

Radiooooo Qt

1.0

Создано системой Doxygen 1.9.1

1 Документация Radiooooo-Qt	1
1.1 Классы:	1
1.2 Заголовочные файлы:	1
1.3 Сигналы и слоты	1
2 Список задач	3
3 Алфавитный указатель групп	5
3.1 Группы	5
4 Иерархический список классов	7
4.1 Иерархия классов	7
5 Алфавитный указатель классов	9
5.1 Классы	9
6 Группы	11
6.1 Основа Configurator	11
6.1.1 Подробное описание	11
6.1.2 Функции	11
6.1.2.1 Configurator()	11
6.2 Файловая система	12
6.2.1 Подробное описание	12
6.3 Настройки	12
6.3.1 Подробное описание	13
6.3.2 Функции	13
6.3.2.1 configToJson()	13
6.3.2.2 getConfig()	13
6.3.2.3 getConfigStr()	14
6.3.2.4 getQuickCountries()	14
6.3.2.5 updateConfig()	14
6.4 Константы	15
6.4.1 Подробное описание	15
6.4.2 Переменные	15
6.4.2.1 defConfig	15
6.4.2.2 quickSetupCountries	16
6.5 Основа Radiooooo	16
6.5.1 Подробное описание	16
6.5.2 Функции	16
6.5.2.1 Radiooooo()	16
6.6 API	17
6.6.1 Подробное описание	17
6.6.2 Функции	17
6.6.2.1 downloadFile()	17
6.6.2.2 getCountries()	18

6.6.2.3	getSongInfo()	18
6.6.2.4	saveFile()	18
6.7	Медиаплеер	19
6.7.1	Подробное описание	19
6.8	Сигналы Radiooooo	19
6.8.1	Подробное описание	20
6.8.2	Сигналы	20
6.8.2.1	forcePause	20
6.8.2.2	updateProgressBar	20
6.8.2.3	updateStatusMsg	20
6.9	Слоты Radiooooo	21
6.9.1	Подробное описание	21
6.9.2	Открытые слоты	21
6.9.2.1	durationChanged	21
6.9.2.2	getMaxDecade	22
6.9.2.3	getMinDecade	22
6.9.2.4	getMoods	22
6.9.2.5	getQuickCountries	23
6.9.2.6	loadConfig	23
6.9.2.7	playTrigger	23
6.9.2.8	setVolume	23
6.9.2.9	stateChanged	24
6.9.2.10	updateConfig	24
6.9.2.11	updateProgress	24
7	Классы	25
7.1	Класс Configurator	25
7.1.1	Подробное описание	26
7.2	Класс Radiooooo	27
7.2.1	Подробное описание	29
	Предметный указатель	31

Глава 1

Документация Radioooooo-Qt

1.1 Классы:

- [Configurator](#) - класс для работы с настройками приложения
- [Radioooooo](#) - класс для работы с API и аудиофайлами

1.2 Заголовочные файлы:

- [configurator.h](#) - заголовочный файл класса [Configurator](#)
- [radioooooo.h](#) - заголовочный файл класса [Radioooooo](#)
- [defaults.h](#) - заголовочный файл со значениями по умолчанию

1.3 Сигналы и слоты

Асинхронные взаимодействия (в основном с UI) происходят при помощи сигналов и слотов. Например, у класса [Radioooooo](#) есть слот [updateConfig](#) для получения сигнала из QML интерфейса.

Объявление слота в C++:

```
public slots:
    void updateConfig(QString param,
                      QString value,
                      bool enable);
```

В QML выполняется такая конструкция:

```
radio.updateConfig(param, value, enable);
```

Объект `radio` добавляется в контекст QML в `main.cpp`:

```
Radioooooo radio;
engine.rootContext()->setContextProperty("radio", &radio);
```


Глава 2

Список задач

Класс [Configurator](#)

Переписать взаимодействие с QML придерживаясь MVP

Член [Radiooooo::forcePause](#) ()

Переписать без вызова QML из C++ (например при помощи проверки флага в QML)

Глава 3

Алфавитный указатель групп

3.1 Группы

Полный список групп.

Основа Configurator	11
Файловая система	12
Настройки	12
Константы	15
Основа Radiooooo	16
API	17
Медиаплеер	19
Сигналы Radiooooo	19
Слоты Radiooooo	21

Глава 4

Иерархический список классов

4.1 Иерархия классов

Иерархия классов.

