

p	Given person.
soFar	What is the depth so far.
barrier	How many more generations to show.
container	Vector of relations in given direction.
up	If the direction is up or not.
func	Which function to recursively call.

Returns

First is what is the width and second is the depth of the tree.

14.21.3.55 importProject

```
void MainWindow::importProject ( ) [slot]
```

Choose file to import from other project.

14.21.3.56 importRootDir

```
void MainWindow::importRootDir ( ) [slot]
```

Choose root directory of other project to import from.

14.21.3.57 importTemplates

```
void MainWindow::importTemplates ( ) [slot]
```

Choose file to import templates.

14.21.3.58 initializeProjectSubView()

```
void MainWindow::initializeProjectSubView (
            ProjectItem * parent,
            Person * person ) [private]
```

Initialize all sub-folder for item in project.

Parameters

parent	Parent item holding these sub items.
person	To which person it is bounded to.

14.21.3.59 initializeProjectView()

```
void MainWindow::initializeProjectView ( )  [private]
```

Initialize project view when loading new family tree.

14.21.3.60 interractWithProjectItem

```
void MainWindow::interractWithProjectItem (
            QTreeWidgetItem * item )  [slot]
```

Show the project item upon clicking it. It depends on the type of the item. For files open the file (same for notes and media) for person select it as a main. For events and relations open dialog with editing them.

Parameters

item	Which item was clicked on.

14.21.3.61 loadSettings()

```
void MainWindow::loadSettings ( )  [private]
```

Load the setting from configuration.

14.21.3.62 olderGenerationNumber()

```
std::pair< size_t, size_t > MainWindow::olderGenerationNumber (
            Person * p,
            int soFar,
            size_t barrier )  [private]
```

Get the number of upper generations for a person.

Parameters

p	Which person we are looking at.
soFar	What is the number so far to the person p.
barrier	How more generations to show.

Returns

First is what is the maximum to the width and second is how high the tree is.

14.21.3.63 openBirthDateDialog

```
void MainWindow::openBirthDateDialog ( )  [slot]
```

Open dialog for custom Birth Date.

14.21.3.64 openCustomDateDialog()

```
void MainWindow::openCustomDateDialog (
             WrappedDate * date )  [private]
```

Open custom date dialog for a date.

Parameters

date	Pointer to the date, that is being edited.

14.21.3.65 openDeathDateDialog

```
void MainWindow::openDeathDateDialog ( )  [slot]
```

Open dialog for custom Death Date.

14.21.3.66 openEventTemplatesDialog

```
void MainWindow::openEventTemplatesDialog ( )  [slot]
```

Open event templates dialog.

14.21.3.67 openExistingProject

```
void MainWindow::openExistingProject ( )  [slot]
```

Open existing root directory of a project.

14.21.3.68 openExternalFile()

```
void MainWindow::openExternalFile (
             const std::string & fileURL )  [static]
```

Open general file in an external app using QDesktopService if possible otherwise opening with a command.

Parameters

fileURL	URL to the file.

14.21.3.69 openGeneralFile()

```
void MainWindow::openGeneralFile (
            QTreeWidgetItem * item,
            FileType type )  [private]
```

Open file after double clicking it.

Parameters

item	Which item was clicked.
type	What is the type of the file.

14.21.3.70 openGeneralProject()

```
void MainWindow::openGeneralProject (
            const std::string & dirPath )  [private]
```

Open general project by opening existing one from the list or new one.

Parameters

dirPath	Path to root directory of the project.

14.21.3.71 openGraphicSettings

```
void MainWindow::openGraphicSettings ( )  [slot]
```

Open dialog for graphic settings.

14.21.3.72 openNewFamilyTreeProject

```
void MainWindow::openNewFamilyTreeProject ( )  [slot]
```

Open new empty project.

14.21.3.73 openRelationTemplatesDialog

```
void MainWindow::openRelationTemplatesDialog ( )  [slot]
```

Open relation templates dialog.

14.21.3.74 openSelectedFile

```
void MainWindow::openSelectedFile (
             QTreeWidgetItem * item )  [slot]
```

Open file after double clicking it.

Parameters

item	Which item was clicked on.

14.21.3.75 openSelectedMedia

```
void MainWindow::openSelectedMedia (
             QTreeWidgetItem * item )  [slot]
```

Open media after double clicking it.

Parameters

item	Which item was clicked on.

14.21.3.76 openSelectedNote

```
void MainWindow::openSelectedNote (
             QTreeWidgetItem * item )  [slot]
```

Open note after double clicking it.

Parameters

item	Which item was clicked on.

14.21.3.77 promoteRelation

```
void MainWindow::promoteRelation ( )  [slot]
```

Promote current relation.

14.21.3.78 refreshDates

```
void MainWindow::refreshDates ( )  [slot]
```

Refresh all dates in persons info tab.

14.21.3.79 refreshEventTab

```
void MainWindow::refreshEventTab ( )  [slot]
```

Refresh list with events.

14.21.3.80 refreshGraphics

```
void MainWindow::refreshGraphics ( )  [slot]
```

Refresh all graphics.

14.21.3.81 refreshPersonsInfoTab()

```
void MainWindow::refreshPersonsInfoTab ( )  [private]
```

Refresh basic information about main person.

14.21.3.82 refreshProjectEventsView()

```
void MainWindow::refreshProjectEventsView (
            ProjectItem * item,
            Person * person )  [private]
```

Refresh project view for events.

Parameters

item	Group item for all events.
person	For which person we are showing events.

14.21.3.83 refreshProjectFilesView()

```
void MainWindow::refreshProjectFilesView (
            ProjectItem * item,
            Person * person )  [private]
```

Refresh project view for files.

Parameters

item	Group item for all files.
person	For which person we are showing files.

14.21.3.84 refreshProjectGeneralFilesView()

```
void MainWindow::refreshProjectGeneralFilesView (
             const QIcon & containerIcon,
             const QString & containerName,
             const QIcon & fileIcon,
             FileType type,
             ProjectItem * item,
             Person * person ) [private]
```

Refresh project view with general files.

Parameters

containerIcon	Icon of the container having all files.
containerName	Name of the container having all files.
fileIcon	Icon for the file itself.
type	Type of the file.
item	Which item is the container.
person	To which person these containers are bounded.

14.21.3.85 refreshProjectMediaView()

```
void MainWindow::refreshProjectMediaView (
             ProjectItem * item,
             Person * person ) [private]
```

Refresh project view for media.

Parameters

item	Group item for all media.
person	For which person we are showing media.

14.21.3.86 refreshProjectNoteView()

```
void MainWindow::refreshProjectNoteView (
             ProjectItem * item,
             Person * person ) [private]
```

Refresh project view for notes.

Parameters

item	Group item for all notes.
person	For which person we are showing notes.

14.21.3.87 refreshProjectRelationsView()

```
void MainWindow::refreshProjectRelationsView (
            ProjectItem * item,
            Person * person )  [private]
```

Refresh project view for relations.

Parameters

item	Group item for all relations.
person	For which person we are showing relations.

14.21.3.88 refreshProjectView

```
void MainWindow::refreshProjectView ( )  [slot]
```

Refresh the whole project that is shown.

14.21.3.89 refreshRelationTab

```
void MainWindow::refreshRelationTab ( )  [slot]
```

Refresh relations lists.

14.21.3.90 refreshTabsWithFiles

```
void MainWindow::refreshTabsWithFiles ( )  [slot]
```

Refresh all files tabs.

14.21.3.91 refreshTagsList

```
void MainWindow::refreshTagsList ( )  [slot]
```

Refresh shown tags.

14.21.3.92 refreshUi()

```
void MainWindow::refreshUi ( )  [private]
```

Refresh the whole UI (tabs, project and graphics).

14.21.3.93 removeEvent

```
void MainWindow::removeEvent ( )  [slot]
```

Remove event from family tree.

14.21.3.94 removeFileFolder

```
void MainWindow::removeFileFolder ( )  [slot]
```

Remove file folder.

14.21.3.95 removeGeneralFolder()

```
void MainWindow::removeGeneralFolder (
            FileType type )  [private]
```

Remove folder from the tree widget.

Parameters

type	Which type is used. To determine which tree widget is used.

14.21.3.96 removeMediaFolder

```
void MainWindow::removeMediaFolder ( )  [slot]
```

Remove media folder.

14.21.3.97 removeNoteFolder

```
void MainWindow::removeNoteFolder ( )  [slot]
```

Remove note folder.

14.21.3.98 removePerson

```
void MainWindow::removePerson ( )  [slot]
```

Remove main person.

14.21.3.99 removeRelation

```
void MainWindow::removeRelation ( )  [slot]
```

Remove current relation.

14.21.3.100 removeTag

```
void MainWindow::removeTag ( )  [slot]
```

Delete current tag selected on the person.

14.21.3.101 renameFile

```
void MainWindow::renameFile ( )  [slot]
```

Rename file or folder with files.

14.21.3.102 renameGeneralFileItem()

```
void MainWindow::renameGeneralFileItem (
             FileType type )  [private]
```

Rename general file tree item.

Parameters

type	Which type is used. To determine which tree widget is used.

14.21.3.103 renameMedia

```
void MainWindow::renameMedia ( )  [slot]
```

Rename media or folder with media.

14.21.3.104 renameNote

```
void MainWindow::renameNote ( )  [slot]
```

Rename note or folder with notes.

14.21.3.105 saveDatabase

```
bool MainWindow::saveDatabase ( )  [slot]
```

Save the database.

Returns

If the database was stored.

14.21.3.106 saveGraphicsSetting

```
void MainWindow::saveGraphicsSetting (
             QColor probandColor,
             QColor promotedColor,
             QColor standardColor,
             QColor highlightedColor,
             int sizeX,
             int sizeY,
             QFont font,
             QColor lineColor,
             QColor textColor,
             int lineWidth,
             int rounding )  [slot]
```

Save graphics settings from its dialog.

Parameters

probandColor	Color used for proband.
promotedColor	Color for promoted Relations.
standardColor	Standard color.
highlightedColor	Highlighted color.
sizeX	Size of the box in x axis.
sizeY	Size of the box in y axis.
font	Used font for the text in the box.
lineColor	Color for lines.
textColor	Color for texts.
lineWidth	Width of the lines.
rounding	Rounding of the boxes.

14.21.3.107 savePersonsInfo()

```
void MainWindow::savePersonsInfo ( )  [private]
```

Save all basic information about main person.

14.21.3.108 selectOlderGenerationOnly

```
void MainWindow::selectOlderGenerationOnly ( )  [slot]
```

De-select all other relation boxes other than older.

14.21.3.109 selectSameGenerationOnly

```
void MainWindow::selectSameGenerationOnly ( )  [slot]
```

De-select all other relation boxes other than same.

14.21.3.110 selectYoungerGenerationOnly

```
void MainWindow::selectYoungerGenerationOnly ( )  [slot]
```

De-select all other relation boxes other than younger.

14.21.3.111 setEnabledDeathAtributes

```
void MainWindow::setEnabledDeathAtributes (
             bool disabled )  [slot]
```

Change the visibility of all widgets related to death attributes.

Parameters

disabled	If the widgets should be disabled or not.

14.21.3.112 setEventTabVisible

```
void MainWindow::setEventTabVisible ( )  [slot]
```

Hide or show the event tab.

14.21.3.113 setFileTabVisible

```
void MainWindow::setFileTabVisible ( )  [slot]
```

Hide or show the file tab.

14.21.3.114 setGeneralTabVisible

```
void MainWindow::setGeneralTabVisible ( )  [slot]
```

Hide or show the general tab.

14.21.3.115 setMainPerson

```
void MainWindow::setMainPerson (
             size_t personId )  [slot]
```

Set the main person by clicking on graphics item.

Parameters

person↩ Id	Person id in the clicked box.

14.21.3.116 setMediaTabVisible

```
void MainWindow::setMediaTabVisible ( )  [slot]
```

Hide or show the media tab.

14.21.3.117 setNoteTabVisible

```
void MainWindow::setNoteTabVisible ( )  [slot]
```

Hide or show the note tab.

14.21.3.118 setNumberOfGenerations()

```
std::pair< int, int > MainWindow::setNumberOfGenerations (
            int up,
            int down )
```

Set the number of generations to show.

Parameters

up	Number of generations above (older).
down	Number of generations below (younger).

Returns

Values that were set before (in the same order, first up then down).

14.21.3.119 setRelationTabVisible

```
void MainWindow::setRelationTabVisible ( ) [slot]
```

Hide or show the relation tab.

14.21.3.120 setTabEast

```
void MainWindow::setTabEast ( ) [slot]
```

Set the setting of the tab to east.

14.21.3.121 setTabNorth

```
void MainWindow::setTabNorth ( ) [slot]
```

Set the setting of the tab to north.

14.21.3.122 setTabSouth

```
void MainWindow::setTabSouth ( ) [slot]
```

Set the setting of the tab to south.

14.21.3.123 setTabWest

```
void MainWindow::setTabWest ( ) [slot]
```

Set the setting of the tab to west.

14.21.3.124 setUiElements()

```
void MainWindow::setUiElements ( )  [private]
```

Set all additional static widgets of the user interface.

14.21.3.125 showGeneralFiles()

```
void MainWindow::showGeneralFiles (
            QTreeWidget * widget,
            VirtualDrive * root,
            FileType type )  [private]
```

Show files from container to its widget.

Parameters

widget	QTreeWidget to show all the files.
root	Root folder of all files.
type	Which type of files are going to be displayed.

14.21.3.126 showHelp

```
void MainWindow::showHelp ( )  [slot]
```

Show generic help.

14.21.3.127 showRenameFile

```
void MainWindow::showRenameFile ( )  [slot]
```

Show prompt for renaming file.

14.21.3.128 showRenameMedia

```
void MainWindow::showRenameMedia ( )  [slot]
```

Show prompt for renaming media.

14.21.3.129 showRenameNote

```
void MainWindow::showRenameNote ( )  [slot]
```

Show prompt for renaming note.

14.21.3.130 splitterChanged

```
void MainWindow::splitterChanged ( )  [slot]
```

When the splitter changed.

14.21.3.131 switchNewTreeProject

```
void MainWindow::switchNewTreeProject ( )  [slot]
```

Save tree and create new empty tree to show.

14.21.3.132 youngerGenerationNumber()

```
std::pair< size_t, size_t > MainWindow::youngerGenerationNumber (
             Person * p,
             size_t soFar,
             size_t barrier )  [private]
```

Compute the maximum number of younger generations of the person.

Parameters

p	Given person.
soFar	What is the depth so far.
barrier	How more generations to show.

Returns

First is what is the maximum to the width and second is how deep the tree is.

14.21.3.133 zoomIn

```
void MainWindow::zoomIn ( )  [slot]
```

Zoom in on the family tree view.

14.21.3.134 zoomOut

```
void MainWindow::zoomOut ( )  [slot]
```

Zoom out of the family tree view.

14.21.4 Member Data Documentation

14.21.4.1 borderRadius_

```
int MainWindow::borderRadius_  [private]
```

Radius of the boxes.

14.21.4.2 font_

```
QFont MainWindow::font_ [private]
```

Used font for view.

14.21.4.3 FT

```
FamilyTree MainWindow::FT [private]
```

Family tree.

14.21.4.4 genSizeDown_

```
int MainWindow::genSizeDown_ [private]
```

How many generations to show in tree view down.

14.21.4.5 genSizeUp_

```
int MainWindow::genSizeUp_ [private]
```

How many generations to show in tree view up.

14.21.4.6 highlightedColor_

```
QColor MainWindow::highlightedColor_ [private]
```

Color for highlighted boxes.

14.21.4.7 linePen_

```
QPen MainWindow::linePen_ [private]
```

Pen used for drawing lines.

14.21.4.8 probandColor_

```
QColor MainWindow::probandColor_ [private]
```

Color of proband.

14.21.4.9 promotedColor_

```
QColor MainWindow::promotedColor_ [private]
```

Color for promoted father, mother and partner.

14.21.4.10 scene

```
QGraphicsScene* MainWindow::scene  [private]
```

Scene shown on the graphical view.

14.21.4.11 sizeX_

```
int MainWindow::sizeX_  [private]
```

X size of the person box.

14.21.4.12 sizeY_

```
int MainWindow::sizeY_  [private]
```

Y size of the person box.

14.21.4.13 standardColor_

```
QColor MainWindow::standardColor_  [private]
```

Color for other relations.

14.21.4.14 textPen_

```
QPen MainWindow::textPen_  [private]
```

Pen used for texts.

14.21.4.15 treeScene

```
QGraphicsScene* MainWindow::treeScene  [private]
```

Tree view scene;.

14.21.4.16 treeView

```
GraphicsView* MainWindow::treeView  [private]
```

Tree view graphics view.

14.21.4.17 ui

```
Ui::MainWindow* MainWindow::ui  [private]
```

Qt UI framework.

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

- graphics/mainwindow.h
- graphics/mainwindow.cpp
- graphics/mainwindow_slots.cpp

14.22 Parser Class Reference

Main class for working with the database.

```
#include <file_parser.h>
```

Public Member Functions

- Parser ()

  Default constructor.
- void checkFileConsistency (std::set< std::string > &database, std::vector< std::string > &found, const std::string &dir)
- void clear ()

  Clear stored data.
- bool containsBackupFile (std::string &backupFile)
- bool copyFile (const std::string &filePath, const std::string &dir, std::string &fileName, size_t id)
- void createEmptyFile (const std::string &dir, const std::string &fileName)
- std::string createResourcesDir (const std::string &target)
- std::string getAbsoluteFilePath (const std::string &dir, const std::string &fileName) const
- void getFileUrl (const std::string &dir, const std::string &fileName, std::string &fileUrl) const
- bool isRootDirectorySet ()
- void log (const std::string &error)
- void makeNewDatabase (const std::string &dirPath)
- std::pair< bool, std::vector< std::string > > readConfig ()
- bool readJSONFile (const std::string &fileName, Json::Value &root, bool inDatabase=true)
- void removeBackup ()

  Remove backup files if there is any.
- void removeFile (const std::string &filename, const std::string &dir)
- void renameFile (const std::string &originalName, const std::string &changedName, const std::string &dir)
- void restoreBackup (const std::string &backupFile)
- bool setDatabase (const std::string &dirPath)
- void writeConfig (bool help)
- void writeConfig (const std::vector< std::string > &rootPaths, bool help)
- void writeHtml (const std::string &html, const std::string &output)
- void writeHtml (const std::string &html, const std::string &dirPath, const std::string &filename)
- void writeJSON (const std::string &data, const std::string &file, bool inDatabase=true)

Private Member Functions

- bool containsBackup (std::string &backupFile, const std::string &filename)
- void createAllDirs ()

  Create all (empty) directories in the root directory.
- void createJsonFile (const std::string &fileName)

Private Attributes

- std::filesystem::path root_

  Path to the root directory.
- std::filesystem::path configPath_

  Path to the configuration file.

14.22.1 Detailed Description

Main class for working with the database.

14.22.2 Constructor & Destructor Documentation

14.22.2.1 Parser()

```
Parser::Parser ( )
```

Default constructor.

14.22.3 Member Function Documentation

14.22.3.1 checkFileConsistency()

```
void Parser::checkFileConsistency (
             std::set< std::string > & database,
             std::vector< std::string > & found,
             const std::string & dir )
```

Check the consistency of files.

Parameters

database	Which files are stored in database, these will be removed from the list if found.
found	Files found and not present in database.
dir	Which directory to look into.

14.22.3.2 clear()

```
void Parser::clear ( )
```

Clear stored data.

14.22.3.3 containsBackup()

```
bool Parser::containsBackup (
            std::string & backupFile,
            const std::string & filename )  [private]
```

If there is backup for this file-name.

Parameters

backupFile	If this backup file exists put there the name of it.
filename	Name of the file checked for having backup.

Returns

True if the backup exists.

14.22.3.4 containsBackupFile()

```
bool Parser::containsBackupFile (
            std::string & backupFile )
```

If the database store some backup files, which may be used. Note there can be only one backup file (when the app is working properly).

Parameters

backupFile	If there is backup there will be stored the name of the file.

Returns

True if there is a backup file.

14.22.3.5 copyFile()

```
bool Parser::copyFile (
            const std::string & filePath,
            const std::string & dir,
            std::string & fileName,
            size_t id )
```

Create a new file by copying existing file on the disk.

Parameters

filePath	Path to the original file on the disk.
dir	Which directory to use in database of the root directory.
fileName	Is the name of the file in directory. Use for return value.
id	Id of the new file, to create the real name.

Returns

True if the process was successful, false otherwise.

14.22.3.6 createAllDirs()

```
void Parser::createAllDirs ( )  [private]
```

Create all (empty) directories in the root directory.

14.22.3.7 createEmptyFile()

```
void Parser::createEmptyFile (
             const std::string & dir,
             const std::string & fileName )
```

Create an empty file.

Parameters

dir	Which directory in database to use.
fileName	What is the real file name in its directory.

14.22.3.8 createJsonFile()

```
void Parser::createJsonFile (
             const std::string & fileName )  [private]
```

Create an empty JSON file.

Parameters

fileName	Name of the file in the root directory.

14.22.3.9 createResourcesDir()

```
std::string Parser::createResourcesDir (
             const std::string & target )
```

Create resource file for diagrams for HTML export.

Parameters

target	Where the resource directory should be (either a path to the directory or an html file).

Returns

String representing the path to the resources directory.

14.22.3.10 getAbsoluteFilePath()

```
std::string Parser::getAbsoluteFilePath (
             const std::string & dir,
             const std::string & fileName ) const
```

Get the absolute path to the file in database.

Parameters

dir	Which directory should have the file.
fileName	What is the file-name in its directory.

Returns

String representing the absolute path.

14.22.3.11 getFileUrl()

```
void Parser::getFileUrl (
             const std::string & dir,
             const std::string & fileName,
             std::string & fileUrl ) const
```

Get the url to the file for opening the file. Format: ' `file://`[file path]'.

Parameters

dir	Which directory should have the file.
fileName	What is the file-name in its directory.
fileUrl	String representing the URL in specified format.

14.22.3.12 isRootDirectorySet()

```
bool Parser::isRootDirectorySet ( )
```

Whether the directory has been already set.

Returns

True if the directory was already chosen.

14.22.3.13 log()

```
void Parser::log (
            const std::string & error )
```

Log error to the error file.

Parameters

error	What error occurred.

14.22.3.14 makeNewDatabase()

```
void Parser::makeNewDatabase (
            const std::string & dirPath )
```

Make new empty database. Create all directories and files.

Parameters

dirPath	Path to the chosen root directory.

14.22.3.15 readConfig()

```
std::pair< bool, std::vector< std::string > > Parser::readConfig ( )
```

Read config file if it exists and return vector of all saved paths.

Returns

If help should be shown and vector of strings representing paths to the directories.

14.22.3.16 readJSONFile()

```
bool Parser::readJSONFile (
            const std::string & fileName,
            Json::Value & root,
            bool inDatabase = true )
```

Load the content of JSON file to Json::Value format.

Parameters

fileName	File in the root directory to be used.
root	Where to store parsed data from the file.
inDatabase	If the file is in database or not, if it is not, then it is for import.

Returns

True if the parsing was successful. False otherwise.

14.22.3.17 removeBackup()

```
void Parser::removeBackup ( )
```

Remove backup files if there is any.

14.22.3.18 removeFile()

```
void Parser::removeFile (
             const std::string & filename,
             const std::string & dir )
```

Remove file with given name in given directory.

Parameters

filename	Name of the file.
dir	Given directory with the file.

14.22.3.19 renameFile()

```
void Parser::renameFile (
             const std::string & originalName,
             const std::string & changedName,
             const std::string & dir )
```

Rename a file in its directory.

Parameters

originalName	Original name of the file.
changedName	New changed name of the file.
dir	Directory where the file is.

14.22.3.20 restoreBackup()

```
void Parser::restoreBackup (
             const std::string & backupFile )
```

Restore backup file.

Parameters

backupFile	Which backup file is going to be restored.

14.22.3.21 setDatabase()

```
bool Parser::setDatabase (
            const std::string & dirPath )
```

Set the root directory and check its content.

Parameters

dirPath	Path to the root directory.

Returns

True if all files and directories exist.

14.22.3.22 writeConfig() [1/2]

```
void Parser::writeConfig (
            bool help )
```

Write only current project path to the configuration.

Parameters

help	If help window should be shown at the start-up.

14.22.3.23 writeConfig() [2/2]

```
void Parser::writeConfig (
            const std::vector< std::string > & rootPaths,
            bool help )
```

Write given paths to the root directories to the configuration file.

Parameters

rootPaths	Vector of all paths to the directories.
help	If help window should be shown at the start-up.

14.22.3.24 writeHtml() [1/2]

```
void Parser::writeHtml (
            const std::string & html,
            const std::string & dirPath,
            const std::string & filename )
```

Write HTML text to an output file (or for CSS).

Parameters

html	HTML text to put in an output file.
dirPath	Path to the root directory.
filename	Name of the file in root directory.

14.22.3.25 writeHtml() [2/2]

```
void Parser::writeHtml (
            const std::string & html,
            const std::string & output )
```

Write HTML text to an output file (or for CSS).

Parameters

html	HTML text to put in an output file.
output	Output file path.

14.22.3.26 writeJSON()

```
void Parser::writeJSON (
            const std::string & data,
            const std::string & file,
            bool inDatabase = true )
```

Write data in JSON formatting to its file. Before create a backup and after saving delete the backup.

Parameters

data	Preprocessed JSON format string of data.
file	The name of the JSON file in the root directory.
inDatabase	If the file is in database or not, if it is not it is for export.

14.22.4 Member Data Documentation

14.22.4.1 configPath_

```
std::filesystem::path Parser::configPath_  [private]
```

Path to the configuration file.

14.22.4.2 root_

```
std::filesystem::path Parser::root_ [private]
```

Path to the root directory.

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

- core/file_parser.h
- core/file_parser.cpp

14.23 Person Class Reference

One person in the family tree.

```
#include <person.h>
```

Public Member Functions

- Person ()

  Default constructor.
- void addEvent (size_t id)
- void addRelation (size_t id)
- void addTag (const std::string &tag, const std::string &value)
- WrappedDate ∗ birthDate ()
- WrappedDate ∗ deathDate ()
- std::pair< bool, const std::string & > existsTag (const std::string &tag) const
- void exportJson (std::ostream &os, bool includingEvents, bool includingRelations)
- const std::string & getAfterTitle () const
- const WrappedDate & getBirthDate () const
- const std::string & getBirthPlace () const
- const WrappedDate & getDeathDate () const
- const std::string & getDeathPlace () const
- const std::vector< size_t > & getEvents () const
- size_t getFather () const
- std::string getFullName () const
- const VirtualDrive & getFilesRoot (FileType type=GENERAL_FILE) const
- VirtualDrive ∗ getFilesRootPointer (FileType type)
- const std::string & getFrontTitle () const
- Gender getGender () const
- size_t getId () const
- const std::string & getMaidenName () const
- size_t getMother () const
- const std::string & getName () const
- size_t getPartner () const
- const std::vector< size_t > & getRelations () const
- const std::string & getSurname () const
- const std::vector< std::pair< std::string, std::string > > & getTags () const

- void importJson (const Json::Value &value)
- bool isAlive () const
- void printHtml (std::ostream &os)
- void promoteRelation (size_t id, Trait type)
- void readJson (const Json::Value &value)
- void removeEvent (size_t id)
- void removeRelation (size_t id)
- void removeTag (const std::string &tag)
- void setAfterTitle (const std::string &title)
- void setBirthDate (int y, int m, int d)
- std::string & setBirthPlace ()
- void setDeathDate (int y, int m, int d)
- std::string & setDeathPlace ()
- void setFather (size_t father)
- void setFilesRoot (const VirtualDrive &drive, FileType type=GENERAL_FILE, size_t plusIndex=0)
- void setFrontTitle (const std::string &title)
- void setGender (Gender g)
- void setId (size_t id)
- void setLives (bool lives)
- std::string & setMaidenName ()
- void setMother (size_t mother)
- std::string & setName ()
- void setPartner (size_t id)
- std::string & setSurname ()
- std::string str () const
- void updateSpecialRelation (size_t relId, Trait trait)

Private Attributes

- WrappedDate dateOfBirth_

  Date of birth.
- WrappedDate dateOfDeath_

  Date of death.
- std::vector< size_t > events_

  All events ids (from Family Tree) of the person.
- size_t father_

  Index of relation with fatherhood trait.
- Gender gender_

  Gender of the person.
- size_t id_

  Id of the person.
- bool lives_

  If the person is alive.
- std::string maidenName_

  Maiden name of the person. Usually for a married woman.
- size_t mother_

  Index of relation with motherhood trait.
- std::string name_

  Name of the person.
- size_t partner_

  Id of the partnership relation.
- std::string placeOfBirth_

Place of birth.

- std::string placeOfDeath_

  Place of death.
- std::vector< size_t > relations_

  All relations ids (from Family Tree).
- VirtualDrive rootFileDrive_

  Root drive for files.
- VirtualDrive rootMediaDrive_

  Root drive for media.
- VirtualDrive rootNoteDrive_

  Root drive for notes.
- std::string surname_

  Surname of the person.
- std::vector< std::pair< std::string, std::string > > tags_

  Persons tags. First is the tag itself and second is the value.
- std::string titleAfter_

  Title that is after the name.
- std::string titleInFront_

  Title that is in front of a name.

14.23.1 Detailed Description

One person in the family tree.

14.23.2 Constructor & Destructor Documentation

14.23.2.1 Person()

```
Person::Person ( )
```

Default constructor.

14.23.3 Member Function Documentation

14.23.3.1 addEvent()

```
void Person::addEvent (
             size_t id )
```

Add new event as its id.

Parameters

id	Its id.

14.23.3.2 addRelation()

```
void Person::addRelation (
             size_t id )
```

Add new general relation as its id.

Parameters

id	Its id.

14.23.3.3 addTag()

```
void Person::addTag (
             const std::string & tag,
             const std::string & value )
```

Add tag to the person.

Parameters

tag	Used tag.
value	Value of the tag.

14.23.3.4 birthDate()

```
WrappedDate * Person::birthDate ( )
```

Get the pointer to the birth date.

Returns

Pointer to the birth date.

14.23.3.5 deathDate()

```
WrappedDate * Person::deathDate ( )
```

Get the pointer to the death date.

Returns

Pointer to the death date.

14.23.3.6 existsTag()

```
std::pair< bool, const std::string & > Person::existsTag (
             const std::string & tag ) const
```

If there exists value with this tag.

Parameters

tag	Given tag.

Returns

True if it exists and then the value.

14.23.3.7 exportJson()

```
void Person::exportJson (
             std::ostream & os,
             bool includingEvents,
             bool includingRelations )
```

Export this person.

Parameters

os	Given output stream.
includingEvents	If events should be included.
includingRelations	If relations should be included.

14.23.3.8 getAfterTitle()

```
const std::string & Person::getAfterTitle ( ) const
```

Get the title after the name.

Returns

Constant reference to the title.

14.23.3.9 getBirthDate()

```
const WrappedDate & Person::getBirthDate ( ) const
```

Get the date of birth.

Returns

Date of birth.

14.23.3.10 getBirthPlace()

```
const std::string & Person::getBirthPlace ( ) const
```

Get the string of place of birth.

Returns

Constant reference to the string of the place of birth.

14.23.3.11 getDeathDate()

```
const WrappedDate & Person::getDeathDate ( ) const
```

Get the date of death.

Returns

Date of death.

14.23.3.12 getDeathPlace()

```
const std::string & Person::getDeathPlace ( ) const
```

Get the string representing the place of death.

Returns

String representing the place of death.

14.23.3.13 getEvents()

```
const std::vector< size_t > & Person::getEvents ( ) const
```

Get constant reference to the list of all events.

Returns

List of events ids.

14.23.3.14 getFather()

```
size_t Person::getFather ( ) const
```

Get the index of fatherhood relation representing father.

Returns

Index of the fatherhood relation.

14.23.3.15 getFilesRoot()

```
const VirtualDrive & Person::getFilesRoot (
            FileType type = GENERAL_FILE ) const
```

Get constant reference to the root drive of files.

Parameters

type	What type of files.

Returns

Root drive for given file type.

14.23.3.16 getFilesRootPointer()

```
VirtualDrive * Person::getFilesRootPointer (
             FileType type )
```

Get the pointer to the files root drive.

Parameters

type	Type of the files.

Returns

Pointer to the root drive.

14.23.3.17 getFrontTitle()

```
const std::string & Person::getFrontTitle ( ) const
```

Get the title in front.

Returns

Constant reference to the title.

14.23.3.18 getFullName()

```
std::string Person::getFullName ( ) const
```

Get the full name of a person in order: ¨Surname + MaidenName + Name¨.

Returns

Full name of a person.

14.23.3.19 getGender()

```
Gender Person::getGender ( ) const
```

Get the gender of the person.

Returns

Person's gender.

14.23.3.20 getId()

```
size_t Person::getId ( ) const
```

Get the id of the person.

Returns

The id of the person.

14.23.3.21 getMaidenName()

```
const std::string & Person::getMaidenName ( ) const
```

Get the maiden name.

Returns

Constant reference to the maiden name.

14.23.3.22 getMother()

```
size_t Person::getMother ( ) const
```

Get the index of motherhood relation representing mother.

Returns

Index of the motherhood relation.

14.23.3.23 getName()

```
const std::string & Person::getName ( ) const
```

Get the name of the person.

Returns

Constant reference to the name.

14.23.3.24 getPartner()

```
size_t Person::getPartner ( ) const
```

Get the id of the partner.

Returns

Id of the partnership relation.

14.23.3.25 getRelations()

```
const std::vector< size_t > & Person::getRelations ( ) const
```

Get constant reference to the list of all relations.

Returns

List of relations ids.

14.23.3.26 getSurname()

```
const std::string & Person::getSurname ( ) const
```

Get the surname of the person.

Returns

Constant reference to the surname.

14.23.3.27 getTags()

```
const std::vector< std::pair< std::string, std::string > > & Person::getTags ( ) const
```

Get all tags.

Returns

Constant reference to the tags container.

14.23.3.28 importJson()

```
void Person::importJson (
             const Json::Value & value )
```

Import person from single file JSON. Some values may not be present from the import.

Parameters

value	JSON value.

14.23.3.29 isAlive()

```
bool Person::isAlive ( ) const
```

If the person is alive or not.

Returns

True if the person lives.

14.23.3.30 printHtml()

```
void Person::printHtml (
             std::ostream & os )
```

Print person details to an output in HTML syntax.

Parameters

os	Output stream.

14.23.3.31 promoteRelation()

```
void Person::promoteRelation (
             size_t id,
             Trait type )
```

Promote relation to the main one. If it is already promoted de-promote it.

Parameters

id	Of the relation.
type	Type of the relation.

14.23.3.32 readJson()

```
void Person::readJson (
             const Json::Value & value )
```

Read data from JSON to this person.

Parameters

value	Loaded JSON data from the file.

14.23.3.33 removeEvent()

```
void Person::removeEvent (
             size_t id )
```

Remove event if the person has it (by its id).

Parameters

id	Its id.

14.23.3.34 removeRelation()

```
void Person::removeRelation (
             size_t id )
```

Remove relation if the person has it (by its id) that includes children and other special relations.

Parameters

id	Its id.

14.23.3.35 removeTag()

```
void Person::removeTag (
             const std::string & tag )
```

Remove tag from the person if it exists.

Parameters

tag	Which tag will be removed.

14.23.3.36 setAfterTitle()

```
void Person::setAfterTitle (
             const std::string & title )
```

Set the title after the name.

Parameters

title	The new title.

14.23.3.37 setBirthDate()

```
void Person::setBirthDate (
             int y,
             int m,
             int d )
```

Set the exact birth date.

Parameters

d	Number of day.
m	Number of month.
y	Number of year.

14.23.3.38 setBirthPlace()

```
std::string & Person::setBirthPlace ( )
```

Set the birth place by giving reference to the string.

Returns

Reference to the birth place.

14.23.3.39 setDeathDate()

```
void Person::setDeathDate (
             int y,
             int m,
             int d )
```

Set the exact death date.

Parameters

d	Number of day.
m	Number of month.
y	Number of year.

14.23.3.40 setDeathPlace()

```
std::string & Person::setDeathPlace ( )
```

Set the death place by giving reference to the string.

Returns

Reference to the death place.

14.23.3.41 setFather()

```
void Person::setFather (
             size_t father )
```

Set father for the person. If father is present put him into relations.

Parameters

father	Index to fatherhood relation.

14.23.3.42 setFilesRoot()

```
void Person::setFilesRoot (
             const VirtualDrive & drive,
             FileType type = GENERAL_FILE,
             size_t plusIndex = 0 )
```

Set the drive by copying the data from the other.

Parameters

drive	Other drive.
type	What type of file is the drive.
plusIndex	What should be added to each file index.

14.23.3.43 setFrontTitle()

```
void Person::setFrontTitle (
             const std::string & title )
```

Set the title in front of a name.

Parameters

title	The new title.

14.23.3.44 setGender()

```
void Person::setGender (
             Gender g )
```

Set the gender.

Parameters

g	Given gender.

14.23.3.45 setId()

```
void Person::setId (
             size_t id )
```

Set the id of the person.

Parameters

id	The new id.

14.23.3.46 setLives()

```
void Person::setLives (
             bool lives )
```

Set if the person lives or not.

Parameters

lives	True if the person is still alive.

14.23.3.47 setMaidenName()

```
std::string & Person::setMaidenName ( )
```

Set the maiden name by getting the reference to it.

Returns

The reference to the maiden name.

14.23.3.48 setMother()

```
void Person::setMother (
             size_t mother )
```

Set mother for the person. If mother is present put her into relations.

Parameters

mother	Index to motherhood relation.

14.23.3.49 setName()

```
std::string & Person::setName ( )
```

Get the reference to the name for changing the name.

Returns

The reference to the name string.

14.23.3.50 setPartner()

```
void Person::setPartner (
             size_t id )
```

Set the partnership relation id.

Parameters

id	Id of the partnership relation.

14.23.3.51 setSurname()

```
std::string & Person::setSurname ( )
```

Get the reference to the surname for changing it.

Returns

The reference to the surname string.

14.23.3.52 str()

```
std::string Person::str ( ) const
```

Get the string representation of person.

Returns

String representing a person.

14.23.3.53 updateSpecialRelation()

```
void Person::updateSpecialRelation (
             size_t relId,
             Trait trait )
```

Update relation if it is special.

Parameters

rel←↩ Id	Which id it has.
trait	Which trait it now has.

14.23.4 Member Data Documentation

14.23.4.1 dateOfBirth_

```
WrappedDate Person::dateOfBirth_  [private]
```

Date of birth.

14.23.4.2 dateOfDeath_

```
WrappedDate Person::dateOfDeath_  [private]
```

Date of death.

14.23.4.3 events_

```
std::vector<size_t> Person::events_  [private]
```

All events ids (from Family Tree) of the person.

14.23.4.4 father_

```
size_t Person::father_  [private]
```

Index of relation with fatherhood trait.

14.23.4.5 gender_

```
Gender Person::gender_  [private]
```

Gender of the person.

14.23.4.6 id_

```
size_t Person::id_  [private]
```

Id of the person.

14.23.4.7 lives_

```
bool Person::lives_ [private]
```

If the person is alive.

14.23.4.8 maidenName_

```
std::string Person::maidenName_ [private]
```

Maiden name of the person. Usually for a married woman.

14.23.4.9 mother_

```
size_t Person::mother_ [private]
```

Index of relation with motherhood trait.

14.23.4.10 name_

```
std::string Person::name_ [private]
```

Name of the person.

14.23.4.11 partner_

```
size_t Person::partner_ [private]
```

Id of the partnership relation.

14.23.4.12 placeOfBirth_

```
std::string Person::placeOfBirth_ [private]
```

Place of birth.

14.23.4.13 placeOfDeath_

```
std::string Person::placeOfDeath_ [private]
```

Place of death.

14.23.4.14 relations_

```
std::vector<size_t> Person::relations_ [private]
```

All relations ids (from Family Tree).

14.23.4.15 rootFileDrive_

```
VirtualDrive Person::rootFileDrive_ [private]
```

Root drive for files.

14.23.4.16 rootMediaDrive_

```
VirtualDrive Person::rootMediaDrive_ [private]
```

Root drive for media.

14.23.4.17 rootNoteDrive_

```
VirtualDrive Person::rootNoteDrive_ [private]
```

Root drive for notes.

14.23.4.18 surname_

```
std::string Person::surname_ [private]
```

Surname of the person.

14.23.4.19 tags_

```
std::vector<std::pair<std::string, std::string> > Person::tags_ [private]
```

Persons tags. First is the tag itself and second is the value.

14.23.4.20 titleAfter_

```
std::string Person::titleAfter_ [private]
```

Title that is after the name.

14.23.4.21 titleInFront_

```
std::string Person::titleInFront_ [private]
```

Title that is in front of a name.

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

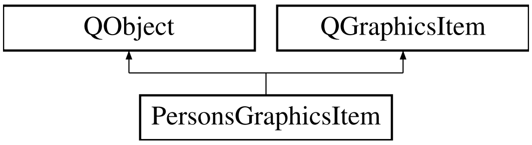
- core/person.h
- core/person.cpp

14.24 PersonsGraphicsItem Class Reference

Box with persons information.

```
#include <graphics_items.h>
```

Inheritance diagram for PersonsGraphicsItem:



Signals

- void itemClicked (size_t personId)

    Signal when the item is pressed.

Public Member Functions

- PersonsGraphicsItem (const QStringList &lines, int sizex, int sizey, Person ∗p, qreal x, qreal y, QColor color, QColor highlighted, QFont font, QWidget ∗parent, QPen ∗pen, QPen ∗textPen, int borderRadius)
- ∼PersonsGraphicsItem ()

    Default destructor.
- QRectF boundingRect () const override
- void paint (QPainter ∗painter, const QStyleOptionGraphicsItem ∗option, QWidget ∗widget) override
- QPainterPath shape () const override

Protected Member Functions

- void hoverEnterEvent (QGraphicsSceneHoverEvent ∗event) override
- void hoverLeaveEvent (QGraphicsSceneHoverEvent ∗event) override
- void mousePressEvent (QGraphicsSceneMouseEvent ∗event) override

Private Attributes

- int borderRadius_

    Border radius of the box.
- int bold_text_

    Which text should be bold.
- QColor color_

    Which color will be used to fill the box.
- QFont font_

    Which font is used.
- QColor highlighted_

    Which color will be used when highlighted.
- bool hovered_

    If the box is hovered.

- QPen ∗ pen_

  Pointer to the used pen.
- Person ∗ person_

  Given person represented in this box.
- int sizex_

  Size of the cell in the x axis.
- int sizey_

  Size of the cell in the y axis.
- QStringList textLines_

  All text lines to show in the text box.
- QPen ∗ textPen_

  Pointer to the text pen.
- qreal x_

  Top left point x position.
- qreal y_

  Top left point y position.

14.24.1 Detailed Description

Box with persons information.

14.24.2 Constructor & Destructor Documentation

14.24.2.1 PersonsGraphicsItem()

```
PersonsGraphicsItem::PersonsGraphicsItem (
             const QStringList & lines,
             int sizex,
             int sizey,
             Person * p,
             qreal x,
             qreal y,
             QColor color,
             QColor highlighted,
             QFont font,
             QWidget * parent,
             QPen * pen,
             QPen * textPen,
             int borderRadius )
```

Default constructor.

Parameters

lines	Each separate line of text to show in the box.
sizex	Size of the cell in the x axis.
sizey	Size of the cell in the y axis.
p	Given person represented in this box.
x	Top left point x position.
y	Top left point y position.
color	Which color will be used to fill the box.

Parameters

highlighted	Which color will be used when the box is highlighted.
font	Which font will be used to type the words.
parent	Which widget will take care of clicking the box.
pen	Which pen is used for lines.
textPen	Which pen is used for text.
borderRadius	What is the radius of the border.

14.24.2.2 ∼PersonsGraphicsItem()

```
PersonsGraphicsItem::∼PersonsGraphicsItem ( )
```

Default destructor.

14.24.3 Member Function Documentation

14.24.3.1 boundingRect()

```
QRectF PersonsGraphicsItem::boundingRect ( ) const  [override]
```

Get the bounding rectangle of the object.

Returns

Rectangle that is bounding the item.

14.24.3.2 hoverEnterEvent()

```
void PersonsGraphicsItem::hoverEnterEvent (
            QGraphicsSceneHoverEvent ∗ event )  [override], [protected]
```

When the box is hovered by a mouse highlight the box.

Parameters

event	Event when the item was hovered.

14.24.3.3 hoverLeaveEvent()

```
void PersonsGraphicsItem::hoverLeaveEvent (
            QGraphicsSceneHoverEvent ∗ event )  [override], [protected]
```

When the box is not hovered anymore by a mouse stop highlighting.

Parameters

event	Event when the item is not hovered anymore.

14.24.3.4 itemClicked

```
void PersonsGraphicsItem::itemClicked (
            size_t personId )  [signal]
```

Signal when the item is pressed.

14.24.3.5 mousePressEvent()

```
void PersonsGraphicsItem::mousePressEvent (
            QGraphicsSceneMouseEvent * event )  [override], [protected]
```

Press event after clicking the object.

Parameters

event	Event when the item was clicked.

14.24.3.6 paint()

```
void PersonsGraphicsItem::paint (
            QPainter * painter,
            const QStyleOptionGraphicsItem * option,
            QWidget * widget )  [override]
```

Paint the object to its painter.

Parameters

painter	Which painter to use to paint the object.
option	Option for the Qt framework (not used).
widget	Widget for the Qt framework (not used).

14.24.3.7 shape()

```
QPainterPath PersonsGraphicsItem::shape ( ) const  [override]
```

Get the shape of the item.

Returns

Get the shape to paint.

14.24.4 Member Data Documentation

14.24.4.1 bold_text_

```
int PersonsGraphicsItem::bold_text_  [private]
```

Which text should be bold.

14.24.4.2 borderRadius_

```
int PersonsGraphicsItem::borderRadius_  [private]
```

Border radius of the box.

14.24.4.3 color_

```
QColor PersonsGraphicsItem::color_  [private]
```

Which color will be used to fill the box.

14.24.4.4 font_

```
QFont PersonsGraphicsItem::font_  [private]
```

Which font is used.

14.24.4.5 highlighted_

```
QColor PersonsGraphicsItem::highlighted_  [private]
```

Which color will be used when highlighted.

14.24.4.6 hovered_

```
bool PersonsGraphicsItem::hovered_  [private]
```

If the box is hovered.

14.24.4.7 pen_

```
QPen* PersonsGraphicsItem::pen_  [private]
```

Pointer to the used pen.

14.24.4.8 person_

```
Person* PersonsGraphicsItem::person_ [private]
```

Given person represented in this box.

14.24.4.9 sizex_

```
int PersonsGraphicsItem::sizex_ [private]
```

Size of the cell in the x axis.

14.24.4.10 sizey_

```
int PersonsGraphicsItem::sizey_ [private]
```

Size of the cell in the y axis.

14.24.4.11 textLines_

```
QStringList PersonsGraphicsItem::textLines_ [private]
```

All text lines to show in the text box.

14.24.4.12 textPen_

```
QPen* PersonsGraphicsItem::textPen_ [private]
```

Pointer to the text pen.

14.24.4.13 x_

```
qreal PersonsGraphicsItem::x_ [private]
```

Top left point x position.

14.24.4.14 y_

```
qreal PersonsGraphicsItem::y_ [private]
```

Top left point y position.

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

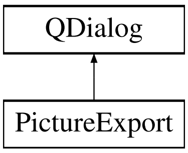
- graphics/graphics_items.h
- graphics/graphics_items.cpp

14.25 PictureExport Class Reference

Dialog for exporting diagram. To choose zoom of the picture.

```
#include <dialogs.h>
```

Inheritance diagram for PictureExport:



Public Slots

- void exportDiagram ()

  When OK is pressed export the diagram.

Public Member Functions

- PictureExport (bool closeDiagram, QWidget ∗parent=nullptr)
- ∼PictureExport ()

  Default destructor.

Private Attributes

- bool closeDiagram_

  Which diagram is being exported.
- Ui::PictureExport ∗ ui

  Used Qt user interface.

14.25.1 Detailed Description

Dialog for exporting diagram. To choose zoom of the picture.

14.25.2 Constructor & Destructor Documentation

14.25.2.1 PictureExport()

```
PictureExport::PictureExport (
            bool closeDiagram,
            QWidget * parent = nullptr )  [explicit]
```

Default constructor.