QObject	
Configurator	25
Radiooooo	27

Глава 5

Алфавитный указатель классов

5.1 Классы

Классы с их кратким описанием.

Configurator	Используется для загрузки и сохранения настроек приложения	25
Radiooooo	Реализует взаимодействие с https://radiooooo.app/ и логику аудиоплеера	27

Глава 6

Группы

6.1 Основа Configurator

Базовые функции и переменные класса [Configurator](#).

Классы

- class [Configurator](#)
Используется для загрузки и сохранения настроек приложения.

Определения типов

- typedef QMap< QString, QList< QString > > [Configurator::ConfigStorage](#)
Тип данных для хранения настроек

Функции

- [Configurator::Configurator](#) (QObject *parent=nullptr)
[Configurator](#) - дефолтный конструктор класса QObject.

6.1.1 Подробное описание

Базовые функции и переменные класса [Configurator](#).

6.1.2 Функции

6.1.2.1 Configurator()

[Configurator::Configurator](#) (
 QObject * parent = nullptr) [explicit]

[Configurator](#) - дефолтный конструктор класса QObject.

Аргументы

parent	- указатель на родительский QObject
--------	-------------------------------------

6.2 Файловая система

Функции и переменные для работы с файловой системой

Функции

- void `Configurator::initDirs ()`
Создаёт папки в домашней дирректории приложения
- void `Configurator::saveConfig ()`
Сохраняет настройки в json файл
- void `Configurator::loadConfig ()`
Загружает настройки из json файла

Переменные

- const QString `Configurator::appDir = ".radiooooo-qt"`
Имя для домашней дирректории приложения
- const QString `Configurator::appDirPath = QDir::homePath() + "/" + appDir`
Путь к домашней дирректории приложения
- const QString `Configurator::configFilePath = appDirPath + "/config.json"`
Путь к json файлу для настроек
- const QString `Configurator::audioDirPath = appDirPath + "/" + "audio-files"`
Путь к папке для хранения аудиофайлов

6.2.1 Подробное описание

Функции и переменные для работы с файловой системой

6.3 Настройки

Функции и переменные для работы с настройками приложения

Функции

- void `Configurator::initConfig ()`
Заполняет настройки со значениями по умолчанию
- void `Configurator::updateConfig (QString param, QString value, bool enable)`
Обновляет настройки приложения
- `ConfigStorage Configurator::getConfig ()`
Геттер для настроек приложения
- QString `Configurator::getConfigStr ()`
Геттер для настроек приложения
- QJsonDocument `Configurator::configToJson (ConfigStorage c)`
Конвертирует настройки в json документ
- QString `Configurator::getQuickCountries ()`
Возвращает список стран для быстрой настройки в виде строки (для QML)

Переменные

- `const QList< QString > Configurator::moods = defMoods`
Все доступные "настроения".
- `const QList< QString > Configurator::allCountries = defCountries`
Все доступные страны
- `const QMap< QString, QString > Configurator::defaultConfig = defConfig`
Настройки по умолчанию
- `const int Configurator::minDecade = 1910`
Минимальный год
- `const int Configurator::maxDecade = 2020`
Максимальный год
- `const QMap< QString, QString > Configurator::quickCountries = quickSetupCountries`
Список стран для быстрой настройки
- `ConfigStorage Configurator::config`
Переменная для хранения настроек

6.3.1 Подробное описание

Функции и переменные для работы с настройками приложения

6.3.2 Функции

6.3.2.1 configToJson()

```
QJsonDocument Configurator::configToJson (
    Configurator::ConfigStorage c )
```

Конвертирует настройки в json документ

Аргументы

c	- список настроек
---	-------------------

Возвращает

QJsonDocument - json документ с настройками

6.3.2.2 getConfig()

```
Configurator::ConfigStorage Configurator::getConfig ( )
```

Геттер для настроек приложения

Возвращает

настройки в виде ConfigStorage - key:value структуры с настройками приложения

6.3.2.3 getConfigStr()

```
QString Configurator::getConfigStr ( )
```

Геттер для настроек приложения

Возвращает

настройки в виде json строки

6.3.2.4 getQuickCountries()

```
QString Configurator::getQuickCountries ( )
```

Возвращает список стран для быстрой настройки в виде строки (для QML)

Возвращает

QString - json строка со списком стран

6.3.2.5 updateConfig()

```
void Configurator::updateConfig (
    QString param,
    QString value,
    bool enable )
```

Обновляет настройки приложения

Аргументы

param	- имя параметра
value	- значение
enable	- флаг, если true, то добавить настройку, иначе удалить

6.4 Константы

Константы приложения

Переменные

- `const QList< QString > defCountries = {"ABW", "AFG", "AGO", "AIA", "ALA", "ALB", "AND", "ARE", "ARG", "ARM", "ASM", "ATA", "ATF", "ATG", "AUS", "AUT", "AZE", "BDI", "BEL", "BEN", "BES", "BFA", "BGD", "BGR", "BHR", "BHS", "BIH", "BLM", "BLR", "BLZ", "BMU", "BOL", "BRA", "BRB", "BRN", "BTN", "BVT", "BWA", "CAF", "CAN", "CKK", "CHE", "CHL", "CHN", "CIV", "CMR", "COD", "COG", "COK", "COL", "COM", "CPV", "CRI", "CUB", "CUW", "CXR", "CYM", "CYP", "CZE", "DEU", "DJI", "DMA", "DNK", "DOM", "DZA", "ECU", "EGY", "ERI", "ESH", "ESP", "EST", "ETH", "FIN", "FJI", "FLK", "FRA", "FRO", "FSM", "GAB", "GBR", "GEO", "GGY", "GHA", "GIB", "GIN", "GLP", "GMB", "GNB", "GNQ", "GRC", "GRD", "GRL", "GTM", "GUF", "GUM", "GUY", "HKG", "HMD", "HND", "HRV", "HTI", "HUN", "IDN", "IMN", "IND", "IOT", "IRL", "IRN", "IRQ", "ISL", "ISR", "ITA", "JAM", "JEY", "JOR", "JPN", "KAZ", "KEN", "KGZ", "KHM", "KIR", "KNA", "KOR", "KWT", "LAO", "LBN", "LBR", "LBY", "LCA", "LIE", "LKA", "LSO", "LTU", "LUX", "LVA", "MAC", "MAF", "MAR", "MCO", "MDA", "MDG", "MDV", "MEX", "MHL", "MKD", "MLI", "MLT", "MMR", "MNE", "MNG", "MNP", "MOZ", "MRT", "MSR", "MTQ", "MUS", "MWI", "MYS", "MYT", "NAM", "NCL", "NER", "NFK", "NGA", "NIC", "NIU", "NLD", "NOR", "NPL", "NRU", "NZL", "OMN", "PAK", "PAN", "PCN", "PER", "PHL", "PLW", "PNG", "POL", "PRI", "PRK", "PRT", "PRY", "PSE", "PYF", "QAT", "REU", "ROU", "RUS", "RWA", "SAU", "SDN", "SEN", "SGP", "SGS", "SHN", "SJM", "SLB", "SLE", "SLV", "SMR", "SOM", "SPM", "SRB", "SSD", "STP", "SUR", "SVK", "SVN", "SWE", "SWZ", "SXM", "SYC", "SYR", "TCA", "TCD", "TGO", "THA", "TJK", "TKL", "TKM", "TLS", "TON", "TTO", "TUN", "TUR", "TUV", "TWN", "TZA", "UGA", "UKR", "UMI", "URY", "USA", "UZB", "VAT", "VCT", "VEN", "VGB", "VIR", "VNM", "VUT", "WLF", "WSM", "YEM", "ZAF", "ZMB", "ZWE"};`

Список ISO-3166-1 Alpha-3 кодов стран

- `const QMap< QString, QString > quickSetupCountries`
Словарь стран для быстрой настройки <ISO-3166-1 Alpha-3 код>: <название страны>
- `const QMap< QString, QString > defConfig`
Настройки по умолчанию

6.4.1 Подробное описание

Константы приложения

6.4.2 Переменные

6.4.2.1 defConfig

`const QMap<QString, QString> defConfig`

Инициализатор

```
= {
    {"decades", "1980"},
    {"isocodes", "GBR"},
    {"moods", "Fast"}
}
```

Настройки по умолчанию

6.4.2.2 quickSetupCountries

```
const QMap<QString, QString> quickSetupCountries
```

Инициализатор

```
{
    {"GBR", "United Kingdom"},
    {"USA", "USA"},
    {"RUS", "Russia"},
    {"AUS", "Australia"},
    {"any", "Any country"}
}
```

Словарь стран для быстрой настройки <ISO-3166-1 Alpha-3 код>: <название страны>

6.5 Основа Radiooooo

Базовые функции и переменные класса [Radiooooo](#).