Parameters

closeDiagram	If the diagram should be close family diagram. Otherwise it is family tree.
parent	The Qt Widget parent.

14.25.2.2 ∼PictureExport()

```
PictureExport::∼PictureExport ( )
```

Default destructor.

14.25.3 Member Function Documentation

14.25.3.1 exportDiagram

```
void PictureExport::exportDiagram ( )  [slot]
```

When OK is pressed export the diagram.

14.25.4 Member Data Documentation

14.25.4.1 closeDiagram_

```
bool PictureExport::closeDiagram_  [private]
```

Which diagram is being exported.

14.25.4.2 ui

```
Ui::PictureExport* PictureExport::ui  [private]
```

Used Qt user interface.

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

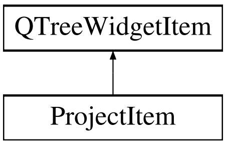
- graphics/dialogs.h
- graphics/dialogs.cpp

14.26 ProjectItem Class Reference

This is a tree widget item to be shown on a project view.

```
#include <graphics_items.h>
```

Inheritance diagram for ProjectItem:



Public Member Functions

- ProjectItem (ProjectItemType type, size_t id=0)
- size_t getId () const
- ProjectItemType type () const

Private Attributes

- size_t id_

  Id of the stored object.
- ProjectItemType type_

  Type of the stored object.

14.26.1 Detailed Description

This is a tree widget item to be shown on a project view.

14.26.2 Constructor & Destructor Documentation

14.26.2.1 ProjectItem()

```
ProjectItem::ProjectItem (
            ProjectItemType type,
            size_t id = 0 )  [explicit]
```

Default constructor.

Parameters

type	What is the type of this item.
id	Given id of the item.

14.26.3 Member Function Documentation

14.26.3.1 getId()

```
size_t ProjectItem::getId ( ) const
```

Get the id of the object.

Returns

Id of the object.

14.26.3.2 type()

```
ProjectItemType ProjectItem::type ( ) const
```

Get the type of the item.

Returns

Type of the project.

14.26.4 Member Data Documentation

14.26.4.1 id_

```
size_t ProjectItem::id_ [private]
```

Id of the stored object.

14.26.4.2 type_

```
ProjectItemType ProjectItem::type_ [private]
```

Type of the stored object.

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

- graphics/graphics_items.h
- graphics/graphics_items.cpp

14.27 Relation Class Reference

Class holding data for a relationship.

```
#include <family_tree_items.h>
```

Public Member Functions

- Relation (Settings *settings)
- size_t getFirstPerson () const
- size_t getId () const
- size_t getSecondPerson () const
- size_t getTemplate () const
- const std::string & getTitle () const
- size_t getTheOtherPerson (size_t current)
- const std::string & getTheOtherPersonName (size_t current)
- Trait getTrait ()
- bool isOlderGeneration (size_t id)
- bool isParent (size_t personId)
- bool isSameGenration ()
- bool isYoungerGeneration (size_t id)
- void printHtml (std::ostream &os, const std::string &firstPerson, const std::string &secondPerson, const std::string &promoted)
- void readJson (const Json::Value &value)
- void setId (size_t id)
- void setPersons (size_t person1, size_t person2)
- void setTemplate (size_t templ)
- std::string str (const std::string &firstPerson, const std::string &secondPerson) const

Private Attributes

- size_t id_

  Id of this relation.
- size_t person1_

  First person of the relation.
- size_t person2_

  Second person of the relation.
- Settings ∗ settings_

  Used settings holding the templates.
- size_t template_

  Relation template it is using.

14.27.1 Detailed Description

Class holding data for a relationship.

14.27.2 Constructor & Destructor Documentation

14.27.2.1 Relation()

```
Relation::Relation (
            Settings ∗ settings )  [explicit]
```

Default constructor.

Parameters

settings	Pointer to the settings of the app.

14.27.3 Member Function Documentation

14.27.3.1 getFirstPerson()

```
size_t Relation::getFirstPerson ( ) const
```

Get the first person of the relation.

Returns

Id of the first person.

14.27.3.2 getId()

```
size_t Relation::getId ( ) const
```

Get the id of the relation.

Returns

Id of the relation.

14.27.3.3 getSecondPerson()

```
size_t Relation::getSecondPerson ( ) const
```

Get the second person of the relation.

Returns

Id of the second person.

14.27.3.4 getTemplate()

```
size_t Relation::getTemplate ( ) const
```

Get the template of the relation.

Returns

Id of the template.

14.27.3.5 getTheOtherPerson()

```
size_t Relation::getTheOtherPerson (
            size_t current )
```

Get the other person from relation.

Parameters

current	Current person.

Returns

Other person in the relation from current.

14.27.3.6 getTheOtherPersonName()

```
const std::string & Relation::getTheOtherPersonName (
            size_t current )
```

Get the name of the other person with respect to the given person as a string.

Parameters

current	Given person id.

Returns

The name of the other person.

14.27.3.7 getTitle()

```
const std::string & Relation::getTitle ( ) const
```

Get the title of the relation.

Returns

Constant reference to the title.

14.27.3.8 getTrait()

```
Trait Relation::getTrait ( )
```

Get the trait of the relation.

Returns

Trait of this relation.

14.27.3.9 isOlderGeneration()

```
bool Relation::isOlderGeneration (
            size_t id )
```

If the given person from the relation is from older generation.

Parameters

id	Id of the given person.

Returns

True if the generation difference from given person - other person is <0.

14.27.3.10 isParent()

```
bool Relation::isParent (
            size_t personId )
```

If the relation is of motherhood or fatherhood trait and the other person is younger.

Parameters

person↩ Id	Id of the person whether it is father or mother.

Returns

True if all given properties hold, otherwise false.

14.27.3.11 isSameGenration()

```
bool Relation::isSameGenration ( )
```

If the generation difference is 0.

Returns

True if the generation difference is 0.

14.27.3.12 isYoungerGeneration()

```
bool Relation::isYoungerGeneration (
             size_t id )
```

If the given person from the relation is from younger generation.

Parameters

id	Id of the given person.

Returns

True if the generation difference from given person - other person is $>$ 0.

14.27.3.13 printHtml()

```
void Relation::printHtml (
             std::ostream & os,
             const std::string & firstPerson,
             const std::string & secondPerson,
             const std::string & promoted )
```

Print relation to an output stream in HTML format.

Parameters

os	Given output stream,
firstPerson	Name of the first person.
secondPerson	Name of the second person.
promoted	What is the name of the promoted version.

14.27.3.14 readJson()

```
void Relation::readJson (
             const Json::Value & value )
```

Read the relation from JSON value.

Parameters

value	The JSON value from file.

14.27.3.15 setId()

```
void Relation::setId (
             size_t id )
```

Set the id of this relation.

Parameters

id	New id.

14.27.3.16 setPersons()

```
void Relation::setPersons (
             size_t person1,
             size_t person2 )
```

Set persons for this relation.

Parameters

person1	Id of the first person.
person2	Id of the second person.

14.27.3.17 setTemplate()

```
void Relation::setTemplate (
             size_t templ )
```

Set the relation template.

Parameters

templ	Id of the template.

14.27.3.18 str()

```
std::string Relation::str (
            const std::string & firstPerson,
            const std::string & secondPerson ) const
```

Get the string representation of relation.

Parameters

firstPerson	String representing the first person.
secondPerson	String representing the second person.

Returns

String representing the relation.

14.27.4 Member Data Documentation

14.27.4.1 id_

```
size_t Relation::id_ [private]
```

Id of this relation.

14.27.4.2 person1_

```
size_t Relation::person1_ [private]
```

First person of the relation.

14.27.4.3 person2_

```
size_t Relation::person2_ [private]
```

Second person of the relation.

14.27.4.4 settings_

```
Settings* Relation::settings_ [private]
```

Used settings holding the templates.

14.27.4.5 template_

```
size_t Relation::template_ [private]
```

Relation template it is using.

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

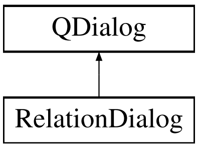
- core/family_tree_items.h
- core/family_tree_items.cpp

14.28 RelationDialog Class Reference

Dialog showing single relation for editing it.

```
#include <dialogs.h>
```

Inheritance diagram for RelationDialog:



Public Slots

- void chooseFirstPerson ()

  Slot for opening dialog to save first person.
- void chooseSecondPerson ()

  Slot for opening dialog to save second person.
- void savePerson (Person ∗person)

  Save person upon choosing it from another dialog.
- void saveRelation ()

  Save relation to the family tree or add new one and save it.
- void showTemplateDetails ()

  Show details about current template.
- void switchPersons ()

  Switch the persons.

Public Member Functions

- RelationDialog (FamilyTree ∗FT, QWidget ∗parent=nullptr, Relation ∗rel=nullptr)
- ∼RelationDialog ()

  Default destructor.

Private Attributes

- FamilyTree ∗ FT_

  Pointer to the family tree.
- Person ∗ person1_

  First person of the relation.
- Person ∗ person2_

  Second person of the relation.
- Relation ∗ rel_

  Pointer to the edited relation, if adding new one it is a null pointer.
- bool saveFirstPerson_

  Whether the first person is to be saved or the second.
- Ui::RelationDialog ∗ ui

  User interface of the Qt framework.

14.28.1 Detailed Description

Dialog showing single relation for editing it.

14.28.2 Constructor & Destructor Documentation

14.28.2.1 RelationDialog()

```
RelationDialog::RelationDialog (
            FamilyTree ∗ FT,
            QWidget ∗ parent = nullptr,
            Relation ∗ rel = nullptr )  [explicit]
```

Default constructor.

Parameters

FT	Pointer to the family tree.
parent	The Qt Widget parent.
rel	Which relation is to be added or edited.

14.28.2.2 ∼RelationDialog()

```
RelationDialog::∼RelationDialog ( )
```

Default destructor.

14.28.3 Member Function Documentation

14.28.3.1 chooseFirstPerson

```
void RelationDialog::chooseFirstPerson ( )  [slot]
```

Slot for opening dialog to save first person.

14.28.3.2 chooseSecondPerson

```
void RelationDialog::chooseSecondPerson ( )  [slot]
```

Slot for opening dialog to save second person.

14.28.3.3 savePerson

```
void RelationDialog::savePerson (
             Person * person )  [slot]
```

Save person upon choosing it from another dialog.

14.28.3.4 saveRelation

```
void RelationDialog::saveRelation ( )  [slot]
```

Save relation to the family tree or add new one and save it.

14.28.3.5 showTemplateDetails

```
void RelationDialog::showTemplateDetails ( )  [slot]
```

Show details about current template.

14.28.3.6 switchPersons

```
void RelationDialog::switchPersons ( )  [slot]
```

Switch the persons.

14.28.4 Member Data Documentation

14.28.4.1 FT_

```
FamilyTree* RelationDialog::FT_  [private]
```

Pointer to the family tree.

14.28.4.2 person1_

```
Person* RelationDialog::person1_  [private]
```

First person of the relation.

14.28.4.3 person2_

```
Person* RelationDialog::person2_ [private]
```

Second person of the relation.

14.28.4.4 rel_

```
Relation* RelationDialog::rel_ [private]
```

Pointer to the edited relation, if adding new one it is a null pointer.

14.28.4.5 saveFirstPerson_

```
bool RelationDialog::saveFirstPerson_ [private]
```

Whether the first person is to be saved or the second.

14.28.4.6 ui

```
Ui::RelationDialog* RelationDialog::ui [private]
```

User interface of the Qt framework.

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

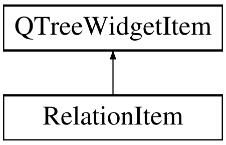
- graphics/dialogs.h
- graphics/dialogs.cpp

14.29 RelationItem Class Reference

Class for showing relation suggestion and holding the data.

```
#include <graphics_items.h>
```

Inheritance diagram for RelationItem:



Public Member Functions

- RelationItem (FamilyTree ∗FT, const RelationSuggestion &suggestion, RelationTemplate ∗templ)
- Relation ∗ getRelation ()
- void refreshTexts ()

    Update the texts.

Private Attributes

- FamilyTree ∗ FT_

  Pointer to the family tree.
- Relation relation_

  Which relation it represents.

14.29.1 Detailed Description

Class for showing relation suggestion and holding the data.

14.29.2 Constructor & Destructor Documentation

14.29.2.1 RelationItem()

```
RelationItem::RelationItem (
            FamilyTree * FT,
            const RelationSuggestion & suggestion,
            RelationTemplate * templ )
```

Default constructor.

Parameters

FT	Pointer to the family tree.
suggestion	Which suggestion this item represents.
templ	Pointer to the used template.

14.29.3 Member Function Documentation

14.29.3.1 getRelation()

```
Relation * RelationItem::getRelation ( )
```

Get the pointer to the relation it represents.

Returns

Pointer to the relation.

14.29.3.2 refreshTexts()

```
void RelationItem::refreshTexts ( )
```

Update the texts.

14.29.4 Member Data Documentation

14.29.4.1 FT_

```
FamilyTree* RelationItem::FT_  [private]
```

Pointer to the family tree.

14.29.4.2 relation_

```
Relation RelationItem::relation_  [private]
```

Which relation it represents.

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

- graphics/graphics_items.h
- graphics/graphics_items.cpp

14.30 RelationSuggestion Struct Reference

Struct for relation suggestions.

```
#include <family_tree_items.h>
```

Public Member Functions

- RelationSuggestion (size_t lockedP, size_t suggestedP, bool par, bool sym, Trait t)

Public Attributes

- size_t lockedPerson

  The one locked person.
- bool parent

  If the locked person should be parent or not.
- size_t suggestedPerson

  Suggested person for the relation.
- bool symetrical

  If the relation is symmetrical or not.
- Trait trait

  What relation should it be.

14.30.1 Detailed Description

Struct for relation suggestions.

14.30.2 Constructor & Destructor Documentation

14.30.2.1 RelationSuggestion()

```
RelationSuggestion::RelationSuggestion (
            size_t lockedP,
            size_t suggestedP,
            bool par,
            bool sym,
            Trait t )
```

Default constructor.

Parameters

lockedP	Locked person.
suggestedP	Suggested person.
par	If it is parent.
sym	If it is symmetrical.
t	Given trait.

14.30.3 Member Data Documentation

14.30.3.1 lockedPerson

```
size_t RelationSuggestion::lockedPerson
```

The one locked person.

14.30.3.2 parent

```
bool RelationSuggestion::parent
```

If the locked person should be parent or not.

14.30.3.3 suggestedPerson

```
size_t RelationSuggestion::suggestedPerson
```

Suggested person for the relation.

14.30.3.4 symetrical

```
bool RelationSuggestion::symetrical
```

If the relation is symmetrical or not.

14.30.3.5 trait

```
Trait RelationSuggestion::trait
```

What relation should it be.

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

- core/family_tree_items.h
- core/family_tree_items.cpp

14.31 RelationTemplate Class Reference

Template for relation. What is the name, description. What is the generation difference. And how are persons in this relation called.

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

Public Member Functions

- RelationTemplate ()

  Default constructor.
- const std::string & getDescription () const
- const std::string & getFirstName () const
- int getGenerationDifference () const
- size_t getId () const
- const std::string & getSecondName () const
- const std::string & getTitle () const
- Trait getTrait () const
- void readJson (const Json::Value &value)
- std::string & setDescription ()
- std::string & setFirstName ()
- void setGenerationDifference (int diff)
- void setId (size_t id)
- std::string & setSecondName ()
- std::string & setTitle ()
- void setTrait (Trait trait)
- std::string str () const

Private Attributes

- std::string description_

  Description of the relation.
- std::string firstName_

  First name in the relation.
- int generationDiff_

  Generation difference: first generation - second generation.
- size_t id_

  Special ID of this template.
- std::string secondName_

  Second name in the relation.
- std::string title_

  Title of the relation.
- Trait trait_

  Special trait of this relation.

14.31.1 Detailed Description

Template for relation. What is the name, description. What is the generation difference. And how are persons in this relation called.

14.31.2 Constructor & Destructor Documentation

14.31.2.1 RelationTemplate()

```
RelationTemplate::RelationTemplate ( )
```

Default constructor.

14.31.3 Member Function Documentation

14.31.3.1 getDescription()

```
const std::string & RelationTemplate::getDescription ( ) const
```

Get the description.

Returns

Constant reference to the description.

14.31.3.2 getFirstName()

```
const std::string & RelationTemplate::getFirstName ( ) const
```

Get the first name of the relation.

Returns

Constant reference to the first name.

14.31.3.3 getGenerationDifference()

```
int RelationTemplate::getGenerationDifference ( ) const
```

Get the generation difference.

Returns

The difference between first and second generation.

14.31.3.4 getId()

```
size_t RelationTemplate::getId ( ) const
```

Get the ID of this relation template.

Returns

ID of this relation template.

14.31.3.5 getSecondName()

```
const std::string & RelationTemplate::getSecondName ( ) const
```

Get the second name of the relation.

Returns

Constant reference to the second name.

14.31.3.6 getTitle()

```
const std::string & RelationTemplate::getTitle ( ) const
```

Get the title of the relation.

Returns

Constant reference to the title.

14.31.3.7 getTrait()

```
Trait RelationTemplate::getTrait ( ) const
```

Get the trait of the relationship.

Returns

Trait of the relationship.

14.31.3.8 readJson()

```
void RelationTemplate::readJson (
            const Json::Value & value )
```

Read data from JSON.

Parameters

value	Given JSON value.

14.31.3.9 setDescription()

```
std::string & RelationTemplate::setDescription ( )
```

Set the description of the relation by giving the reference to the description.

Returns

Reference to the description.

14.31.3.10 setFirstName()

```
std::string & RelationTemplate::setFirstName ( )
```

Set the first name by giving the reference to the first name.

Returns

Reference to the first name.

14.31.3.11 setGenerationDifference()

```
void RelationTemplate::setGenerationDifference (
             int diff )
```

Set the generation difference (first - second).

Parameters

diff	Given difference.

14.31.3.12 setId()

```
void RelationTemplate::setId (
             size_t id )
```

Set the ID of this relation template.

Parameters

id	New ID.

14.31.3.13 setSecondName()

```
std::string & RelationTemplate::setSecondName ( )
```

Set the second name by giving the reference to the second name.

Returns

Reference to the second name.

14.31.3.14 setTitle()

```
std::string & RelationTemplate::setTitle ( )
```

Set the title of relation by giving the reference to the title.

Returns

Reference to the title.

14.31.3.15 setTrait()

```
void RelationTemplate::setTrait (
             Trait trait )
```

Set the trait for the relation.

Parameters

trait	Given trait.

14.31.3.16 str()

```
std::string RelationTemplate::str ( ) const
```

Represent relation template as a string.

Returns

Newly constructed string representing relation template.

14.31.4 Member Data Documentation

14.31.4.1 description_

```
std::string RelationTemplate::description_ [private]
```

Description of the relation.

14.31.4.2 firstName_

```
std::string RelationTemplate::firstName_ [private]
```

First name in the relation.

14.31.4.3 generationDiff_

```
int RelationTemplate::generationDiff_  [private]
```

Generation difference: first generation - second generation.

14.31.4.4 id_

```
size_t RelationTemplate::id_  [private]
```

Special ID of this template.

14.31.4.5 secondName_

```
std::string RelationTemplate::secondName_  [private]
```

Second name in the relation.

14.31.4.6 title_

```
std::string RelationTemplate::title_  [private]
```

Title of the relation.

14.31.4.7 trait_

```
Trait RelationTemplate::trait_  [private]
```

Special trait of this relation.

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

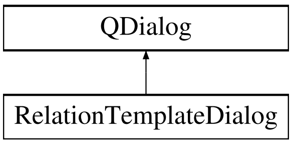
- core/config.h
- core/config.cpp

14.32 RelationTemplateDialog Class Reference

Dialog showing all templates and possibly editing them or creating new ones.

```
#include <dialogs.h>
```

Inheritance diagram for RelationTemplateDialog:



Public Slots

- void addDefaults ()

  Add default templates.
- void addNewTemplate ()

  Clear boxes and set them for new template.
- void removeTemplate ()

  Remove current template from the templates.
- void saveTemplate ()

  Either save edits to the template or add new item.
- void showTemplate (QTreeWidgetItem ∗item)

Public Member Functions

- RelationTemplateDialog (FamilyTree ∗FT, QWidget ∗parent=nullptr)
- ∼RelationTemplateDialog ()

  Default destructor.

Private Member Functions

- void clear ()

  Clear all text boxes.
- void refreshListWidget ()

  Refresh list of all relation templates.
- void setEnabledWidgets (bool enabled)
- void saveTemplate (RelationTemplate ∗rel)

Private Attributes

- bool addingNew_

  When we are adding new template.
- FamilyTree ∗ FT_

  Pointer to the family tree.
- Ui::RelationTemplateDialog ∗ ui

  User interface of the Qt framework.

14.32.1 Detailed Description

Dialog showing all templates and possibly editing them or creating new ones.

14.32.2 Constructor & Destructor Documentation

14.32.2.1 RelationTemplateDialog()

```
RelationTemplateDialog::RelationTemplateDialog (
            FamilyTree ∗ FT,
            QWidget ∗ parent = nullptr )  [explicit]
```

Default constructor.

Parameters

FT	Pointer to the family tree.
parent	The Qt Widget parent.

14.32.2.2 ∼RelationTemplateDialog()

```
RelationTemplateDialog::∼RelationTemplateDialog ( )
```

Default destructor.

14.32.3 Member Function Documentation

14.32.3.1 addDefaults

```
void RelationTemplateDialog::addDefaults ( )  [slot]
```

Add default templates.

14.32.3.2 addNewTemplate

```
void RelationTemplateDialog::addNewTemplate ( )  [slot]
```

Clear boxes and set them for new template.

14.32.3.3 clear()

```
void RelationTemplateDialog::clear ( )  [private]
```

Clear all text boxes.

14.32.3.4 refreshListWidget()

```
void RelationTemplateDialog::refreshListWidget ( )  [private]
```

Refresh list of all relation templates.

14.32.3.5 removeTemplate

```
void RelationTemplateDialog::removeTemplate ( )  [slot]
```

Remove current template from the templates.

14.32.3.6 saveTemplate [1/2]

```
void RelationTemplateDialog::saveTemplate ( )  [slot]
```

Either save edits to the template or add new item.

14.32.3.7 saveTemplate() [2/2]

```
void RelationTemplateDialog::saveTemplate (
             RelationTemplate * rel )  [private]
```

Store the given template from ui boxes.

Parameters

rel	Pointer to which relation to overwrite.

14.32.3.8 setEnabledWidgets()

```
void RelationTemplateDialog::setEnabledWidgets (
            bool enabled )  [private]
```

Set enabled for all widgets.

Parameters

enabled	If the widgets should be enabled or not.

14.32.3.9 showTemplate

```
void RelationTemplateDialog::showTemplate (
            QTreeWidgetItem * item )  [slot]
```

Show template details to its text boxes.

Parameters

item	Which item was clicked.

14.32.4 Member Data Documentation

14.32.4.1 addingNew_

```
bool RelationTemplateDialog::addingNew_  [private]
```

When we are adding new template.

14.32.4.2 FT_

```
FamilyTree* RelationTemplateDialog::FT_  [private]
```

Pointer to the family tree.

14.32.4.3 ui

```
Ui::RelationTemplateDialog* RelationTemplateDialog::ui  [private]
```

User interface of the Qt framework.

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

- graphics/dialogs.h
- graphics/dialogs.cpp

14.33 Settings Class Reference

Class holding all data about the settings of the app and all templates.

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

Public Member Functions

- Settings ()

  Default constructor.
- void addDefaultEvents ()

  Add default event templates.
- void addDefaultRelations ()

  Add default relation templates.
- size_t addEventTemplate (EventTemplate &et)
- size_t addRelationTemplate (RelationTemplate &rt)
- void addTag (const std::string &tag)
- void clear ()

  Clear all data.
- void exportTemplates (std::ostream &os, const std::vector< size_t > &relIds, const std::vector< size_t > &eventIds)
- const AppSettings & getAppSettings () const
- std::optional< EventTemplate * > getEventTemplate (size_t id) const
- const std::map< size_t, std::unique_ptr< EventTemplate > > & getEventTemplates () const
- size_t getGlobalMainPerson () const
- std::optional< RelationTemplate * > getRelationTemplate (size_t id) const
- const std::map< size_t, std::unique_ptr< RelationTemplate > > & getRelationTemplates () const
- const std::vector< std::string > & getTags () const
- size_t importTemplates (const Json::Value &value)
- std::pair< size_t, size_t > indexes ()
- void readJson (const Json::Value &value)
- void reIndex (size_t plusEvent, size_t plusRel)
- void removeEventTemplate (size_t index)
- void removeRelationTemplate (size_t index)
- void removeTag (const std::string &tag)
- AppSettings & setAppSettings ()
- void setGlobalMainPerson (size_t id)

Private Member Functions

- void addDefaultEvent (const std::string &title, const std::string &description, bool date, bool place, bool morePeople, std::vector< std::string > roles)
- void addDefaultRelation (const std::string &title, const std::string &description, const std::string &firstName, const std::string &secondName, int genDiff, Trait trait)

Private Attributes

- AppSettings appSettings_

  Application settings.
- size_t eventTemplateIndex_

  First empty index for event template.
- std::map< size_t, std::unique_ptr< EventTemplate > > eventTemplates_

  All events templates.
- size_t globalMainPerson_

  Main person that is global, so that on app start it will be chosen as the main person.
- size_t relationTemplateIndex_

  First empty index for relation template.
- std::map< size_t, std::unique_ptr< RelationTemplate > > relationsTemplates_

  All relationsTemplates.
- std::vector< std::string > tags_

  All tags.

14.33.1 Detailed Description

Class holding all data about the settings of the app and all templates.

14.33.2 Constructor & Destructor Documentation

14.33.2.1 Settings()

```
Settings::Settings ( )
```

Default constructor.

14.33.3 Member Function Documentation

14.33.3.1 addDefaultEvent()

```
void Settings::addDefaultEvent (
            const std::string & title,
            const std::string & description,
            bool date,
            bool place,
            bool morePeople,
            std::vector< std::string > roles )  [private]
```

Add single default event.

Parameters

title	Title of the new event template.
description	Description of the new event template.
date	If the event template contains date.
place	If the event template contains date.
morePeople	If more people are present.
roles	Which roles, if any, the event template has.

14.33.3.2 addDefaultEvents()

```
void Settings::addDefaultEvents ( )
```

Add default event templates.

14.33.3.3 addDefaultRelation()

```
void Settings::addDefaultRelation (
             const std::string & title,
             const std::string & description,
             const std::string & firstName,
             const std::string & secondName,
             int genDiff,
             Trait trait )  [private]
```

Add single default relation.

Parameters

title	Title of the new relation template.
description	Description of the new relation template.
firstName	First name of the person.
secondName	Second name of the person.
genDiff	Generation difference.
trait	Which trait it has.

14.33.3.4 addDefaultRelations()

```
void Settings::addDefaultRelations ( )
```

Add default relation templates.

14.33.3.5 addEventTemplate()

```
size_t Settings::addEventTemplate (
             EventTemplate & et )
```

Add event template to the vector. Also re-index if necessary (id = 0).

Parameters

et	Given event template.

Returns

Id of the template.

14.33.3.6 addRelationTemplate()

```
size_t Settings::addRelationTemplate (
            RelationTemplate & rt )
```

Add relation event to the vector. Also re-index if necessary (id = 0).

Parameters

rt	Given relation template.

Returns

Id of the template.

14.33.3.7 addTag()

```
void Settings::addTag (
            const std::string & tag )
```

Add tag to the vector. If it is not present.

Parameters

tag	New tag.

14.33.3.8 clear()

```
void Settings::clear ( )
```

Clear all data.

14.33.3.9 exportTemplates()

```
void Settings::exportTemplates (
            std::ostream & os,
            const std::vector< size_t > & relIds,
            const std::vector< size_t > & eventIds )
```

Export only certain templates in JSON.

Parameters

os	Which output stream to use for writing the data.
relIds	Vector of relation templates ids.
eventIds	Vector of event templates ids.

14.33.3.10 getAppSettings()

```
const AppSettings & Settings::getAppSettings ( ) const
```

Get the app settings.

Returns

Constant reference to the app settings.

14.33.3.11 getEventTemplate()

```
std::optional< EventTemplate * > Settings::getEventTemplate (
             size_t id ) const
```

Get exactly one event template.

Parameters

id	The id of the event.

Returns

Optionally pointer to the template or nothing.

14.33.3.12 getEventTemplates()

```
const std::map< size_t, std::unique_ptr< EventTemplate > > & Settings::getEventTemplates ( )
const
```

Get all the event templates.

Returns

Constant reference to the vector of all templates.

14.33.3.13 getGlobalMainPerson()

```
size_t Settings::getGlobalMainPerson ( ) const
```

Get the global main person.

Returns

Id of the main person.

14.33.3.14 getRelationTemplate()

```
std::optional< RelationTemplate * > Settings::getRelationTemplate (
             size_t id ) const
```

Get exactly one relation template.

Parameters

id	The id of the relation.

Returns

Optionally pointer to the template or nothing.

14.33.3.15 getRelationTemplates()

```
const std::map< size_t, std::unique_ptr< RelationTemplate > > & Settings::getRelationTemplates
( ) const
```

Get all the relation templates.

Returns

Constant reference to the vector of all templates.

14.33.3.16 getTags()

```
const std::vector< std::string > & Settings::getTags ( ) const
```

Return the reference to all tags.

Returns

Constant reference to the container.

14.33.3.17 importTemplates()

```
size_t Settings::importTemplates (
             const Json::Value & value )
```

Import templates from JSON value.

Parameters

value	Given JSON Value.

Returns

Number of imported templates.

14.33.3.18 indexes()

```
std::pair< size_t, size_t > Settings::indexes ( )
```

Get the free index for templates.

Returns

Pair of event templates index and then relation templates.

14.33.3.19 readJson()

```
void Settings::readJson (
             const Json::Value & value )
```

Read the configuration from its JSON value.

Parameters

value	Given JSON value.

14.33.3.20 reIndex()

```
void Settings::reIndex (
             size_t plusEvent,
             size_t plusRel )
```

Re-index all templates to plus given number.

Parameters

plusEvent	What is the number to add to event templates.
plusRel	What is the number to add to relation templates.

14.33.3.21 removeEventTemplate()

```
void Settings::removeEventTemplate (
             size_t index )
```

Remove event template from all events.

Parameters

index	On which position the event in vector is.

14.33.3.22 removeRelationTemplate()

```
void Settings::removeRelationTemplate (
             size_t index )
```

Remove relation template from all relations.

Parameters

index	On which position the relation in vector is.

14.33.3.23 removeTag()

```
void Settings::removeTag (
             const std::string & tag )
```

Remove tag from the container.

Parameters

tag	Delete same tag if it exists.

14.33.3.24 setAppSettings()

```
AppSettings & Settings::setAppSettings ( )
```

Get the reference to the app settings to edit it.

Returns

Reference to the app settings.

14.33.3.25 setGlobalMainPerson()

```
void Settings::setGlobalMainPerson (
             size_t id )
```

Set the global main person.

Parameters

id	Id of the new main person.

14.33.4 Member Data Documentation

14.33.4.1 appSettings_

```
AppSettings Settings::appSettings_  [private]
```

Application settings.

14.33.4.2 eventTemplateIndex_

```
size_t Settings::eventTemplateIndex_ [private]
```

First empty index for event template.

14.33.4.3 eventTemplates_

```
std::map<size_t, std::unique_ptr<EventTemplate> > Settings::eventTemplates_ [private]
```

All events templates.

14.33.4.4 globalMainPerson_

```
size_t Settings::globalMainPerson_ [private]
```

Main person that is global, so that on app start it will be chosen as the main person.

14.33.4.5 relationsTemplates_

```
std::map<size_t, std::unique_ptr<RelationTemplate> > Settings::relationsTemplates_ [private]
```

All relationsTemplates.

14.33.4.6 relationTemplateIndex_

```
size_t Settings::relationTemplateIndex_ [private]
```

First empty index for relation template.

14.33.4.7 tags_

```
std::vector<std::string> Settings::tags_ [private]
```

All tags.

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

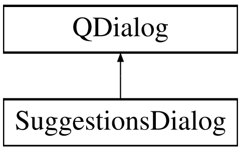
- core/config.h
- core/config.cpp

14.34 SuggestionsDialog Class Reference

Dialog for relation suggestions.

```
#include <dialogs.h>
```

Inheritance diagram for SuggestionsDialog:



Public Slots

- void removeSuggestion ()

  Remove current suggestion.
- void save ()

  Save all suggestions.
- void selectedSuggestion (QTreeWidgetItem ∗item)
- void swapPersons ()

  Swap persons roles.
- void switchedTemplate ()

  Switch the template description.

Public Member Functions

- SuggestionsDialog (FamilyTree ∗FT, std::vector< RelationSuggestion > suggestions, QWidget ∗parent=nullptr)
- ∼SuggestionsDialog ()

  Default destructor.

Private Attributes

- std::vector< RelationTemplate ∗ > fatherhoods

  Vector of all templates for fatherhood relations.
- FamilyTree ∗ FT_

  Pointer to the family tree.
- std::vector< RelationTemplate ∗ > motherhoods

  Vector of all templates for motherhood relations.
- std::vector< RelationTemplate ∗ > partnerships

  Vector of all templates for partnership relations.
- std::vector< RelationTemplate ∗ > siblings

  Vector of all templates for siblings relations.
- Ui::SuggestionsDialog ∗ ui

  Used Qt user interface.

14.34.1 Detailed Description

Dialog for relation suggestions.

14.34.2 Constructor & Destructor Documentation

14.34.2.1 SuggestionsDialog()

```
SuggestionsDialog::SuggestionsDialog (
            FamilyTree * FT,
            std::vector< RelationSuggestion > suggestions,
            QWidget * parent = nullptr )
```

Default constructor.

Parameters

FT	Pointer to the family tree.
suggestions	All the suggestions for relations.
parent	The Qt Widget parent.

14.34.2.2 ∼SuggestionsDialog()

```
SuggestionsDialog::∼SuggestionsDialog ( )
```

Default destructor.

14.34.3 Member Function Documentation

14.34.3.1 removeSuggestion

```
void SuggestionsDialog::removeSuggestion ( )  [slot]
```

Remove current suggestion.

14.34.3.2 save

```
void SuggestionsDialog::save ( )  [slot]
```

Save all suggestions.

14.34.3.3 selectedSuggestion

```
void SuggestionsDialog::selectedSuggestion (
            QTreeWidgetItem * item )  [slot]
```

When a suggestion is clicked.

Parameters

item	Which suggestion was clicked.

14.34.3.4 swapPersons

```
void SuggestionsDialog::swapPersons ( )  [slot]
```

Swap persons roles.

14.34.3.5 switchedTemplate

```
void SuggestionsDialog::switchedTemplate ( )  [slot]
```

Switch the template description.

14.34.4 Member Data Documentation

14.34.4.1 fatherhoods

```
std::vector<RelationTemplate*> SuggestionsDialog::fatherhoods  [private]
```

Vector of all templates for fatherhood relations.

14.34.4.2 FT_

```
FamilyTree* SuggestionsDialog::FT_  [private]
```

Pointer to the family tree.

14.34.4.3 motherhoods

```
std::vector<RelationTemplate*> SuggestionsDialog::motherhoods  [private]
```

Vector of all templates for motherhood relations.

14.34.4.4 partnerships

```
std::vector<RelationTemplate*> SuggestionsDialog::partnerships  [private]
```

Vector of all templates for partnership relations.

14.34.4.5 siblings

```
std::vector<RelationTemplate*> SuggestionsDialog::siblings  [private]
```

Vector of all templates for siblings relations.

14.34.4.6 ui

```
Ui::SuggestionsDialog* SuggestionsDialog::ui  [private]
```

Used Qt user interface.

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

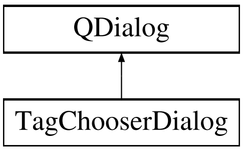
- graphics/dialogs.h
- graphics/dialogs.cpp

14.35 TagChooserDialog Class Reference

Dialog for editing or creating tags for a person.

```
#include <dialogs.h>
```

Inheritance diagram for TagChooserDialog:



Public Slots

- void changeVisibleWidgets (int index)
- void saveTag ()

  Save the tag.

Public Member Functions

- TagChooserDialog (FamilyTree ∗FT, const QString &tag, const QString &value, QWidget ∗parent=nullptr)
- ∼TagChooserDialog ()

  Default destructor.

Private Attributes

- FamilyTree ∗ FT_

  Pointer to the family tree.
- bool newOne_

  If the tag is being edited or new one is being created.
- Ui::TagChooserDialog ∗ ui

  User interface of the Qt framework.

14.35.1 Detailed Description

Dialog for editing or creating tags for a person.

14.35.2 Constructor & Destructor Documentation

14.35.2.1 TagChooserDialog()

```
TagChooserDialog::TagChooserDialog (
             FamilyTree * FT,
             const QString & tag,
             const QString & value,
             QWidget * parent = nullptr )
```

Default constructor.

Parameters

FT	Pointer to the family tree.
tag	What to show as a tag.
value	What to show as a value of the tag.
parent	The Qt Widget parent.

14.35.2.2 ∼TagChooserDialog()

```
TagChooserDialog::∼TagChooserDialog ( )
```

Default destructor.

14.35.3 Member Function Documentation

14.35.3.1 changeVisibleWidgets

```
void TagChooserDialog::changeVisibleWidgets (
             int index )  [slot]
```

Slot for showing/hiding custom tag edit.

Parameters

index	Which index was from the combo box chosen.

14.35.3.2 saveTag

```
void TagChooserDialog::saveTag ( )  [slot]
```

Save the tag.

14.35.4 Member Data Documentation

14.35.4.1 FT_

```
FamilyTree* TagChooserDialog::FT_ [private]
```

Pointer to the family tree.

14.35.4.2 newOne_

```
bool TagChooserDialog::newOne_ [private]
```

If the tag is being edited or new one is being created.

14.35.4.3 ui

```
Ui::TagChooserDialog* TagChooserDialog::ui [private]
```

User interface of the Qt framework.

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

- graphics/dialogs.h
- graphics/dialogs.cpp

14.36 VirtualDrive Class Reference

Class for virtualization of folder structure for each person.

```
#include <person.h>
```

Public Member Functions

- VirtualDrive ()=default

  Default constructor.
- VirtualDrive (const std::string &name)
- void addFile (size_t fileId)
- VirtualDrive * addSubdrive ()
- VirtualDrive * addSubdrive (const std::string &name)
- void copyDrive (const VirtualDrive &other, size_t plusIndex=0)
- const std::vector< size_t > & getFiles () const
- const std::string & getName () const
- const std::vector< std::unique_ptr< VirtualDrive > > & getSubdrives () const
- void readJson (const Json::Value &value)
- void remove (const std::string &subfolderName)
- void removeFile (size_t id)
- void rename (const std::string &name)
- void usedFiles (std::vector< size_t > &used) const

Private Attributes

- std::vector< size_t > files_

  Container of all files.
- std::string name_

  Name of the folder.
- std::vector< std::unique_ptr< VirtualDrive > > subdrives_

  Container of pointers to the sub-folders.

14.36.1 Detailed Description

Class for virtualization of folder structure for each person.

14.36.2 Constructor & Destructor Documentation

14.36.2.1 VirtualDrive() [1/2]

```
VirtualDrive::VirtualDrive ( )  [default]
```

Default constructor.

14.36.2.2 VirtualDrive() [2/2]

```
VirtualDrive::VirtualDrive (
             const std::string & name )  [explicit]
```

Constructor for known name.

Parameters

name	Name of this folder.

14.36.3 Member Function Documentation

14.36.3.1 addFile()

```
void VirtualDrive::addFile (
             size_t fileId )
```

Add file to the folder by its id. Only if it is not already there.

Parameters

file↩ Id	Id of the file.

14.36.3.2 addSubdrive() [1/2]

```
VirtualDrive * VirtualDrive::addSubdrive ( )
```

Add sub-folder to this folder.

Returns

Pointer to the new sub-folder.

14.36.3.3 addSubdrive() [2/2]

```
VirtualDrive * VirtualDrive::addSubdrive (
             const std::string & name )
```

Add sub-folder to this folder.

Parameters

name	Name of the sub-folder.

Returns

Pointer to the new sub-folder.

14.36.3.4 copyDrive()

```
void VirtualDrive::copyDrive (
             const VirtualDrive & other,
             size_t plusIndex = 0 )
```

Recursively copy the other drive. Used for import mainly.

Parameters

other	Other drive.
plusIndex	What should be added to each file index.

14.36.3.5 getFiles()

```
const std::vector< size_t > & VirtualDrive::getFiles ( ) const
```

Get constant reference to the container of files.

Returns

Constant reference to the container of files.

14.36.3.6 getName()

```
const std::string & VirtualDrive::getName ( ) const
```

Get the name of the folder.

Returns

Name of the folder.

14.36.3.7 getSubdrives()

```
const std::vector< std::unique_ptr< VirtualDrive > > & VirtualDrive::getSubdrives ( ) const
```

Get constant reference to the container of files.

Returns

Constant reference to the container of pointers to virtual drives.

14.36.3.8 readJson()

```
void VirtualDrive::readJson (
             const Json::Value & value )
```

Load data from JSON value.

Parameters

value	Given JSON value.

14.36.3.9 remove()

```
void VirtualDrive::remove (
             const std::string & subfolderName )
```

Remove sub-folder and all its sub-folders and files.

Parameters

subfolderName	Name of the sub-folder.

14.36.3.10 removeFile()

```
void VirtualDrive::removeFile (
             size_t id )
```

Remove one file in this folder.

Parameters

id	Id of the file.

14.36.3.11 rename()

```
void VirtualDrive::rename (
            const std::string & name )
```

Rename this folder.

Parameters

name	New name of this folder.

14.36.3.12 usedFiles()

```
void VirtualDrive::usedFiles (
            std::vector< size_t > & used ) const
```

Vector of all used files.

Parameters

used	Vector of all files contained in this folder and sub-folders.

14.36.4 Member Data Documentation

14.36.4.1 files_

```
std::vector<size_t> VirtualDrive::files_  [private]
```

Container of all files.

14.36.4.2 name_

```
std::string VirtualDrive::name_  [private]
```

Name of the folder.

14.36.4.3 subdrives_

```
std::vector<std::unique_ptr<VirtualDrive> > VirtualDrive::subdrives_  [private]
```

Container of pointers to the sub-folders.

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

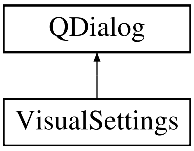
- core/person.h
- core/person.cpp

14.37 VisualSettings Class Reference

Dialog for editing and showing graphic settings.

```
#include <dialogs.h>
```

Inheritance diagram for VisualSettings:



Public Slots

- void highlightedColor ()

  Show dialog for color change upon clicking on highlighted color change.
- void lineColor ()

  Change the color of line.
- void refresh ()

  Upon clicking the refresh button load the values and show the graphics.
- void probandColor ()

  Show dialog for color change upon clicking on proband color change.
- void promotedColor ()

  Show dialog for color change upon clicking on promoted color change.
- void saveAndExit ()

  Emit signal for saving the configuration and exit the dialog.
- void standardColor ()

  Show dialog for color change upon clicking on standard color change.
- void textColor ()

  Change the color of a text.

Signals

- void saveSettings (QColor probandColor, QColor promotedColor, QColor standardColor, QColor highlightedColor, int sizeX, int sizeY, QFont font, QColor lineColor, QColor textColor, int lineWidth, int rounding)

Public Member Functions

- VisualSettings (QColor probandColor, QColor promotedColor, QColor standardColor, QColor highlighted, int sizeX, int sizeY, QFont font, QPen ∗pen, QPen ∗textPen, int borderRadius, QWidget ∗parent=nullptr)
- ∼VisualSettings ()

  Default destructor.

Private Attributes

- int borderRadius_

  Border radius of the box.
- QFont font_

  Used font.
- QColor highlightedColor_

  Color for highlighting the box.
- QPen pen_

  Which pen is used for lines.
- QColor probandColor_

  Color for proband.
- QColor promotedColor_

  Color for promoted relations.
- QGraphicsScene ∗ scene

  Used graphics scene.
- int sizeX_

  Size of the box in x axis.
- int sizeY_

  Size of the box in y axis.
- QColor standardColor_

  Color for standard relations.
- QPen textPen_

  Pen used for texts.
- Ui::VisualSettings ∗ ui

  Used Qt user interface.

14.37.1 Detailed Description

Dialog for editing and showing graphic settings.

14.37.2 Constructor & Destructor Documentation

14.37.2.1 VisualSettings()

```
VisualSettings::VisualSettings (
             QColor probandColor,
             QColor promotedColor,
             QColor standardColor,
             QColor highlighted,
             int sizeX,
             int sizeY,
             QFont font,
             QPen ∗ pen,
             QPen ∗ textPen,
             int borderRadius,
             QWidget ∗ parent = nullptr )
```

Default constructor.

Parameters

probandColor	Color used for proband.
promotedColor	Color for promoted Relations.
standardColor	Standard color.
highlighted	Color for highlighting the box.
sizeX	Size of the box in x axis.
sizeY	Size of the box in y axis.
font	Used font for the text in the box.
pen	Which pen is used for lines.
textPen	Which pen is used for text.
borderRadius	What is the border radius of the box.
parent	The Qt Widget parent.

14.37.2.2 ∼VisualSettings()

```
VisualSettings::∼VisualSettings ( )
```

Default destructor.

14.37.3 Member Function Documentation

14.37.3.1 highlightedColor

```
void VisualSettings::highlightedColor ( )  [slot]
```

Show dialog for color change upon clicking on highlighted color change.

14.37.3.2 lineColor

```
void VisualSettings::lineColor ( )  [slot]
```

Change the color of line.

14.37.3.3 probandColor

```
void VisualSettings::probandColor ( )  [slot]
```

Show dialog for color change upon clicking on proband color change.

14.37.3.4 promotedColor

```
void VisualSettings::promotedColor ( )  [slot]
```

Show dialog for color change upon clicking on promoted color change.

14.37.3.5 refresh

```
void VisualSettings::refresh ( )  [slot]
```

Upon clicking the refresh button load the values and show the graphics.

14.37.3.6 saveAndExit

```
void VisualSettings::saveAndExit ( )  [slot]
```

Emit signal for saving the configuration and exit the dialog.

14.37.3.7 saveSettings

```
void VisualSettings::saveSettings (
             QColor probandColor,
             QColor promotedColor,
             QColor standardColor,
             QColor highlightedColor,
             int sizeX,
             int sizeY,
             QFont font,
             QColor lineColor,
             QColor textColor,
             int lineWidth,
             int rounding )  [signal]
```

Signal to emit when OK button is pressed to save this configuration.

Parameters

probandColor	Color used for proband.
promotedColor	Color for promoted Relations.
standardColor	Standard color.
highlightedColor	Highlighted color.
sizeX	Size of the box in x axis.
sizeY	Size of the box in y axis.
font	Used font for the text in the box.
lineColor	Color for lines.
textColor	Color for texts.
lineWidth	Width of the lines.
rounding	Rounding of the boxes.

14.37.3.8 standardColor

```
void VisualSettings::standardColor ( )  [slot]
```

Show dialog for color change upon clicking on standard color change.

14.37.3.9 textColor

```
void VisualSettings::textColor ( )  [slot]
```

Change the color of a text.

14.37.4 Member Data Documentation

14.37.4.1 borderRadius_

```
int VisualSettings::borderRadius_  [private]
```

Border radius of the box.

14.37.4.2 font_

```
QFont VisualSettings::font_  [private]
```

Used font.

14.37.4.3 highlightedColor_

```
QColor VisualSettings::highlightedColor_  [private]
```

Color for highlighting the box.

14.37.4.4 pen_

```
QPen VisualSettings::pen_  [private]
```

Which pen is used for lines.

14.37.4.5 probandColor_

```
QColor VisualSettings::probandColor_  [private]
```

Color for proband.

14.37.4.6 promotedColor_

```
QColor VisualSettings::promotedColor_  [private]
```

Color for promoted relations.

14.37.4.7 scene

```
QGraphicsScene* VisualSettings::scene  [private]
```

Used graphics scene.

14.37.4.8 sizeX_

```
int VisualSettings::sizeX_  [private]
```

Size of the box in x axis.

14.37.4.9 sizeY_

```
int VisualSettings::sizeY_  [private]
```

Size of the box in y axis.

14.37.4.10 standardColor_

```
QColor VisualSettings::standardColor_  [private]
```

Color for standard relations.

14.37.4.11 textPen_

```
QPen VisualSettings::textPen_  [private]
```

Pen used for texts.

14.37.4.12 ui

```
Ui::VisualSettings* VisualSettings::ui  [private]
```

Used Qt user interface.

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

- graphics/dialogs.h
- graphics/dialogs.cpp

14.38 WrappedDate Class Reference

Class holding two dates and date description. Always first date is the main one.

```
#include <date.h>
```

Public Member Functions

- WrappedDate ()

  Default constructor.
- void clear ()

  Clear all values.
- const Date & getFirstDate () const
- const Date & getSecondDate () const
- const std::string & getText () const
- bool isExact () const
- bool isUnknown () const
- void readJson (const Json::Value &value)
- void setFirstDate (int y, int m, int d)
- void setSecondDate (int y, int m, int d)
- std::string & setText ()
- std::string str () const
- void updateDate ()

  Switch the dates if the first one is empty.

Private Attributes

- Date date1_

  Main date, if interval is set it is the first day.
- Date date2_

  Second date for interval only, otherwise empty.
- std::string text_

  If text was given.

14.38.1 Detailed Description

Class holding two dates and date description. Always first date is the main one.

14.38.2 Constructor & Destructor Documentation

14.38.2.1 WrappedDate()

```
WrappedDate::WrappedDate ( )
```

Default constructor.

14.38.3 Member Function Documentation

14.38.3.1 clear()

```
void WrappedDate::clear ( )
```

Clear all values.

14.38.3.2 getFirstDate()

```
const Date & WrappedDate::getFirstDate ( ) const
```

Get the first date.

Returns

Constant reference to the first date.

14.38.3.3 getSecondDate()

```
const Date & WrappedDate::getSecondDate ( ) const
```

Get the second date.

Returns

Constant reference to the second date.

14.38.3.4 getText()

```
const std::string & WrappedDate::getText ( ) const
```

Get the text.

Returns

Constant reference to the text.

14.38.3.5 isExact()

```
bool WrappedDate::isExact ( ) const
```

Check if the date is exact or not. To be exact only the first date has to be exact, the second got to be empty.

Returns

True if the first date is exact & second empty.

14.38.3.6 isUnknown()

```
bool WrappedDate::isUnknown ( ) const
```

Check if the date is unknown. To be unknown all values are 0 and text is empty.

Returns

If the date is unknown.

14.38.3.7 readJson()

```
void WrappedDate::readJson (
             const Json::Value & value )
```

Read the data from JSON formatting.

Parameters

value	Given JSON value.

14.38.3.8 setFirstDate()

```
void WrappedDate::setFirstDate (
             int y,
             int m,
             int d )
```

Set the first date.

Parameters

y	Year number.
m	Month number.
d	Day number.

14.38.3.9 setSecondDate()

```
void WrappedDate::setSecondDate (
             int y,
             int m,
             int d )
```

Set the second date.

Parameters

y	Year number.
m	Month number.
d	Day number.

14.38.3.10 setText()

```
std::string & WrappedDate::setText ( )
```

Set the text for the date by giving the reference to the text.

Returns

Text used in the date.

14.38.3.11 str()

```
std::string WrappedDate::str ( ) const
```

Represent the date by a string.

Returns

Newly constructed string.

14.38.3.12 updateDate()

```
void WrappedDate::updateDate ( )
```

Switch the dates if the first one is empty.

14.38.4 Member Data Documentation

14.38.4.1 date1_

```
Date WrappedDate::date1_ [private]
```

Main date, if interval is set it is the first day.

14.38.4.2 date2_

```
Date WrappedDate::date2_ [private]
```

Second date for interval only, otherwise empty.

14.38.4.3 text_

```
std::string WrappedDate::text_ [private]
```

If text was given.

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

- core/date.h
- core/date.cpp

Chapter 15

File Documentation

15.1 core/config.cpp File Reference

Source code for configuring the app such as special evens and relations.

```
#include ¨config.h¨
```

Functions

- std::ostream & operator<< (std::ostream &os, const EventTemplate &et)
- std::ostream & operator<< (std::ostream &os, const RelationTemplate &rt)
- std::ostream & operator<< (std::ostream &os, const Color &color)
- std::ostream & operator<< (std::ostream &os, const AppSettings &as)
- std::ostream & operator<< (std::ostream &os, const Settings &settings)

15.1.1 Detailed Description

Source code for configuring the app such as special evens and relations.

15.1.2 Function Documentation

15.1.2.1 operator<<() [1/5]

```
std::ostream & operator<< (
            std::ostream & os,
            const AppSettings & as )
```

Output the app settings to stream in JSON format.

Parameters

os	Given stream.
as	Given app settings.

Returns

Reference to the changed stream.

15.1.2.2 operator<<() [2/5]

```
std::ostream & operator<< (
            std::ostream & os,
            const Color & color )
```

Output the color to stream in JSON format.

Parameters

os	Given stream.
color	Given color.

Returns

Reference to the changed stream.

15.1.2.3 operator<<() [3/5]

```
std::ostream & operator<< (
            std::ostream & os,
            const EventTemplate & et )
```

Output the event template to stream in JSON format.

Parameters

os	Given stream.
et	Given event template.

Returns

Reference to the changed stream.

15.1.2.4 operator<<() [4/5]

```
std::ostream & operator<< (
            std::ostream & os,
            const RelationTemplate & rt )
```

Output the relation template to stream in JSON format.

Parameters

os	Given stream.
rt	Given relation template.

Returns

Reference to the changed stream.

15.1.2.5 operator<<() [5/5]

```
std::ostream & operator<< (
            std::ostream & os,
            const Settings & setting )
```

Output all the settings to stream in JSON format.

Parameters

os	Given stream.
setting	Given settings.

Returns

Reference to the changed stream.

15.2 core/config.h File Reference

Header file for configuration of the app. Also includes templates for events and relations.

```
#include <vector>
#include <string>
#include <ostream>
#include <map>
#include <optional>
#include <json/json.h>
#include <climits>
#include <algorithm>
#include ¨strings.h¨
```

Classes

- class EventTemplate

  Template for events. It specifies whether the event includes a date, a place or if more persons are involved and if they have some special roles.
- class RelationTemplate

  Template for relation. What is the name, description. What is the generation difference. And how are persons in this relation called.

- struct Color

  Struct for general color.
- struct AppSettings

  Struct for app settings.
- class Settings

  Class holding all data about the settings of the app and all templates.

Namespaces

- namespace trait

  Namespace for everything with relation traits.

Enumerations

- enum Trait {
  None , Fatherhood , Motherhood , Partnership ,
  Sibling }

  Common traits of each relations.
- enum TabPosition { North , South , East , West }

  Enumeration for tab position.

Functions

- std::ostream & operator<< (std::ostream &os, const EventTemplate &et)
- std::string trait::str (Trait t)
- std::ostream & operator<< (std::ostream &os, const RelationTemplate &rt)
- std::ostream & operator<< (std::ostream &os, const Color &color)
- std::ostream & operator<< (std::ostream &os, const AppSettings &as)
- std::ostream & operator<< (std::ostream &os, const Settings &setting)

Variables

- constexpr Trait trait::AllTraits [ ] = {None, Fatherhood, Motherhood, Partnership, Sibling}

  All possible traits.

15.2.1 Detailed Description

Header file for configuration of the app. Also includes templates for events and relations.

15.2.2 Enumeration Type Documentation

15.2.2.1 TabPosition

```
enum TabPosition
```

Enumeration for tab position.

Enumerator

North	
South	
East	
West	

15.2.2.2 Trait

```
enum Trait
```

Common traits of each relations.

Enumerator

None	
Fatherhood	
Motherhood	
Partnership	
Sibling	

15.2.3 Function Documentation

15.2.3.1 operator<<() [1/5]

```
std::ostream & operator<< (
             std::ostream & os,
             const AppSettings & as )
```

Output the app settings to stream in JSON format.

Parameters

os	Given stream.
as	Given app settings.

Returns

Reference to the changed stream.

15.2.3.2 operator<<() [2/5]

```
std::ostream & operator<< (
             std::ostream & os,
             const Color & color )
```

Output the color to stream in JSON format.

Parameters

os	Given stream.
color	Given color.

Returns

Reference to the changed stream.

15.2.3.3 operator<<() [3/5]

```
std::ostream & operator<< (
            std::ostream & os,
            const EventTemplate & et )
```

Output the event template to stream in JSON format.

Parameters

os	Given stream.
et	Given event template.

Returns

Reference to the changed stream.

15.2.3.4 operator<<() [4/5]

```
std::ostream & operator<< (
            std::ostream & os,
            const RelationTemplate & rt )
```

Output the relation template to stream in JSON format.

Parameters

os	Given stream.
rt	Given relation template.

Returns

Reference to the changed stream.

15.2.3.5 operator<<() [5/5]

```
std::ostream & operator<< (
            std::ostream & os,
            const Settings & setting )
```

Output all the settings to stream in JSON format.

Parameters

os	Given stream.
setting	Given settings.

Returns

Reference to the changed stream.

15.3 config.h

Go to the documentation of this file.

```

#ifndef config_h_
#define config_h_

#include <vector>
#include <string>
#include <ostream>
#include <map>
#include <optional>
#include <json/json.h>
#include <climits>
#include <algorithm>
#include "strings.h"

class EventTemplate{
    public:
        EventTemplate();
        bool containsDate() const;
        bool containsPlace() const;
        const std::string& getDescription() const;
        size_t getId() const;
        const std::vector<std::string>& getRoles() const;
        const std::string& getTitle() const;
        bool hasMorePeopleInvolved() const;
        bool hasRoles() const;
        void readJson(const Json::Value& value);
        void setContainsDate(bool hasDate);
        void setContainsPlace(bool hasPlace);
        std::string& setDescription();
        void setId(size_t id);
        void setMorePeopleInvolved(bool morePeople);
        void setRoles(std::vector<std::string>& roles);
        std::string& setTitle();
        std::string str() const;
    private:
        bool containsDate_;
        bool containsPlace_;
        std::string description_;
        size_t id_;
        bool morePeopleInvolved_;
        std::vector<std::string> roles_;
        std::string title_;
};

std::ostream& operator«(std::ostream& os, const EventTemplate& et);

enum Trait {None, Fatherhood, Motherhood, Partnership, Sibling};

namespace trait{
    constexpr Trait AllTraits[] = {None, Fatherhood, Motherhood, Partnership, Sibling};
    std::string str(Trait t);
}

class RelationTemplate{
    public:
        RelationTemplate();
        const std::string& getDescription() const;
        const std::string& getFirstName() const;
        int getGenerationDifference() const;
        size_t getId() const;
        const std::string& getSecondName() const;
        const std::string& getTitle() const;
        Trait getTrait() const;
        void readJson(const Json::Value& value);
        std::string& setDescription();
        std::string& setFirstName();
        void setGenerationDifference(int diff);
        void setId(size_t id);
        std::string& setSecondName();
        std::string& setTitle();
        void setTrait(Trait trait);
        std::string str() const;
    private:
        std::string description_;
        std::string firstName_;
        int generationDiff_;
        size_t id_;
        std::string secondName_;
        std::string title_;
        Trait trait_;
};

std::ostream& operator«(std::ostream& os, const RelationTemplate& rt);

enum TabPosition {North, South, East, West};

struct Color{
    explicit Color(int r = 255, int g = 255, int b = 255);
    int red;
    int green;
    int blue;
    void readJson(const Json::Value& value);
};

std::ostream& operator«(std::ostream& os, const Color& color);

struct AppSettings{
    std::string fontFamily;
    int fontSize;
    int genSizeDown;
    int genSizeUp;
    Color highlightedColor;
    Color lineColor;
    int lineWidth;
    Color probandColor;
    Color promoted;
    int radius;
    void readJson(const Json::Value& value);
    void setDefault();
    bool showEventTab;
    bool showFileTab;
    bool showGeneralTab;
    bool showMediaTab;
    bool showNoteTab;
    bool showRelationTab;
    int sizeX;
    int sizeY;
    int splitterPositionOne;
    int splitterPositionTwo;
    Color standard;
    TabPosition tabPosition;
    Color textColor;
};

std::ostream& operator«(std::ostream& os, const AppSettings& as);

class Settings{
    public:
        Settings();
        void addDefaultEvents();
        void addDefaultRelations();
        size_t addEventTemplate(EventTemplate& et);
        size_t addRelationTemplate(RelationTemplate& rt);
        void addTag(const std::string& tag);
        void clear();
        void exportTemplates(std::ostream& os, const std::vector<size_t>& relIds, const
std::vector<size_t>& eventIds);
        const AppSettings& getAppSettings() const;
        std::optional<EventTemplate*> getEventTemplate(size_t id) const;
        const std::map<size_t, std::unique_ptr<EventTemplate»& getEventTemplates() const;
        size_t getGlobalMainPerson() const;
        std::optional<RelationTemplate*> getRelationTemplate(size_t id) const;
        const std::map<size_t, std::unique_ptr<RelationTemplate»& getRelationTemplates() const;
        const std::vector<std::string>& getTags() const;
        size_t importTemplates(const Json::Value& value);
        std::pair<size_t, size_t> indexes();
        void readJson(const Json::Value& value);
        void reIndex(size_t plusEvent, size_t plusRel);
        void removeEventTemplate(size_t index);
        void removeRelationTemplate(size_t index);
        void removeTag(const std::string& tag);
        AppSettings& setAppSettings();
        void setGlobalMainPerson(size_t id);
    private:
        void addDefaultEvent(const std::string& title, const std::string& description, bool date, bool place, bool morePeople, std::vector<std::string> roles);
        void addDefaultRelation(const std::string& title, const std::string& description, const std::string& firstName, const std::string& secondName, int genDiff, Trait trait);
        AppSettings appSettings_;
        size_t eventTemplateIndex_;
        std::map<size_t, std::unique_ptr<EventTemplate» eventTemplates_;
        size_t globalMainPerson_;
        size_t relationTemplateIndex_;
        std::map<size_t, std::unique_ptr<RelationTemplate» relationsTemplates_;
        std::vector<std::string> tags_;
};

std::ostream& operator «(std::ostream& os, const Settings& setting);

#endif
```

15.4 core/date.cpp File Reference

Source file for date formats.

```
#include ¨date.h¨
```

Functions

- std::ostream & operator<< (std::ostream &os, const Date &d)
- std::ostream & operator<< (std::ostream &os, const WrappedDate &wd)

15.4.1 Detailed Description

Source file for date formats.

15.4.2 Function Documentation

15.4.2.1 operator<<() [1/2]

```
std::ostream & operator<< (
            std::ostream & os,
            const Date & d )
```

Writing date to the stream in format yyyy-mm-dd. Instead of writing for example 05, there will be just 5.

Parameters

os	Given stream.
d	Given date.

Returns

Reference to the changed stream.

15.4.2.2 operator<<() [2/2]

```
std::ostream & operator<< (
            std::ostream & os,
            const WrappedDate & wd )
```

Writing wrapped date to given stream in JSON formatting.

Parameters

os	Given stream.
wd	Given WrappedDate.

Returns

Reference to the changed stream.

15.5 core/date.h File Reference

Header file for date formats.

```
#include <string>
#include <ostream>
#include <sstream>
#include <json/json.h>
#include <ctime>
#include ¨strings.h¨
```

Classes

- class Date

  Basic date format. Holding three numbers for year, month and day. If some of the numbers are not defined 0 is present.
- class WrappedDate

  Class holding two dates and date description. Always first date is the main one.

Functions

- std::ostream & operator<< (std::ostream &os, const Date &d)
- std::ostream & operator<< (std::ostream &os, const WrappedDate &wd)

15.5.1 Detailed Description

Header file for date formats.

15.5.2 Function Documentation

15.5.2.1 operator<<() [1/2]

```
std::ostream & operator<< (
            std::ostream & os,
            const Date & d )
```

Writing date to the stream in format yyyy-mm-dd. Instead of writing for example 05, there will be just 5.

Parameters

os	Given stream.
d	Given date.

Returns

Reference to the changed stream.

15.5.2.2 operator<<() [2/2]

```
std::ostream & operator<< (
            std::ostream & os,
            const WrappedDate & wd )
```

Writing wrapped date to given stream in JSON formatting.

Parameters

os	Given stream.
wd	Given WrappedDate.

Returns

Reference to the changed stream.

15.6 date.h

Go to the documentation of this file.

```

#ifndef date_h_
#define date_h_

#include <string>
#include <ostream>
#include <sstream>
#include <json/json.h>
#include <ctime>
#include "strings.h"

class Date{
    public:
        explicit Date(int y = 0, int m = 0, int d = 0);
        int getDay() const;
        int getMonth() const;
        int getYear() const;
        bool isEmpty() const;
        bool isExact() const;
        void read(const std::string& date);
        void setDay(int d);
        void setMonth(int m);
        void setYear(int y);
        std::string str() const;
    private:
        int day;
        int month;
        int year;
};

std::ostream& operator«(std::ostream& os, const Date& d);

class WrappedDate{
    public:
        WrappedDate();
        void clear();
        const Date& getFirstDate() const;
        const Date& getSecondDate() const;
        const std::string& getText() const;
        bool isExact() const;
        bool isUnknown() const;
        void readJson(const Json::Value& value);
        void setFirstDate(int y, int m, int d);
        void setSecondDate(int y, int m, int d);
        std::string& setText();
        std::string str() const;
        void updateDate();
    private:
        Date date1_;
        Date date2_;
        std::string text_;
};

std::ostream& operator«(std::ostream& os, const WrappedDate& wd);

#endif
```

15.7 core/family_tree.cpp File Reference

Source file for Family tree.

```
#include ¨family_tree.h¨
```

15.7.1 Detailed Description

Source file for Family tree.

15.8 core/family_tree.h File Reference

Header file for Family tree.

```
#include <string>
#include <sstream>
#include <vector>
#include <ostream>
#include <map>
#include <set>
#include <memory>
```

```
#include <functional>
#include <json/json.h>
#include ¨file_parser.h¨
#include ¨config.h¨
#include ¨date.h¨
#include ¨person.h¨
#include ¨strings.h¨
#include ¨family_tree_items.h¨
```

Classes

- class FamilyTree

  Main class for holding all data of a family tree. Also it acts as a bridge to the core of the application.

15.8.1 Detailed Description

Header file for Family tree.

15.9 family_tree.h

Go to the documentation of this file.

```

#ifndef family_tree_
#define family_tree_

#include <string>
#include <sstream>
#include <vector>
#include <ostream>
#include <map>
#include <set>
#include <memory>
#include <functional>
#include <json/json.h>
#include "file_parser.h"
#include "config.h"
#include "date.h"
#include "person.h"
#include "strings.h"
#include "family_tree_items.h"

class FamilyTree{
    public:
        FamilyTree();
        Event* addEvent();
        Event* addEvent(size_t index);
        File* addFile(const std::string& name, FileType type = GENERAL_FILE);
        File* addFile(const std::string& name, size_t index, FileType type = GENERAL_FILE);
        Person* addPerson();
        Person* addPerson(size_t index);
        Relation* addRelation();
        Relation* addRelation(size_t index);
        void append(FamilyTree& other);
        std::pair<bool, std::string> checkFileConsistence();
        void clear();
        void clearProjectPaths();
        size_t copyFile(const std::string& filePath, FileType type = GENERAL_FILE);
        void createDatabase(const std::string& dirPath);
        size_t createEmptyFile(const std::string& name, FileType type = GENERAL_FILE);
        void enableHelpOnStartup(bool help);
        void exportProject(const std::string& filePath, bool includingEvents, bool
includingRelations);
        std::string createResourcesDir(const std::string& target);
        void exportTemplates(const std::vector<size_t>& eventTemplates, const std::vector<size_t>&
relTemplates, const std::string& filename);
        std::optional<Event*> getEvent(size_t id);
        const std::map<size_t, std::unique_ptr<Event»& getEvents() const;
        std::optional<File*> getFile(size_t id, FileType type = GENERAL_FILE);
        void getFileAbsolutePath(std::string& absolutePath, size_t id, FileType type = GENERAL_FILE);
        const std::map<size_t, std::unique_ptr<File»& getFiles(FileType type = GENERAL_FILE);
        std::vector<std::string> getOrphanFiles();
        Person* getMainPerson();
        const std::map<size_t, std::unique_ptr<Person»& getPersons();
        std::optional<Person*> getPerson(size_t id);
        const std::vector<std::string>& getProjectPaths() const;
        std::optional<Relation*> getRelation(size_t id);
        const std::map<size_t, std::unique_ptr<Relation»& getRelations();
        std::vector<RelationSuggestion> getRelationSuggestions(size_t relId);
        Settings* getSettings();
        void getUrlFile(size_t id, std::string& fileUrl, FileType type = GENERAL_FILE);
        bool importProject(const std::string& filePath);
        bool importRootDir(const std::string& dirPath);
        std::pair<bool, size_t> importTemplates(const std::string& filePath);
        bool isDirectorySet();
        bool isSaved() const;
        void log(const std::string& error);
        std::pair<bool, bool> openDatabase(const std::string& dirPath, std::string& errorMessage, std::string& backupFile);
        bool openDatabase(std::string& errorMessage);
        bool openSingleFileDatabase(const std::string& filePath);
        void printHtml(const std::string& dirPath);
        void printMainPersonHtml(const std::string& outputFile);
        void promoteRelation(size_t id);
        void removeBackup();
        void removeEvent(size_t id);
        void removeEventTemplate(size_t id);
        size_t removeOrphanFiles();
        void removePerson();
        void removeRelation(size_t id);
        void removeRelationTemplate(size_t id);
        void renameFile(size_t id, const std::string& newFilename, FileType type);
        bool restoreBackup(const std::string& backupFile, std::string& errorMessage);
        void setMainPerson(Person* p);
        void setMainPerson(size_t id);
        void setRelation(size_t relId, size_t pers1Id, size_t pers2Id, size_t templId);
        void setUnsaved();
        bool showHelpOnStartup();
        bool storeDatabase();
        size_t templatesBasedOnEventTemplate(size_t templateId);
        size_t templatesBasedOnRelationTemplate(size_t templateId);
        void updateEventsWithTemplate(size_t templ);
        bool writeConfig(const std::string& newPath);
    private:
        std::map<size_t, std::unique_ptr<Event» allEvents_;
        std::map<size_t, std::unique_ptr<File» allFiles_;
        std::map<size_t, std::unique_ptr<File» allMedia_;
        std::map<size_t, std::unique_ptr<File» allNotes_;
        std::map<size_t, std::unique_ptr<Person» allPersons_;
        std::map<size_t, std::unique_ptr<Relation» allRelations_;
        bool checkFileTypeConsistence(std::ostream& os, FileType type);
        size_t event_index_;
        size_t file_index_;
        void getGeneralOrphaFiles(std::vector<std::string>& files, const std::map<size_t, std::unique_ptr<File»& container, FileType type);
        void getParentSuggestions(std::vector<RelationSuggestion>& suggestions, const Person* parent, const Person* child, size_t id, Trait trait,
                            const std::set<size_t>& forbiddenPersonParent, const std::set<size_t>& forbiddenPersonChild);
        void getPartnersSuggestions(std::vector<RelationSuggestion>& suggestions, const Person* person, const Person* second, size_t id, const std::set<size_t>& forbiddenPersons);
        void getSiblingsSuggestions(std::vector<RelationSuggestion>& suggestions, const Person* person, const Person* second, size_t id, const std::set<size_t>& forbiddenPersons);
        bool loadDatabaseFile(const std::string& file, std::function<void(const Json::Value&)> reader, std::string& errorMessage);
        Person* mainPerson_;
        size_t media_index_;
        size_t note_index_;
        bool openHelp;
        Parser parser_;
        size_t person_index_;
        void printCss(const std::string& dirPath);
        void printFiles(std::ostream& os, const std::map<size_t, std::unique_ptr<File»& container, const std::string& label);
        void printEvents(std::ostream& os);
        void printHtmlPerson(std::ostream& os, bool links, Person* p);
        void printPeople(std::ostream& os);
        void printRelations(std::ostream& os);
        std::vector<std::string> projectPaths;
        void readJsonEvents(const Json::Value& value);
        void readJsonFiles(const Json::Value& value, std::map<size_t, std::unique_ptr<File»& container, const std::string& label, size_t& index);
        void readJsonPeople(const Json::Value& value);
        void readJsonRelations(const Json::Value& value);
        size_t relation_index_;
        size_t removeGeneralOrphanFile(std::map<size_t, std::unique_ptr<File»& container, FileType type, const std::string& dir);
        bool savedChanges_;
        Settings settings_;
        void storeFileDatabase(FileType type);
};

#endif
```

15.10 core/family_tree_items.cpp File Reference

Source file for classes closely related to family tree.

`#include ¨family_tree_items.h¨`

Functions

- std::ostream & operator<< (std::ostream &os, const Event &e)
- std::ostream & operator<< (std::ostream &os, const Relation &r)
- std::ostream & operator<< (std::ostream &os, const File &f)

15.10.1 Detailed Description

Source file for classes closely related to family tree.

15.10.2 Function Documentation

15.10.2.1 operator<<() [1/3]

```
std::ostream & operator<< (
            std::ostream & os,
            const Event & e )
```

Using << operator on event to print event in JSON formatting.

Parameters

os	Given output stream.
e	Given event.

Returns

Changed output stream.

15.10.2.2 operator<<() [2/3]

```
std::ostream & operator<< (
            std::ostream & os,
            const File & f )
```

Output file to JSON format.

Parameters

os	Given stream.
f	Given file.

Returns

Changed stream.

15.10.2.3 operator<<() [3/3]

```
std::ostream & operator<< (
            std::ostream & os,
            const Relation & r )
```

Output the relation in JSON format.

Parameters

os	Given output stream.
r	Given relation.

Returns

Changed output stream.

15.11 core/family_tree_items.h File Reference

Header file for classes closely related to family tree such as Event, Relation and File.

```
#include <string>
#include <sstream>
#include <vector>
#include <ostream>
#include <json/json.h>
#include ¨config.h¨
#include ¨date.h¨
#include ¨strings.h¨
```

Classes

- class Event

  Class holding data for a single event.
- class Relation

  Class holding data for a relationship.
- struct RelationSuggestion

  Struct for relation suggestions.
- class File

  Meta-data about one file (media or note). Its real path is [id]-[filename].

Functions

- std::ostream & operator<< (std::ostream &os, const Event &e)
- std::ostream & operator<< (std::ostream &os, const Relation &r)
- std::ostream & operator<< (std::ostream &os, const File &f)

15.11.1 Detailed Description

Header file for classes closely related to family tree such as Event, Relation and File.

15.11.2 Function Documentation

15.11.2.1 operator<<() [1/3]

```
std::ostream & operator<< (
             std::ostream & os,
             const Event & e )
```

Using << operator on event to print event in JSON formatting.

Parameters

os	Given output stream.
e	Given event.

Returns

Changed output stream.

15.11.2.2 operator<<() [2/3]

```
std::ostream & operator<< (
             std::ostream & os,
             const File & f )
```

Output file to JSON format.

Parameters

os	Given stream.
f	Given file.

Returns

Changed stream.

15.11.2.3 operator<<() [3/3]

```
std::ostream & operator<< (
            std::ostream & os,
            const Relation & r )
```

Output the relation in JSON format.

Parameters

os	Given output stream.
r	Given relation.

Returns

Changed output stream.

15.12 family_tree_items.h

Go to the documentation of this file.

```

#ifndef family_tree_items_
#define family_tree_items_

#include <string>
#include <sstream>
#include <vector>
#include <ostream>
#include <json/json.h>
#include "config.h"
#include "date.h"
#include "strings.h"

class Event{
    public:
        explicit Event(Settings* settings);
        void addPerson(size_t person, const std::string& role = events::NO_ROLE);
        void clearPersons();
        WrappedDate* date();
        const WrappedDate& getDate() const;
        size_t getId() const;
        const std::vector<std::pair<std::string, size_t»& getPersons() const;
        const std::string& getPlace() const;
        const Settings& getSettings() const;
        size_t getTemplate() const;
        const std::string& getText() const;
        void printHtml(std::ostream& os, const std::vector<std::pair<std::string, std::string»& persons);
        void readJson(const Json::Value& value);
        bool removePerson(size_t id);
        WrappedDate& setDate();
        void setId(size_t id);
        void setPlace(const std::string& place);
        void setTemplate(size_t templ);
        void setText(const std::string& text);
        std::string str() const;
        std::vector<size_t> updateToTemplate();
    private:
        WrappedDate date_;
        size_t id_;
        std::vector<std::pair<std::string, size_t» persons_;
        std::string place_;
        Settings* settings_;
        size_t template_;
        std::string text_;

};

std::ostream& operator «(std::ostream& os, const Event& e);

class Relation{
    public:
        explicit Relation(Settings* settings);
        size_t getFirstPerson() const;
        size_t getId() const;
        size_t getSecondPerson() const;
        size_t getTemplate() const;
        const std::string& getTitle() const;
        size_t getTheOtherPerson(size_t current);
        const std::string& getTheOtherPersonName(size_t current);
        Trait getTrait();
        bool isOlderGeneration(size_t id);
        bool isParent(size_t personId);
        bool isSameGenration();
        bool isYoungerGeneration(size_t id);
        void printHtml(std::ostream& os, const std::string& firstPerson, const std::string& secondPerson, const std::string& promoted);
        void readJson(const Json::Value& value);
        void setId(size_t id);
        void setPersons(size_t person1, size_t person2);
        void setTemplate(size_t templ);
        std::string str(const std::string& firstPerson, const std::string& secondPerson) const;
    private:
        size_t id_;
        size_t person1_;
        size_t person2_;
        Settings* settings_;
        size_t template_;
};

std::ostream& operator«(std::ostream& os, const Relation& r);

struct RelationSuggestion{
    public:
        RelationSuggestion(size_t lockedP, size_t suggestedP, bool par, bool sym, Trait t);
        size_t lockedPerson;
        bool parent;
        size_t suggestedPerson;
        bool symetrical;
        Trait trait;
};

class File{
    public:
        File();
        const std::string& getFilename() const;
        size_t getId() const;
        std::string getRealName() const;
        void readJson(const Json::Value& value);
        void setFilename(const std::string& fileName);
        void setId(size_t id);
    private:
        std::string filename_;
        size_t id_;
};

std::ostream& operator «(std::ostream& os, const File& f);

#endif
```

15.13 core/file_parser.cpp File Reference

Source file for working with predefined database.

```
#include ¨file_parser.h¨
```

15.13.1 Detailed Description

Source file for working with predefined database.

15.14 core/file_parser.h File Reference

The header file for working with the database and other files.

```
#include <filesystem>
#include <iostream>
#include <fstream>
#include <string>
#include <sstream>
#include <json/json.h>
#include <chrono>
#include <iomanip>
#include <ctime>
#include <set>
#include ¨strings.h¨
```

Classes

- class Parser

  Main class for working with the database.

Namespaces

- namespace parser

  Namespace for all strings representing files and directories in the database.

Variables

- const std::string parser::FILES_DIR = ¨Files¨

  Directory for all documents and files in general.
- const std::string parser::JSON_CONFIG = ¨Config.json¨

  JSON file-name for configuration and templates.
- const std::string parser::JSON_EVENTS = ¨Events.json¨

  JSON file-name for events.
- const std::string parser::JSON_FILES = ¨Files.json¨

  JSON file-name for files meta-data.
- const std::string parser::JSON_MEDIA = ¨Media.json¨

  JSON file-name for media meta-data.
- const std::string parser::JSON_NOTES = ¨Notes.json¨

  JSON file-name for notes meta-data.
- const std::string parser::JSON_PERSONS = ¨Persons.json¨

  JSON file-name for persons.
- const std::string parser::JSON_RELATIONS = ¨Relations.json¨

  JSON file-name for relations.
- const std::string parser::MEDIA_DIR = ¨Media¨

  Directory for all pictures and other media.
- const std::string parser::NOTES_DIR = ¨Notes¨

  Directory for all notes.
- const std::string parser::ERROR_LOG = ¨.error.log¨

  File for writing error log.

15.14.1 Detailed Description

The header file for working with the database and other files.

15.15 file_parser.h

Go to the documentation of this file.

```

#ifndef file_parser_h_
#define file_parser_h_

#include <filesystem>
#include <iostream>
#include <fstream>
#include <string>
#include <sstream>
#include <json/json.h>
#include <chrono>
#include <iomanip>
#include <ctime>
#include <set>
#include "strings.h"

namespace parser{
    const std::string FILES_DIR = "Files";
    const std::string JSON_CONFIG = "Config.json";
    const std::string JSON_EVENTS = "Events.json";
    const std::string JSON_FILES = "Files.json";
    const std::string JSON_MEDIA = "Media.json";
    const std::string JSON_NOTES = "Notes.json";
    const std::string JSON_PERSONS = "Persons.json";
    const std::string JSON_RELATIONS = "Relations.json";
    const std::string MEDIA_DIR = "Media";
    const std::string NOTES_DIR = "Notes";
    const std::string ERROR_LOG = ".error.log";
}

class Parser{
    public:
        Parser();
        void checkFileConsistency(std::set<std::string>& database, std::vector<std::string>& found, const std::string& dir);
        void clear();
        bool containsBackupFile(std::string& backupFile);
        bool copyFile(const std::string& filePath, const std::string& dir, std::string& fileName, size_t id);
        void createEmptyFile(const std::string& dir, const std::string& fileName);
        std::string createResourcesDir(const std::string& target);
        std::string getAbsoluteFilePath(const std::string& dir, const std::string& fileName) const;
        void getFileUrl(const std::string& dir, const std::string& fileName, std::string& fileUrl) const;
        bool isRootDirectorySet();
        void log(const std::string&  error);
        void makeNewDatabase(const std::string& dirPath);
        std::pair<bool, std::vector<std::string» readConfig();
        bool readJSONFile(const std::string& fileName, Json::Value& root, bool inDatabase = true);
        void removeBackup();
        void removeFile(const std::string& filename, const std::string& dir);
        void renameFile(const std::string& originalName, const std::string& changedName, const std::string& dir);
        void restoreBackup(const std::string& backupFile);
        bool setDatabase(const std::string& dirPath);
        void writeConfig(bool help);
        void writeConfig(const std::vector<std::string>& rootPaths, bool help);
        void writeHtml(const std::string& html, const std::string& output);
        void writeHtml(const std::string& html, const std::string& dirPath, const std::string& filename);
        void writeJSON(const std::string& data, const std::string& file, bool inDatabase = true);
    private:
        bool containsBackup(std::string& backupFile, const std::string& filename);
        void createAllDirs();
        void createJsonFile(const std::string& fileName);
        std::filesystem::path root_;
        std::filesystem::path configPath_;
};

#endif
```

15.16 core/person.cpp File Reference

Source file for all classes closely related to person.

```
#include "person.h"
```

Functions

- std::ostream & operator<< (std::ostream &os, const VirtualDrive &vd)
- void printSizeTContainer (std::ostream &os, const std::vector< size_t > &container, const std::string &label)
- std::ostream & operator<< (std::ostream &os, const Person &p)

15.16.1 Detailed Description

Source file for all classes closely related to person.

15.16.2 Function Documentation

15.16.2.1 operator<<() [1/2]

```
std::ostream & operator<< (
            std::ostream & os,
            const Person & p )
```

Write the data about the person in JSON format.

Parameters

os	Is the given stream.
p	Is the given person.

Returns

Reference to the used output stream.

15.16.2.2 operator<<() [2/2]

```
std::ostream & operator<< (
            std::ostream & os,
            const VirtualDrive & vd )
```

Print virtual drive to JSON format.

Parameters

os	Given output stream.
vd	Given VirtualDrive.

Returns

Changed output stream.

15.16.2.3 printSizeTContainer()

```
void printSizeTContainer (
            std::ostream & os,
            const std::vector< size_t > & container,
            const std::string & label )
```

Print to JSON formatting container of unsigned integers with given label.

Parameters

os	Given stream where to print.
container	Given container of ids.
label	The label of JSON array.

15.17 core/person.h File Reference

Header file for all classes closely related to person.

```
#include <string>
#include <vector>
#include <ostream>
#include <sstream>
#include <json/json.h>
#include ¨date.h¨
#include ¨strings.h¨
#include ¨config.h¨
```

Classes

- class VirtualDrive

  Class for virtualization of folder structure for each person.
- class Person

  One person in the family tree.

Namespaces

- namespace gender

  Namespace for constants with gender enumeration.

Enumerations

- enum Gender { Other , Male , Female }

  Gender types: Male, Female and Other.
- enum FileType { GENERAL_FILE , MEDIA , NOTE }

  Enumeration on special types of files.

Functions

- std::ostream & operator<< (std::ostream &os, const VirtualDrive &vd)
- std::string gender::str (Gender g)
- void printSizeTContainer (std::ostream &os, const std::vector< size_t > &container, const std::string &label)
- std::ostream & operator<< (std::ostream &os, const Person &p)

Variables

- constexpr Gender gender::AllGenders [ ] = {Other, Male, Female}

  All genders in a list.

15.17.1 Detailed Description

Header file for all classes closely related to person.

15.17.2 Enumeration Type Documentation

15.17.2.1 FileType

```
enum FileType
```

Enumeration on special types of files.

Enumerator

GENERAL_FILE	
MEDIA	
NOTE	

15.17.2.2 Gender

```
enum Gender
```

Gender types: Male, Female and Other.

Enumerator

Other	
Male	
Female	

15.17.3 Function Documentation

15.17.3.1 operator<<() [1/2]

```
std::ostream & operator<< (
            std::ostream & os,
            const Person & p )
```

Write the data about the person in JSON format.

Parameters

os	Is the given stream.
p	Is the given person.

Returns

Reference to the used output stream.

15.17.3.2 operator<<() [2/2]

```
std::ostream & operator<< (
            std::ostream & os,
            const VirtualDrive & vd )
```

Print virtual drive to JSON format.

Parameters

os	Given output stream.
vd	Given VirtualDrive.

Returns

Changed output stream.

15.17.3.3 printSizeTContainer()

```
void printSizeTContainer (
            std::ostream & os,
            const std::vector< size_t > & container,
            const std::string & label )
```

Print to JSON formatting container of unsigned integers with given label.

Parameters

os	Given stream where to print.
container	Given container of ids.
label	The label of JSON array.

15.18 person.h

Go to the documentation of this file.

```

#ifndef person_h_
#define person_h_

#include <string>
#include <vector>
#include <ostream>
#include <sstream>
#include <json/json.h>
#include "date.h"
#include "strings.h"
#include "config.h"

class VirtualDrive{
    public:
        VirtualDrive() = default;
        explicit VirtualDrive(const std::string& name);
        void addFile(size_t fileId);
        VirtualDrive* addSubdrive();
        VirtualDrive* addSubdrive(const std::string& name);
        void copyDrive(const VirtualDrive& other, size_t plusIndex = 0);
        const std::vector<size_t>& getFiles() const;
        const std::string& getName() const;
        const std::vector<std::unique_ptr<VirtualDrive»& getSubdrives() const;
        void readJson(const Json::Value& value);
        void remove(const std::string& subfolderName);
        void removeFile(size_t id);
        void rename(const std::string& name);
        void usedFiles(std::vector<size_t>& used) const;
    private:
        std::vector<size_t> files_;
        std::string name_;
        std::vector<std::unique_ptr<VirtualDrive» subdrives_;
};

std::ostream& operator«(std::ostream& os, const VirtualDrive& vd);

enum Gender {Other, Male, Female};

namespace gender{
    constexpr Gender AllGenders[] = {Other, Male, Female};
    std::string str(Gender g);
}

enum FileType{GENERAL_FILE, MEDIA, NOTE};

class Person{
    public:
        Person();
        void addEvent(size_t id);
        void addRelation(size_t id);
        void addTag(const std::string& tag, const std::string& value);
        WrappedDate* birthDate();
        WrappedDate* deathDate();
        std::pair<bool, const std::string&> existsTag(const std::string& tag) const;
        void exportJson(std::ostream& os, bool includingEvents, bool includingRelations);
        const std::string& getAfterTitle() const;
        const WrappedDate& getBirthDate() const;
        const std::string& getBirthPlace() const;
        const WrappedDate& getDeathDate() const;
        const std::string& getDeathPlace() const;
        const std::vector<size_t>& getEvents() const;
        size_t getFather() const;
        std::string getFullName() const;
        const VirtualDrive& getFilesRoot(FileType type = GENERAL_FILE) const;
        VirtualDrive* getFilesRootPointer(FileType type);
        const std::string& getFrontTitle() const;
        Gender getGender() const;
        size_t getId() const;
        const std::string& getMaidenName() const;
        size_t getMother() const;
        const std::string& getName() const;
        size_t getPartner() const;
        const std::vector<size_t>& getRelations() const;
        const std::string& getSurname() const;
        const std::vector<std::pair<std::string, std::string»& getTags() const;
        void importJson(const Json::Value& value);
        bool isAlive() const;
        void printHtml(std::ostream& os);
        void promoteRelation(size_t id, Trait type);
        void readJson(const Json::Value& value);
        void removeEvent(size_t id);
        void removeRelation(size_t id);
        void removeTag(const std::string& tag);
        void setAfterTitle(const std::string& title);
        void setBirthDate(int y, int m, int d);
        std::string& setBirthPlace();
        void setDeathDate(int y, int m, int d);
        std::string& setDeathPlace();
        void setFather(size_t father);
        void setFilesRoot(const VirtualDrive& drive, FileType type = GENERAL_FILE, size_t plusIndex = 0);
        void setFrontTitle(const std::string& title);
        void setGender(Gender g);
        void setId(size_t id);
        void setLives(bool lives);
        std::string& setMaidenName();
        void setMother(size_t mother);
        std::string& setName();
        void setPartner(size_t id);
        std::string& setSurname();
        std::string str() const;
        void updateSpecialRelation(size_t relId, Trait trait);
    private:
        WrappedDate dateOfBirth_;
        WrappedDate dateOfDeath_;
        std::vector<size_t> events_;
        size_t father_;
        Gender gender_;
        size_t id_;
        bool lives_;
        std::string maidenName_;
        size_t mother_;
        std::string name_;
        size_t partner_;
        std::string placeOfBirth_;
        std::string placeOfDeath_;
        std::vector<size_t> relations_;
        VirtualDrive rootFileDrive_;
        VirtualDrive rootMediaDrive_;
        VirtualDrive rootNoteDrive_;
        std::string surname_;
        std::vector<std::pair<std::string, std::string» tags_;
        std::string titleAfter_;
        std::string titleInFront_;
};

void printSizeTContainer(std::ostream& os, const std::vector<size_t>& container, const std::string& label);

std::ostream& operator«(std::ostream& os, const Person& p);

#endif
```

15.19 core/strings.h File Reference

Is file containing all English strings.

```
#include <string>
```

Namespaces

- namespace person

  Namespace for strings related to a person.
- namespace tagchooser

  Namespace for strings shown in tag chooser dialog.
- namespace addfile

  Namespace for strings shown in add file dialog.
- namespace button

  Namespace for general texts on buttons.
- namespace error

Namespace for errors and exceptions messages.

- namespace relation

  Namespace for strings for default relations.
- namespace events

  Namespace for strings for default events.
- namespace html

  Namespace for html tags and everything with html.
- namespace jsonlabel

  Namespace for all JSON labels.

Variables

- const std::string EMPTY_STRING = ¨¨

  Empty string.
- const std::string person::UNKNOWN_STR = ¨Unknown¨

  String for unknown tags.
- const std::string person::SURNAME_STR = ¨Surname¨

  String for default surname.
- const std::string person::NAME_STR = ¨Name¨

  String for default name.
- const std::string person::ROOT = ¨root¨

  String for default root directories.
- const std::string person::DATE_FROM = ¨From: ¨

  String for date interval from.
- const std::string person::DATE_TO = ¨ to: ¨

  String for date interval to.
- const std::string tagchooser::CUSTOM_TAG = ¨+ Custom tag¨

  Last custom tag choice.
- const std::string addfile::NONE = ¨None¨

  String for no extension.
- const std::string button::ADD_NEW = ¨Add new¨

  Text for adding new element.
- const std::string button::CANCEL = ¨Cancel¨

  Text for cancelling.
- const std::string button::REMOVE = ¨Remove¨

  text for removing element.
- const std::string error::UNKNOWN_EXCEP_STR = ¨Unknown exception catch.¨

  String for catching unknown exception.
- const std::string error::EXCEPTION_STR = ¨Exception: ¨

  String for adding before exception description.
- const std::string error::CORRUPTED_DATABASE = ¨Given database is corrupted: ¨

  String shown when database is corrupted.
- const std::string error::EMPTY_BOXES = ¨All boxes must not be empty.¨

  String for showing that some boxes were left empty.
- const std::string error::NO_CHOSEN_FILE = ¨No file was chosen.¨

  No file chosen and empty box.
- const std::string error::NO_ROLE = ¨Given role 'no role' cannot be used. It would be considered as no special role.¨

  When person tries to add ¨no role¨ role, which is allocated for no roles.
- const std::string error::FORBIDDEN_CHARS = ¨Text cannot contain forbidden characters. &lt; &gt; \¨¨

When in any box where inserted forbidden characters.

- const std::string relation::FATHERHOOD = ¨Fatherhood¨

  String for Fatherhood.
- const std::string relation::FATHERHOOD_DESC = ¨Relation between father and his child.¨

  String for Fatherhood description.
- const std::string relation::FATHER = ¨Father¨

  String for father.
- const std::string relation::CHILD = ¨Child¨

  String for child.
- const std::string relation::MOTHERHOOD = ¨Motherhood¨

  String for Motherhood.
- const std::string relation::MOTHERHOOD_DESC = ¨Relation between mother and her child.¨

  String for Motherhood description.
- const std::string relation::MOTHER = ¨Mother¨

  String for mother.
- const std::string relation::PARTNERSHIP = ¨Partnership¨

  String for Partnership.
- const std::string relation::PARTNERSHIP_DESC = ¨Relation between two partners.¨

  String for description of partnership.
- const std::string relation::PARTNER = ¨Partner¨

  String for partner.
- const std::string relation::MARRIAGE = ¨Marriage¨

  String for Marriage.
- const std::string relation::MARRIAGE_DESC = ¨Bound by law between two people. Usually a man (husband) and a woman (wife).¨

  String for Marriage description.
- const std::string relation::WIFE = ¨Wife¨

  String for wife.
- const std::string relation::HUSBAND = ¨Husband¨

  String for husband.
- const std::string relation::ADOPTIVE_FATHERHOOD = ¨Adoptive fatherhood¨

  String for title for adoptive fatherhood.
- const std::string relation::DESC_ADOPTIVE_FATHERHOOD = ¨Fatherhood provided by adopting a child.¨

  String for description for adoptive fatherhood.
- const std::string relation::ADOPTIVE_FATHER = ¨Adoptive father¨

  String for adoptive father.
- const std::string relation::ADOPTED_CHILD = ¨Adopted child¨

  String for adopted child.
- const std::string relation::ADOPTIVE_MOTHERHOOD = ¨Adoptive motherhood¨

  String for title for adoptive motherhood.
- const std::string relation::DESC_ADOPTIVE_MOTHERHOOD = ¨Motherhood provided by adopting a child.¨

  String for description for adoptive motherhood.
- const std::string relation::ADOPTIVE_MOTHER = ¨Adoptive mother¨

  String for adoptive father.
- const std::string relation::BROTHERHHOD = ¨Brotherhood¨

  String for brotherhood.
- const std::string relation::DESC_BROTHERHOOD = ¨Relation between two brothers.¨

  String for brotherhood description.
- const std::string relation::BROTHER = ¨Brother¨

  String for brother.
- const std::string relation::SISTERHOOD = ¨Sisterhood¨

String for sisterhood.

- const std::string relation::DESC_SISTERHOOD = ¨Relation between two sisters.¨

  String for sisterhood description.
- const std::string relation::SISTER = ¨Sister¨

  String for sister.
- const std::string relation::SIBLINGS = ¨Siblings¨

  String for siblings.
- const std::string relation::DESC_SIBLINGS = ¨Relation between brother and sister.¨

  String for siblings description.
- const std::string events::WEDDING = ¨Wedding¨

  String for wedding.
- const std::string events::DESC_WEDDING = ¨A social event mainly between soon to be wife and husband.¨

  String for description of wedding.
- const std::string events::GROOM = ¨Groom¨

  String for groom.
- const std::string events::BRIDE = ¨Bride¨

  String for bride.
- const std::string events::BEST_MAN = ¨Best man¨

  String for best man.
- const std::string events::MAID_OF_HONOR = ¨Maid of Honor¨

  String for maid of honor.
- const std::string events::DIVORCE = ¨Divorce¨

  String for divorce.
- const std::string events::DESC_DIVORCE = ¨Divorce is the process of terminating a marriage or marital union.¨

  String for divorce description.
- const std::string events::FORMER_HUSBAND = ¨Former husband¨

  String for former husband.
- const std::string events::FORMER_WIFE = ¨Former wife¨

  String for former wife.
- const std::string events::FUNERAL = ¨Funeral¨

  String for funeral.
- const std::string events::DESC_FUNERAL = ¨A funeral is a ceremony connected with the final disposition of a corpse, such as a burial or cremation, with the attendant observances.¨

  String for funeral description.
- const std::string events::DEAD_PERSON = ¨Dead person¨

  String for dead person.
- const std::string events::GRADUATION = ¨Graduation¨

  String for graduation.
- const std::string events::DESC_GRADUATION = ¨Graduation is the awarding of a diploma to a student by an educational institution.¨

  String for graduation description.
- const std::string events::GRADUATE = ¨Graduate¨

  String for graduate.
- const std::string events::NO_ROLE = ¨no role¨

  String for no role in event.
- const std::string html::RESOURCES = ¨resources¨

  Where will resources be saved.
- const std::string html::PREFIX_CLOSE = ¨/close-¨

  Prefix for close family diagram.
- const std::string html::PREFIX_FAMILY_TREE = ¨/tree-¨

Prefix for family tree diagram.

- const std::string html::CUSTOM_CSS = ¨styles.css¨

  Custom CSS file name.
- const std::string html::GENERAL_INFO = ¨General information¨

  General information.
- const std::string html::BIRTH_DATE = ¨Birth date: ¨

  Birth date.
- const std::string html::BIRTH_PLACE = ¨Birth place: ¨

  Birth place.
- const std::string html::DEATH_DATE = ¨Date of death: ¨

  Date of death.
- const std::string html::DEATH_PLACE = ¨Place of death: ¨

  Place of death.
- const std::string html::GENDER = ¨Gender: ¨

  Gender.
- const std::string html::CUSTOM_TAGS = ¨Custom tags:¨

  Custom tags.
- const std::string html::DESCRIPTION = ¨Description:¨

  Description.
- const std::string html::NOTE = ¨Note:¨

  Note.
- const std::string html::DATE = ¨Date: ¨

  Date.
- const std::string html::PLACE = ¨Place: ¨

  Place.
- const std::string html::PERSONS = ¨Persons:¨

  Persons.
- const std::string html::ALL_PERSONS = ¨All persons¨

  All persons.
- const std::string html::EVENTS = ¨Events¨

  Events.
- const std::string html::FAMILY_TREE = ¨Family tree¨

  Family tree.
- const std::string html::FAMILY_TREE_DIAGRAM = ¨Family tree diagram¨

  Family tree diagram.
- const std::string html::RELATIONS = ¨Relations¨

  Relations.
- const std::string html::CLOSE_FAMILY = ¨Close family¨

  Close family.
- const std::string html::CLOSE_FAMILY_DIAGRAM = ¨Close family diagram¨

  Close family diagram.
- const std::string jsonlabel::TITLE = ¨title¨

  JSON Label for title.
- const std::string jsonlabel::ID = ¨id¨

  JSON Label for id.
- const std::string jsonlabel::DESCRIPTION = ¨description¨

  JSON Label for description.
- const std::string jsonlabel::CONTAINS_DATE = ¨contains date¨

  JSON Label for if the event contains a date or not.
- const std::string jsonlabel::MORE_PEOPLE = ¨more people¨

  JSON Label for if the event contains more people.

- const std::string jsonlabel::SPECIAL_ROLES = ¨special roles¨

    JSON Label for if the event has special roles.
- const std::string jsonlabel::NAME = ¨name¨

    JSON Label for name.
- const std::string jsonlabel::FIRST_NAME = ¨first name¨

    JSON Label for first name.
- const std::string jsonlabel::SECOND_NAME = ¨second name¨

    JSON Label for second name.
- const std::string jsonlabel::TRAIT = ¨trait¨

    JSON Label for trait.
- const std::string jsonlabel::GEN_DIFF = ¨generation difference¨

    JSON Label for generation difference.
- const std::string jsonlabel::REL_TEMPL = ¨relation templates¨

    JSON Label for relations templates.
- const std::string jsonlabel::EVE_TEMPL = ¨event templates¨

    JSON Label for event templates.
- const std::string jsonlabel::TYPE = ¨type¨

    JSON Label for type.
- const std::string jsonlabel::DATE = ¨date¨

    JSON Label for date.
- const std::string jsonlabel::LAST_DATE = ¨last date¨

    JSON Label for last date.
- const std::string jsonlabel::TEXT = ¨text¨

    JSON Label for text.
- const std::string jsonlabel::TAGS = ¨tags¨

    JSON Label for tags.
- const std::string jsonlabel::TAG = ¨tag¨

    JSON Label for single tag.
- const std::string jsonlabel::RELATIONS = ¨relations¨

    JSON Label for relations.
- const std::string jsonlabel::EVENTS = ¨events¨

    JSON Label for events.
- const std::string jsonlabel::SURNAME = ¨surname¨

    JSON Label for surname.
- const std::string jsonlabel::MAIDEN_NAME = ¨maiden name¨

    JSON Label for maiden name.
- const std::string jsonlabel::VALUE = ¨value¨

    JSON Label for value.
- const std::string jsonlabel::BIRTH = ¨birth¨

    JSON Label for birth.
- const std::string jsonlabel::DEATH = ¨death¨

    JSON Label for death.
- const std::string jsonlabel::VIRTUAL_DRIVE = ¨virtual drive¨

    JSON Label for virtual drive.
- const std::string jsonlabel::MEDIA = ¨media¨

    JSON Label for media.
- const std::string jsonlabel::NOTES = ¨notes¨

    JSON Label for notes.
- const std::string jsonlabel::FILES = ¨files¨

    JSON Label for files.
- const std::string jsonlabel::PERSONS = ¨persons¨

JSON Label for persons.

- const std::string jsonlabel::TEMPLATE = ¨template¨

  JSON Label for for template.
- const std::string jsonlabel::TEMPLATES = ¨templates¨

  JSON Label for for templates.
- const std::string jsonlabel::FIRST_PERSON = ¨first person¨

  JSON Label for first person.
- const std::string jsonlabel::SECOND_PERSON = ¨second person¨

  JSON Label for second person.
- const std::string jsonlabel::PLACE = ¨place¨

  JSON Label for place.
- const std::string jsonlabel::ROLE = ¨role¨

  JSON Label for role.
- const std::string jsonlabel::PERSON = ¨person¨

  JSON Label for person.
- const std::string jsonlabel::GENDER = ¨gender¨

  JSON Label for gender.
- const std::string jsonlabel::LIVES = ¨lives¨

  JSON Label for if the person lives or not.
- const std::string jsonlabel::FRONT_TITLE = ¨title in front¨

  JSON Label for title in front of a name.
- const std::string jsonlabel::AFTER_TITLE = ¨title after¨

  JSON Label for title after the name.
- const std::string jsonlabel::FATHER = ¨father¨

  JSON Label for father.
- const std::string jsonlabel::MOTHER = ¨mother¨

  JSON Label for mother.
- const std::string jsonlabel::PARTNER = ¨partner¨

  JSON Label for partner.
- const std::string jsonlabel::MOVEABLE_TABS = ¨moveable tabs¨

  JSON Label for setting moveable tabs.
- const std::string jsonlabel::TAB_POSITION = ¨tab position¨

  JSON Label for tab position.
- const std::string jsonlabel::GLOBAL_MAIN_PERSON = ¨global main person¨

  JSON Label for global main person.
- const std::string jsonlabel::CONTAINS_PLACE = ¨contains place¨

  JSON Label if an event template contains a place.
- const std::string jsonlabel::SHOW_GENERAL_TAB = ¨show general tab¨

  JSON Label for app settings if general tab should be visible.
- const std::string jsonlabel::SHOW_EVENT_TAB = ¨show event tab¨

  JSON Label for app settings if event tab should be visible.
- const std::string jsonlabel::SHOW_FILE_TAB = ¨show file tab¨

  JSON Label for app settings if file tab should be visible.
- const std::string jsonlabel::SHOW_MEDIA_TAB = ¨show media tab¨

  JSON Label for app settings if media tab should be visible.
- const std::string jsonlabel::SHOW_NOTE_TAB = ¨show note tab¨

  JSON Label for app settings if note tab should be visible.
- const std::string jsonlabel::SHOW_RELATION_TAB = ¨show relation tab¨

  JSON Label for app settings if relation tab should be visible.
- const std::string jsonlabel::APP_SETTINGS = ¨app settings¨

  JSON Label for app settings.

- const std::string jsonlabel::DRIVES = ¨drives¨

  JSON Label for drives.
- const std::string jsonlabel::EXPORT = ¨export¨

  JSON Label for export.
- const std::string jsonlabel::RED = ¨red¨

  JSON Label for red.
- const std::string jsonlabel::BLUE = ¨blue¨

  JSON Label for blue.
- const std::string jsonlabel::GREEN = ¨green¨

  JSON Label for green.
- const std::string jsonlabel::FONT_FAMILY = ¨font family¨

  JSON Label for font family.
- const std::string jsonlabel::FONT_SIZE = ¨font size¨

  JSON Label for font size.
- const std::string jsonlabel::HIGHLIGHTED_COLOR = ¨highlighted color¨

  JSON Label highlighted color.
- const std::string jsonlabel::PROBANT_COLOR = ¨proband color¨

  JSON Label for proband color.
- const std::string jsonlabel::PROMOTED_COLOR = ¨promoted color¨

  JSON Label promoted color.
- const std::string jsonlabel::STANDARD_COLOR = ¨standard color¨

  JSON Label standard color.
- const std::string jsonlabel::X_SIZE = ¨x size¨

  JSON Label for size in the x axis.
- const std::string jsonlabel::Y_SIZE = ¨y size¨

  JSON Label for size in the y axis.
- const std::string jsonlabel::GEN_SIZE_UP = ¨number of generations up¨

  JSON Label for how many generations are being shown upwards.
- const std::string jsonlabel::GEN_SIZE_DOWN = ¨number of generations down¨

  JSON Label for how many generations are being shown downwards.
- const std::string jsonlabel::DIR = ¨dir¨

  JSON Label for directory.
- const std::string jsonlabel::ROOT = ¨root¨

  JSON Label for root.
- const std::string jsonlabel::HELP = ¨help¨

  JSON Label for help.
- const std::string jsonlabel::SPLITTER_POSITION_ONE = ¨splitter position one¨

  JSON Label for splitter position one.
- const std::string jsonlabel::SPLITTER_POSITION_TWO = ¨splitter position two¨

  JSON Label for splitter position two.
- const std::string jsonlabel::TEXT_COLOR = ¨text color¨

  JSON Label for the color of text.
- const std::string jsonlabel::LINE_COLOR = ¨line color¨

  JSON Label for the line color.
- const std::string jsonlabel::LINE_WIDTH = ¨line width¨

  JSON Label for line width.
- const std::string jsonlabel::BORDER_RADIUS = ¨radius¨

  JSON Label border radius.

15.19.1 Detailed Description

Is file containing all English strings.

15.19.2 Variable Documentation

15.19.2.1 EMPTY_STRING

```
const std::string EMPTY_STRING = ""
```

Empty string.

15.20 strings.h

Go to the documentation of this file.

```

#ifndef strings_h_
#define strings_h_

#include <string>

const std::string EMPTY_STRING = "";

// ======================================================================
// Person strings
// ======================================================================

namespace person{
    const std::string UNKNOWN_STR = "Unknown";
    const std::string SURNAME_STR = "Surname";
    const std::string NAME_STR = "Name";
    const std::string ROOT = "root";
    const std::string DATE_FROM = "From: ";
    const std::string DATE_TO = " to: ";
}

// ======================================================================
// Tag chooser strings
// ======================================================================

namespace tagchooser{
    const std::string CUSTOM_TAG = "+ Custom tag";
}

// ======================================================================
// Add file
// ======================================================================

namespace addfile{
    const std::string NONE = "None";
}

// ======================================================================
// General buttons texts.
// ======================================================================

namespace button{
    const std::string ADD_NEW = "Add new";
    const std::string CANCEL = "Cancel";
    const std::string REMOVE = "Remove";
}

// ======================================================================
// Exceptions and errors strings.
// ======================================================================

namespace error{
    const std::string UNKNOWN_EXCEP_STR = "Unknown exception catch.";
    const std::string EXCEPTION_STR = "Exception: ";
    const std::string CORRUPTED_DATABASE = "Given database is corrupted: ";
    const std::string EMPTY_BOXES = "All boxes must not be empty.";
    const std::string NO_CHOSEN_FILE = "No file was chosen.";
    const std::string NO_ROLE = "Given role 'no role' cannot be used. It would be considered as no special role.";
    const std::string FORBIDDEN_CHARS = "Text cannot contain forbidden characters. &lt; &gt; \"";
}

// =====================================================================
// Default relations.
// =====================================================================

namespace relation{
    const std::string FATHERHOOD = "Fatherhood";
    const std::string FATHERHOOD_DESC = "Relation between father and his child.";
    const std::string FATHER = "Father";
    const std::string CHILD = "Child";
    const std::string MOTHERHOOD = "Motherhood";
    const std::string MOTHERHOOD_DESC = "Relation between mother and her child.";
    const std::string MOTHER = "Mother";
    const std::string PARTNERSHIP = "Partnership";
    const std::string PARTNERSHIP_DESC = "Relation between two partners.";
    const std::string PARTNER = "Partner";
    const std::string MARRIAGE = "Marriage";
    const std::string MARRIAGE_DESC = "Bound by law between two people. Usually a man (husband) and a woman (wife).";
    const std::string WIFE = "Wife";
    const std::string HUSBAND = "Husband";
    const std::string ADOPTIVE_FATHERHOOD = "Adoptive fatherhood";
    const std::string DESC_ADOPTIVE_FATHERHOOD = "Fatherhood provided by adopting a child.";
    const std::string ADOPTIVE_FATHER = "Adoptive father";
    const std::string ADOPTED_CHILD = "Adopted child";
    const std::string ADOPTIVE_MOTHERHOOD = "Adoptive motherhood";
    const std::string DESC_ADOPTIVE_MOTHERHOOD = "Motherhood provided by adopting a child.";
    const std::string ADOPTIVE_MOTHER = "Adoptive mother";
    const std::string BROTHERHHOD = "Brotherhood";
    const std::string DESC_BROTHERHOOD = "Relation between two brothers.";
    const std::string BROTHER = "Brother";
    const std::string SISTERHOOD = "Sisterhood";
    const std::string DESC_SISTERHOOD = "Relation between two sisters.";
    const std::string SISTER = "Sister";
    const std::string SIBLINGS = "Siblings";
    const std::string DESC_SIBLINGS = "Relation between brother and sister.";
}

// =====================================================================
// Default events.
// =====================================================================

namespace events{
    const std::string WEDDING = "Wedding";
    const std::string DESC_WEDDING = "A social event mainly between soon to be wife and husband.";;
    const std::string GROOM = "Groom";
    const std::string BRIDE = "Bride";
    const std::string BEST_MAN = "Best man";
    const std::string MAID_OF_HONOR = "Maid of Honor";
    const std::string DIVORCE = "Divorce";
    const std::string DESC_DIVORCE = "Divorce is the process of terminating a marriage or marital union.";
    const std::string FORMER_HUSBAND = "Former husband";
    const std::string FORMER_WIFE = "Former wife";
    const std::string FUNERAL = "Funeral";
    const std::string DESC_FUNERAL = "A funeral is a ceremony connected with the final disposition of a corpse, such as a burial or cremation, with the attendant observances.";
    const std::string DEAD_PERSON = "Dead person";
    const std::string GRADUATION = "Graduation";
    const std::string DESC_GRADUATION = "Graduation is the awarding of a diploma to a student by an educational institution.";
    const std::string GRADUATE = "Graduate";
    const std::string NO_ROLE = "no role";
}

// =====================================================================
// HTML Tags and everything with html.
// =====================================================================

namespace html{
    const std::string RESOURCES = "resources";
    const std::string PREFIX_CLOSE = "/close-";
    const std::string PREFIX_FAMILY_TREE = "/tree-";
    const std::string CUSTOM_CSS = "styles.css";
    const std::string GENERAL_INFO = "General information";
    const std::string BIRTH_DATE = "Birth date: ";
    const std::string BIRTH_PLACE = "Birth place: ";
    const std::string DEATH_DATE = "Date of death: ";
    const std::string DEATH_PLACE = "Place of death: ";
    const std::string GENDER = "Gender: ";
    const std::string CUSTOM_TAGS = "Custom tags:";
    const std::string DESCRIPTION = "Description:";
    const std::string NOTE = "Note:";
    const std::string DATE = "Date: ";
    const std::string PLACE = "Place: ";
    const std::string PERSONS = "Persons:";
    const std::string ALL_PERSONS = "All persons";
    const std::string EVENTS = "Events";
    const std::string FAMILY_TREE = "Family tree";
    const std::string FAMILY_TREE_DIAGRAM = "Family tree diagram";
    const std::string RELATIONS = "Relations";
    const std::string CLOSE_FAMILY = "Close family";
    const std::string CLOSE_FAMILY_DIAGRAM = "Close family diagram";
}

// =====================================================================
// JSON Labels.
// =====================================================================

namespace jsonlabel{
    const std::string TITLE = "title";
    const std::string ID = "id";
    const std::string DESCRIPTION = "description";
    const std::string CONTAINS_DATE = "contains date";
    const std::string MORE_PEOPLE = "more people";
    const std::string SPECIAL_ROLES = "special roles";
    const std::string NAME = "name";
    const std::string FIRST_NAME = "first name";
    const std::string SECOND_NAME = "second name";
    const std::string TRAIT = "trait";
    const std::string  GEN_DIFF = "generation difference";
    const std::string REL_TEMPL = "relation templates";
    const std::string EVE_TEMPL = "event templates";
    const std::string TYPE = "type";
    const std::string DATE = "date";
    const std::string LAST_DATE = "last date";
    const std::string TEXT = "text";
    const std::string TAGS = "tags";
    const std::string TAG = "tag";
    const std::string RELATIONS = "relations";
    const std::string EVENTS = "events";
    const std::string SURNAME = "surname";
    const std::string MAIDEN_NAME = "maiden name";
    const std::string VALUE = "value";
    const std::string BIRTH = "birth";
    const std::string DEATH = "death";
    const std::string VIRTUAL_DRIVE = "virtual drive";
    const std::string MEDIA = "media";
    const std::string NOTES = "notes";
    const std::string FILES = "files";
    const std::string PERSONS = "persons";
    const std::string TEMPLATE = "template";
    const std::string TEMPLATES = "templates";
    const std::string FIRST_PERSON = "first person";
    const std::string SECOND_PERSON = "second person";
    const std::string PLACE = "place";
    const std::string ROLE = "role";
    const std::string PERSON = "person";
    const std::string GENDER = "gender";
    const std::string LIVES = "lives";
    const std::string FRONT_TITLE = "title in front";
    const std::string AFTER_TITLE = "title after";
    const std::string FATHER = "father";
    const std::string MOTHER = "mother";
    const std::string PARTNER = "partner";
    const std::string MOVEABLE_TABS = "moveable tabs";
    const std::string TAB_POSITION = "tab position";
    const std::string GLOBAL_MAIN_PERSON = "global main person";
    const std::string CONTAINS_PLACE = "contains place";
    const std::string SHOW_GENERAL_TAB = "show general tab";
    const std::string SHOW_EVENT_TAB = "show event tab";
    const std::string SHOW_FILE_TAB = "show file tab";
    const std::string SHOW_MEDIA_TAB = "show media tab";
    const std::string SHOW_NOTE_TAB = "show note tab";
    const std::string SHOW_RELATION_TAB = "show relation tab";
    const std::string APP_SETTINGS = "app settings";
    const std::string DRIVES = "drives";
    const std::string EXPORT = "export";
    const std::string RED = "red";
    const std::string BLUE = "blue";
    const std::string GREEN = "green";
    const std::string FONT_FAMILY = "font family";
    const std::string FONT_SIZE = "font size";
    const std::string HIGHLIGHTED_COLOR = "highlighted color";
    const std::string PROBANT_COLOR = "proband color";
    const std::string PROMOTED_COLOR = "promoted color";
    const std::string STANDARD_COLOR = "standard color";
    const std::string X_SIZE = "x size";
    const std::string Y_SIZE = "y size";
    const std::string GEN_SIZE_UP = "number of generations up";
    const std::string GEN_SIZE_DOWN = "number of generations down";
    const std::string DIR = "dir";
    const std::string ROOT = "root";
    const std::string HELP = "help";
    const std::string SPLITTER_POSITION_ONE = "splitter position one";
    const std::string SPLITTER_POSITION_TWO = "splitter position two";
    const std::string TEXT_COLOR = "text color";
    const std::string LINE_COLOR = "line color";
    const std::string LINE_WIDTH = "line width";
    const std::string BORDER_RADIUS = "radius";
}

#endif
```

15.21 graphics/dialogs.cpp File Reference

Source file for implementing all dialogs.

```
#include ¨dialogs.h¨
```

15.21.1 Detailed Description

Source file for implementing all dialogs.

15.22 graphics/dialogs.h File Reference

This file contains all window dialogs that will be shown. These classes are relatively small but there are many of them.

```
#include ¨ui_date_dialog.h¨
#include ¨ui_tag_chooser.h¨
#include ¨ui_add_file.h¨
#include ¨ui_relation_templates.h¨
#include ¨ui_event_templates.h¨
#include ¨ui_event_dialog.h¨
#include ¨ui_relation_dialog.h¨
#include ¨ui_help_dialog.h¨
#include ¨ui_choose_person.h¨
#include ¨ui_visual_settings.h¨
#include ¨ui_export_templates.h¨
#include ¨ui_export_project.h¨
#include ¨ui_html_export.h¨
#include ¨ui_picture_export.h¨
#include ¨ui_suggestions.h¨
#include ¨graphics_items.h¨
#include <QtCore>
#include <QtWidgets>
#include <string>
#include <typeinfo>
#include <iostream>
#include ¨../core/family_tree.h¨
#include ¨../core/strings.h¨
#include ¨mainwindow.h¨
```

Classes

- class DateDialog

  Dialog for editing custom dates.
- class TagChooserDialog

  Dialog for editing or creating tags for a person.
- class AddFileDialog

  Dialog for adding file.
- class RelationTemplateDialog

  Dialog showing all templates and possibly editing them or creating new ones.
- class EventTamplatesDialog

  Dialog showing all event templates and possibly editing them or creating new ones.
- class ChoosePersonDialog

  Dialog for choosing person with find window. Parent have to have slot `savePerson()`.
- class RelationDialog

  Dialog showing single relation for editing it.
- class EventDialog

  Dialog showing single event for editing it.
- class HelpDialog

  Dialog with text edit to show help or other texts.
- class VisualSettings

  Dialog for editing and showing graphic settings.
- class ExportTemplatesDialog

  Dialog for exporting some of the event and relation templates.
- class ExportProjectDialog

  Dialog for exporting project.
- class PictureExport

  Dialog for exporting diagram. To choose zoom of the picture.
- class HtmlExport

  Dialog for exporting HTML for one person and all persons.
- class SuggestionsDialog

  Dialog for relation suggestions.

Namespaces

- namespace Ui

  Namespace for classes having all its user interface items.

15.22.1 Detailed Description

This file contains all window dialogs that will be shown. These classes are relatively small but there are many of them.

15.23 dialogs.h

Go to the documentation of this file.

```

#ifndef dialogs_h_
#define dialogs_h_

// Header files created from .ui files.
#include "ui_date_dialog.h"
#include "ui_tag_chooser.h"
#include "ui_add_file.h"
#include "ui_relation_templates.h"
#include "ui_event_templates.h"
#include "ui_event_dialog.h"
#include "ui_relation_dialog.h"
#include "ui_help_dialog.h"
#include "ui_choose_person.h"
#include "ui_visual_settings.h"
#include "ui_export_templates.h"
#include "ui_export_project.h"
#include "ui_html_export.h"
#include "ui_picture_export.h"
#include "ui_suggestions.h"

#include "graphics_items.h"
#include <QtCore>
#include <QtWidgets>
#include <string>
#include <typeinfo>
#include <iostream>
#include "../core/family_tree.h"
#include "../core/strings.h"
#include "mainwindow.h"

QT_BEGIN_NAMESPACE
namespace Ui {
    class DateDialog;
    class TagChooserDialog;
    class AddFileDialog;
    class RelationTemplateDialog;
    class EventTamplatesDialog;
    class EventDialog;
    class RelationDialog;
    class HelpDialog;
    class ChoosePersonDialog;
    class VisualSettings;
    class ExportTemplatesDialog;
    class ExportProjectDialog;
    class PictureExport;
    class HtmlExport;
    class SuggestionsDialog;
}
QT_END_NAMESPACE

class DateDialog : public QDialog {
    Q_OBJECT
    public:
        explicit DateDialog(FamilyTree* FT, WrappedDate* date = nullptr, QWidget* parent = nullptr);
        ~DateDialog();
    public slots:
        void saveDate();
        void unknownDate(bool isUnknown);
    private:
        WrappedDate* date_;
        FamilyTree* FT_;
        Ui::DateDialog *ui;
};

class TagChooserDialog : public QDialog{
    Q_OBJECT
    public:
        TagChooserDialog(FamilyTree* FT, const QString& tag, const QString& value, QWidget* parent = nullptr);
        ~TagChooserDialog();
    public slots:
        void changeVisibleWidgets(int index);
        void saveTag();
    private:
        FamilyTree* FT_;
        bool newOne_;
        Ui::TagChooserDialog *ui;
};

class AddFileDialog : public QDialog{
    Q_OBJECT
    public:
        AddFileDialog(FileType type, const std::vector<std::string>& extensions, VirtualDrive* parentDrive, FamilyTree* FT, QWidget* parent);
        ~AddFileDialog();
    public slots:
        void copyFile();
        void createOrChooseFile();
        void openFile(QTreeWidgetItem* item);
    private:
        FamilyTree* FT_;
        size_t getId(QTreeWidgetItem* item);
        VirtualDrive* drive_;
        FileType type_;
        Ui::AddFileDialog* ui;
};

class RelationTemplateDialog : public QDialog{
    Q_OBJECT
    public:
        explicit RelationTemplateDialog(FamilyTree* FT, QWidget* parent = nullptr);
        ~RelationTemplateDialog();
    public slots:
        void addDefaults();
        void addNewTemplate();
        void removeTemplate();
        void saveTemplate();
        void showTemplate(QTreeWidgetItem* item);
    private:
        bool addingNew_;
        void clear();
        FamilyTree* FT_;
        void refreshListWidget();
        void setEnabledWidgets(bool enabled);
        void saveTemplate(RelationTemplate* rel);
        Ui::RelationTemplateDialog* ui;
};

class EventTamplatesDialog : public QDialog{
    Q_OBJECT
    public:
        explicit EventTamplatesDialog(FamilyTree* FT, QWidget* parent = nullptr);
        ~EventTamplatesDialog();
    public slots:
        void addDefaults();
        void addNewRole();
        void addNewTemplate();
        void removeRole();
        void removeTemplate();
        void saveTemplate();
        void showTemplate(QTreeWidgetItem* item);
    private:
        bool addingNew_;
        void clear();
        FamilyTree* FT_;
        void refreshListWidget();
        void saveTemplate(EventTemplate* event);
        void setEnabledWidgets(bool enabled);
        Ui::EventTamplatesDialog* ui;
};

class ChoosePersonDialog : public QDialog{
    Q_OBJECT
    public:
        ChoosePersonDialog(FamilyTree* FT, Person* person, QWidget* parent = nullptr);
        ~ChoosePersonDialog();
    signals:
        void savePerson(Person* person);
    public slots:
        void clearFilter();
        void filterPersons();
        void saveSelectedPerson();
    private:
        FamilyTree* FT_;
        Person* person_;
        Ui::ChoosePersonDialog* ui;
};

class RelationDialog : public QDialog{
    Q_OBJECT
    public:
        explicit RelationDialog(FamilyTree* FT, QWidget* parent = nullptr, Relation* rel = nullptr);
        ~RelationDialog();
    public slots:
        void chooseFirstPerson();
        void chooseSecondPerson();
        void savePerson(Person* person);
        void saveRelation();
        void showTemplateDetails();
        void switchPersons();
    private:
        FamilyTree* FT_;
        Person* person1_;
        Person* person2_;
        Relation* rel_;
        bool saveFirstPerson_;
        Ui::RelationDialog* ui;
};

class EventDialog : public QDialog{
    Q_OBJECT
    public:
        explicit EventDialog(FamilyTree* FT, QWidget* parent = nullptr, Event* event = nullptr);
        ~EventDialog();
    public slots:
        void addEmptyRole();
        void choosePerson();
        void openCustomDate();
        void refreshDates();
        void refreshPersons(QTreeWidgetItem* item);
        void removeEmptyRole();
        void removePerson();
        void saveEvent();
        void savePerson(Person* person);
        void showTemplate();
    private:
        WrappedDate date_;
        Event* event_;
        FamilyTree* FT_;
        Person* main_;
        void showEventDetails();
        Ui::EventDialog*  ui;

};

class HelpDialog : public QDialog{
    Q_OBJECT
    public:
        explicit HelpDialog(FamilyTree* FT, QWidget* parent = nullptr);
        ~HelpDialog();
    public slots:
        void openHelp(bool help);
    private:
        FamilyTree* FT_;
        Ui::HelpDialog* ui;
};

class VisualSettings : public QDialog{
    Q_OBJECT
    public:
        VisualSettings(QColor probandColor, QColor promotedColor, QColor standardColor, QColor highlighted, int sizeX, int sizeY, QFont font,
                        QPen* pen, QPen* textPen, int borderRadius, QWidget* parent = nullptr);
        ~VisualSettings();
    signals:
        void saveSettings(QColor probandColor, QColor promotedColor, QColor standardColor, QColor highlightedColor, int sizeX, int sizeY, QFont font,
                          QColor lineColor, QColor textColor, int lineWidth, int rounding);
    public slots:
        void highlightedColor();
        void lineColor();
        void refresh();
        void probandColor();
        void promotedColor();
        void saveAndExit();
        void standardColor();
        void textColor();
    private:
        int borderRadius_;
        QFont font_;
        QColor highlightedColor_;
        QPen pen_;
        QColor probandColor_;
        QColor promotedColor_;
        QGraphicsScene* scene;
        int sizeX_;
        int sizeY_;
        QColor standardColor_;
        QPen textPen_;
        Ui::VisualSettings* ui;
};

class ExportTemplatesDialog : public QDialog{
    Q_OBJECT
    public:
        explicit ExportTemplatesDialog(FamilyTree* FT, QWidget* parent = nullptr);
        ~ExportTemplatesDialog();
    public slots:
        void exportTemplates();
    private:
        FamilyTree* FT_;
        Ui::ExportTemplatesDialog* ui;
};

class ExportProjectDialog : public QDialog{
    Q_OBJECT
    public:
        explicit ExportProjectDialog(FamilyTree* FT, QWidget* parent = nullptr);
        ~ExportProjectDialog();
    public slots:
        void checkPerson();
        void exportProject();
    private:
        FamilyTree* FT_;
        Ui::ExportProjectDialog* ui;
};

class PictureExport : public QDialog{
    Q_OBJECT
    public:
        explicit PictureExport(bool closeDiagram, QWidget* parent = nullptr);
        ~PictureExport();
    public slots:
        void exportDiagram();
    private:
        bool closeDiagram_;
        Ui::PictureExport* ui;

};

class HtmlExport : public QDialog{
    Q_OBJECT
    public:
        HtmlExport(FamilyTree* FT, bool all, int up, int down, QWidget* parent = nullptr);
        ~HtmlExport();
    public slots:
        void allAboveGenerations(bool all);
        void allBelowGenerations(bool all);
        void exportAllHtml();
        void exportMainHtml();
    private:
        FamilyTree* FT_;
        Ui::HtmlExport* ui;
};

class SuggestionsDialog : public QDialog{
    Q_OBJECT
    public:
        SuggestionsDialog(FamilyTree* FT, std::vector<RelationSuggestion> suggestions, QWidget* parent = nullptr);
        ~SuggestionsDialog();
    public slots:
        void removeSuggestion();
        void save();
        void selectedSuggestion(QTreeWidgetItem* item);
        void swapPersons();
        void switchedTemplate();
    private:
        std::vector<RelationTemplate*> fatherhoods;
        FamilyTree* FT_;
        std::vector<RelationTemplate*> motherhoods;
        std::vector<RelationTemplate*> partnerships;
        std::vector<RelationTemplate*> siblings;
        Ui::SuggestionsDialog* ui;
};

#endif
```

15.24 graphics/graphics_items.cpp File Reference

This files contains implementation of all graphics items, that are inherited from Qt framework.

```
#include ¨graphics_items.h¨
```

Functions

- bool containsForb (const QString &value)

15.24.1 Detailed Description

This files contains implementation of all graphics items, that are inherited from Qt framework.

15.24.2 Function Documentation

15.24.2.1 containsForb()

```
bool containsForb (
            const QString & value )
```

To clarify that the value contains a forbidden character.

Parameters

value	Which value is checked.

Returns

True if it contains any of forbidden characters.

15.25 graphics/graphics_items.h File Reference

This files contains all graphics items, that are inherited from Qt framework.

```
#include <QtCore>
#include <QtWidgets>
#include <string>
#include <typeinfo>
#include <iostream>
#include ¨../core/family_tree.h¨
#include ¨../core/strings.h¨
```

Classes

- class FileTreeItem

  TreeWidget Item for general file.
- class ProjectItem

  This is a tree widget item to be shown on a project view.
- class EventRoleItem

  Tree widget item for persons in events.
- class PersonsGraphicsItem

Box with persons information.

- class CellGraphicsItem

  Cell around all relations.

- class GraphicsView

  Class for creating QGraphicsView with using wheel for zoom in and out. And grabing by mouse moves the view.

- class RelationItem

  Class for showing relation suggestion and holding the data.

Enumerations

- enum ProjectItemType {
  PROJECT_PERSON , PROJECT_GROUP , PROJECT_FILE , PROJECT_MEDIA ,
  PROJECT_NOTE , PROJECT_EVENTS , PROJECT_RELATIONS }

  What type of a item is to be shown.

Functions

- bool containsForb (const QString &value)

15.25.1 Detailed Description

This files contains all graphics items, that are inherited from Qt framework.

15.25.2 Enumeration Type Documentation

15.25.2.1 ProjectItemType

```
enum ProjectItemType
```

What type of a item is to be shown.

Enumerator

PROJECT_PERSON	
PROJECT_GROUP	
PROJECT_FILE	
PROJECT_MEDIA	
PROJECT_NOTE	
PROJECT_EVENTS	
PROJECT_RELATIONS	

15.25.3 Function Documentation

15.25.3.1 containsForb()

```
bool containsForb (
            const QString & value )
```

To clarify that the value contains a forbidden character.

Parameters

value	Which value is checked.

Returns

True if it contains any of forbidden characters.

15.26 graphics_items.h

Go to the documentation of this file.

```

#ifndef graphics_items_h_
#define graphics_items_h_

#include <QtCore>
#include <QtWidgets>
#include <string>
#include <typeinfo>
#include <iostream>
#include "../core/family_tree.h"
#include "../core/strings.h"

bool containsForb(const QString& value);

class FileTreeItem : public QTreeWidgetItem{
    public:
        FileTreeItem(VirtualDrive* drive, FamilyTree* FT, const QIcon& icon, FileType type);
        FileTreeItem(size_t id, FamilyTree* FT, const QIcon& icon, FileType type);
        void addFile(size_t id, const QIcon& icon);
        void addSubDrive(const QString& folder);
        VirtualDrive* getDrive();
        size_t getId();
        bool isDirectory();
        bool removeFolder();
        bool rename(const QString& name);
    private:
        VirtualDrive* drive_;
        size_t fileId_;
        FamilyTree* FT_;
        bool isDir_;
        void removeFile(size_t id);
        FileType type_;
};

enum ProjectItemType {PROJECT_PERSON, PROJECT_GROUP, PROJECT_FILE, PROJECT_MEDIA, PROJECT_NOTE, PROJECT_EVENTS, PROJECT_RELATIONS};

class ProjectItem : public QTreeWidgetItem{
    public:
        explicit ProjectItem(ProjectItemType type, size_t id = 0);
        size_t getId() const;
        ProjectItemType type() const;
    private:
        size_t id_;
        ProjectItemType type_;
};

class EventRoleItem : public QTreeWidgetItem{
    public:
        explicit EventRoleItem(const std::string& role, size_t id = 0);
        size_t getId();
        void setPerson(const QString& name, size_t id = 0);
    private:
        size_t id_;
};

class PersonsGraphicsItem : public QObject, public QGraphicsItem{
    Q_OBJECT
    public:
        PersonsGraphicsItem(const QStringList& lines, int sizex, int sizey, Person* p, qreal x, qreal y,
                            QColor color, QColor highlighted, QFont font, QWidget* parent, QPen* pen, QPen* textPen, int borderRadius);
        ~PersonsGraphicsItem();
        QRectF boundingRect() const override;
        void paint(QPainter* painter, const QStyleOptionGraphicsItem* option, QWidget* widget) override;
        QPainterPath shape() const override;
    signals:
        void itemClicked(size_t personId);
    protected:
        void hoverEnterEvent(QGraphicsSceneHoverEvent *event) override;
        void hoverLeaveEvent(QGraphicsSceneHoverEvent *event) override;
        void mousePressEvent(QGraphicsSceneMouseEvent* event) override;
    private:
        int borderRadius_;
        int bold_text_;
        QColor color_;
        QFont font_;
        QColor highlighted_;
        bool hovered_;
        QPen* pen_;
        Person* person_;
        int sizex_;
        int sizey_;
        QStringList textLines_;
        QPen* textPen_;
        qreal x_;
        qreal y_;
};

class CellGraphicsItem : public QGraphicsItem{
    public:
        CellGraphicsItem(int sizex, int sizey, qreal x, qreal y, QPen* pen, int borderRadius);
        QRectF boundingRect() const override;
        void paint(QPainter* painter, const QStyleOptionGraphicsItem* option, QWidget* widget) override;
        QPainterPath shape() const override;
    private:
        int borderRadius_;
        QPen* pen_;
        int sizex_;
        int sizey_;
        qreal x_;
        qreal y_;
};

class GraphicsView : public QGraphicsView {
    public:
        explicit GraphicsView(QGraphicsScene* scene, QWidget* parent = nullptr);
    protected:
        void wheelEvent(QWheelEvent* event) override;
};

class RelationItem : public QTreeWidgetItem{
    public:
        RelationItem(FamilyTree* FT, const RelationSuggestion& suggestion, RelationTemplate* templ);
        Relation* getRelation();
        void refreshTexts();
    private:
        FamilyTree* FT_;
        Relation relation_;
};

#endif
```

15.27 graphics/mainwindow.cpp File Reference

Implementation of all standard methods. For slots see mainwindow_slots.cpp.

```
#include ¨mainwindow.h¨
```

15.27.1 Detailed Description

Implementation of all standard methods. For slots see mainwindow_slots.cpp.

15.28 graphics/mainwindow.h File Reference

This header file contains everything around the main window.

```
#include <QtCore>
#include <QtWidgets>
#include <string>
#include <cstdlib>
#include ¨ui_mainwindow.h¨
#include ¨dialogs.h¨
#include ¨graphics_items.h¨
#include ¨../core/family_tree.h¨
#include ¨../core/strings.h¨
```

Classes

- class MainWindow

  Class representing main window using Qt library.

Namespaces

- namespace Ui

  Namespace for classes having all its user interface items.

15.28.1 Detailed Description

This header file contains everything around the main window.

15.29 mainwindow.h

Go to the documentation of this file.

```

#ifndef mainwindow_h_
#define mainwindow_h_

#include <QtCore>
#include <QtWidgets>
#include <string>
#include <cstdlib>

#include "ui_mainwindow.h"

#include "dialogs.h"
#include "graphics_items.h"
#include "../core/family_tree.h"
#include "../core/strings.h"

QT_BEGIN_NAMESPACE
namespace Ui {
    class MainWindow;
}
QT_END_NAMESPACE

class MainWindow : public QMainWindow {
    Q_OBJECT
public:
    explicit MainWindow(QWidget *parent = nullptr);
    ~MainWindow();
    bool exportCloseFamilyDiagram(double scale, const std::string& filename = EMPTY_STRING);
    bool exportTreeDiagram(double scale, const std::string& filename = EMPTY_STRING);
    std::pair<int, int> setNumberOfGenerations(int up, int down);
    static void openExternalFile(const std::string& fileURL);
public slots:
    void addNewEvent();
    void addNewFile();
    void addNewFileFolder();
    void addNewMedia();
    void addNewMediaFolder();
    void addNewNote();
    void addNewNoteFolder();
    void addNewPerson();
    void addNewRelation();
    void allAboveGenerations(bool selected);
    void allBelowGenerations(bool selected);
    void changeGenerationsDown(int value);
    void changeGenerationsUp(int value);
    void checkStoreRefreshPersonsInfo();
    void clearProjectFilter();
    void clearProjectPaths();
    void disableFileFolderPrompt();
    void disableMediaFolderPrompt();
    void disableNoteFolderPrompt();
    void editEvent(QTreeWidgetItem* item);
    void editRelation(QTreeWidgetItem* item);
    void editTag(QTreeWidgetItem* item = nullptr);
    void enableFileFolderPrompt();
    void enableMediaFolderPrompt();
    void enableNoteFolderPrompt();
    void exportCloseFamilyDiagramDialog();
    void exportHtml();
    void exportMainHtml();
    void exportProject();
    void exportTemplates();
    void exportTreeDiagramDialog();
    void filterProjectItems();
    void findPerson();
    void fitTreeInView();
    void importProject();
    void importRootDir();
    void importTemplates();
    void interactWithProjectItem(QTreeWidgetItem* item);
    void openBirthDateDialog();
    void openDeathDateDialog();
    void openEventTemplatesDialog();
    void openExistingProject();
    void openGraphicSettings();
    void openNewFamilyTreeProject();
    void openRelationTemplatesDialog();
    void openSelectedFile(QTreeWidgetItem* item);
    void openSelectedMedia(QTreeWidgetItem* item);
    void openSelectedNote(QTreeWidgetItem* item);
    void promoteRelation();
    void refreshDates();
    void refreshEventTab();
    void refreshGraphics();
    void refreshProjectView();
    void refreshRelationTab();
    void refreshTabsWithFiles();
    void refreshTagsList();
    void removeEvent();
    void removeFileFolder();
    void removeMediaFolder();
    void removeNoteFolder();
    void removePerson();
    void removeRelation();
    void removeTag();
    void renameFile();
    void renameMedia();
    void renameNote();
    bool saveDatabase();
    void saveGraphicsSetting(QColor probandColor, QColor promotedColor, QColor standardColor, QColor highlightedColor, int sizeX, int sizeY, QFont font,
                             QColor lineColor, QColor textColor, int lineWidth, int rounding);
    void selectOlderGenerationOnly();
    void selectSameGenerationOnly();
    void selectYoungerGenerationOnly();
    void setEnabledDeathAtributes(bool disabled);
    void setEventTabVisible();
    void setFileTabVisible();
    void setGeneralTabVisible();
    void setMainPerson(size_t personId);
    void setMediaTabVisible();
    void setNoteTabVisible();
    void setRelationTabVisible();
    void setTabEast();
    void setTabNorth();
    void setTabSouth();
    void setTabWest();
    void showHelp();
    void showRenameFile();
    void showRenameMedia();
    void showRenameNote();
    void splitterChanged();
    void switchNewTreeProject();
    void zoomIn();
    void zoomOut();
protected:
    void closeEvent(QCloseEvent *event);
private:
    void addGeneralFile(FileType type, const std::vector<std::string>& extensions, FileTreeItem* item);
    void addGeneralFolder(FileType type);
    int borderRadius_;
    void clearUi();
    void connectAllSlots();
    void editEvent(size_t id);
    void editRelation(size_t id);
    void enableGeneralFolderPrompt(bool visible, FileType type);
    void enableGeneralFolderRenamePrompt(FileType type);
    bool exportGeneralDiagram(const QString& filename, QGraphicsScene* gscene, double scale = 1);
    QFont font_;
    FamilyTree FT;
    void drawCloseContainer(const std::vector<Relation*>& container,
                                    qreal sizeX, qreal sizeY, qreal topLeftX, qreal topLeftY,
                                    qreal lineX1, qreal lineY1, qreal lineX2, qreal lineY2,
                                    size_t partner, size_t mother, size_t father, int gap,
                                    bool movex, bool movey);
    void drawCloseFamily();
    void drawFamilyTree();
    void drawGeneralGenerations(Person* p, qreal width, qreal midx, qreal midy, int horizontalGap, int verticalGap, size_t barrier,
                                    const std::vector<size_t>& container, bool up,
                                    std::function<std::pair<size_t, size_t>(Person*, size_t, size_t)> numberFunc,
                                    std::function<void(Person*, qreal, qreal, qreal, int, int, size_t)> drawFunc);
    void drawOlderGenerations(Person* p, qreal width, qreal midx, qreal midy, int horizontalGap, int verticalGap, size_t barrier);
    void drawPartner(qreal canvasMidX, qreal canvasMidY, int horizontalGap);
    void drawYoungerGenerations(Person* p, qreal width, qreal midx, qreal midy, int horizontalGap, int verticalGap, size_t barrier);
    std::pair<size_t, size_t> generalGenerationNumber(Person* p, size_t soFar, size_t barrier,
                                                   const std::vector<size_t>& container, bool up,
                                                   std::function<std::pair<size_t, size_t>(Person*, size_t, size_t)> func);
    int genSizeDown_;
    int genSizeUp_;
    QColor highlightedColor_;
    QPen linePen_;
    void loadSettings();
    std::pair<size_t, size_t> olderGenerationNumber(Person* p, int soFar, size_t barrier);
    void openCustomDateDialog(WrappedDate* date);
    void initializeProjectSubView(ProjectItem* parent, Person* person);
    void initializeProjectView();
    void openGeneralFile(QTreeWidgetItem* item, FileType type);
    void openGeneralProject(const std::string& dirPath);
    QColor probandColor_;
    void refreshPersonsInfoTab();
    void refreshProjectEventsView(ProjectItem* item, Person* person);
    void refreshProjectGeneralFilesView(const QIcon& containerIcon, const QString& containerName, const QIcon& fileIcon, FileType type, ProjectItem* item, Person* person);
    void refreshProjectFilesView(ProjectItem* item, Person* person);
    void refreshProjectMediaView(ProjectItem* item, Person* person);
    void refreshProjectNoteView(ProjectItem* item, Person* person);
    void refreshProjectRelationsView(ProjectItem* item, Person* person);
    void refreshUi();
    void removeGeneralFolder(FileType type);
    void renameGeneralFileItem(FileType type);
    QGraphicsScene* scene;
    void savePersonsInfo();
    void setUiElements();
    void showGeneralFiles(QTreeWidget* widget, VirtualDrive* root, FileType type);
    int sizeX_;
    int sizeY_;
    QColor promotedColor_;
    QColor standardColor_;
    QPen textPen_;
    QGraphicsScene* treeScene;
    GraphicsView* treeView;
    Ui::MainWindow* ui;
    std::pair<size_t, size_t> youngerGenerationNumber(Person* p, size_t soFar, size_t barrier);
};
```

```
00542
00543 #endif
```

15.30 graphics/mainwindow_slots.cpp File Reference

Implementation of all slots of the main window. For other methods see mainwindow.cpp.

```
#include ¨mainwindow.h¨
```

15.30.1 Detailed Description

Implementation of all slots of the main window. For other methods see mainwindow.cpp.

15.31 graphics/resources/icons.md File Reference

15.32 help texts/events.md File Reference

15.33 help texts/files.md File Reference

15.34 help texts/generic.md File Reference

15.35 help texts/layout.md File Reference

15.36 help texts/persons.md File Reference

15.37 help texts/relations.md File Reference

15.38 index.md File Reference

15.39 main.cpp File Reference

The main file for starting the application.

```
#include ¨graphics/mainwindow.h¨
#include ¨core/strings.h¨
#include <iostream>
```

Functions

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

15.39.1 Detailed Description

The main file for starting the application.

15.39.2 Function Documentation

15.39.2.1 main()

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

Main function for the application.

Parameters

argc	Number of arguments.
argv	List of all arguments.

Returns

The exit code of the application.

Index