Классы

- class [Radiooooo](#)

Реализует взаимодействие с <https://radiooooo.app/> и логику аудиоплеера

Перечисления

- enum { IDLE , PLAYING , PAUSED , DOWNLOADING }

Все состояния аудиоплеера

Функции

- [Radiooooo::Radiooooo](#) (QObject *parent=nullptr)
[Radiooooo](#) - дефолтный конструктор класса QObject.

Переменные

- int [Radiooooo::state](#) = IDLE
Текущее состояние аудиоплеера

6.5.1 Подробное описание

Базовые функции и переменные класса [Radiooooo](#).

6.5.2 Функции

6.5.2.1 Radiooooo()

```
Radiooooo::Radiooooo (
    QObject * parent = nullptr ) [explicit]
```

[Radiooooo](#) - дефолтный конструктор класса QObject.

Аргументы

parent	- указатель на родительский QObject
--------	-------------------------------------

6.6 API

Функции и переменные для работы с API.

Функции

- QObject [Radiooooo::getSongInfo](#) ()
Получает данные через API [Radiooooo](#).
- QObject [Radiooooo::getCountries](#) (QString decade)
Получает доступные исокоды
- QString [Radiooooo::downloadFile](#) (QString path, QString url)
Скачивает файл по URL.
- bool [Radiooooo::saveFile](#) (QString path, QByteArray data)
Сохраняет файл

Переменные

- const QString [Radiooooo::baseAPI](#) = "https://radiooooo.app"
Корневой эндпоинт для API [Radiooooo](#).
- const QString [Radiooooo::getCodesUrl](#) = baseAPI + "/country/mood?decade="
- const QString [Radiooooo::getSongUrl](#) = baseAPI + "/play"
- QNetworkAccessManager * [Radiooooo::netManager](#)
Указатель на QNetworkAccessManager (класс Qt для работы с сетью)

6.6.1 Подробное описание

Функции и переменные для работы с API.

6.6.2 Функции

6.6.2.1 downloadFile()

```
QString Radiooooo::downloadFile (
    QString path,
    QString url ) [private]
```

Скачивает файл по URL.

Аргументы

path	- имя файла для сохранения
url	- url откуда скачать файл

Возвращает

QString - полный путь к файлу

6.6.2.2 getCountries()

```
QJsonObject Radiooooo::getCountries (
    QString decade ) [private]
```

Получает доступные исокоды

Аргументы

Десятилетие	(например 1970, 1980)
-------------	-----------------------

Возвращает

QJsonObject со списком кодов

Заметки

Не проверяется кратность

6.6.2.3 getSongInfo()

```
QJsonObject Radiooooo::getSongInfo ( ) [private]
```

Получает данные через API [Radiooooo](#).

Возвращает

QJsonObject со всей информацией о треке

6.6.2.4 saveFile()

```
bool Radiooooo::saveFile (
    QString path,
    QByteArray data ) [private]
```

Сохраняет файл

Аргументы

path	- путь куда сохранить
data	- QByteArray содержимое файла

Возвращает

true, если удалось сохранить

6.7 Медиаплеер

Функции и переменные для работы с медиаплеером

Функции

- void [Radiooooo::playNext](#) ()
Сохраняет файл и запускает его через QMediaPlayer.

Переменные

- QString [Radiooooo::nowPlaying](#)
Строка, которая хранит название трека и название группы
- qint64 [Radiooooo::audioDuration](#) = 0
Продолжительность трека
- [Configurator](#) * [Radiooooo::cfg](#)
Указатель на [Configurator](#).
- QMediaPlayer * [Radiooooo::mediaPlayer](#)
Указатель на QMediaPlayer (класс Qt для работы с медиафайлами)

6.7.1 Подробное описание

Функции и переменные для работы с медиаплеером

6.8 Сигналы Radiooooo

Сигналы класса [Radiooooo](#).

Сигналы

- void [Radiooooo::updateProgressBar](#) (double progress)
Обновляет прогресс бар
- void [Radiooooo::updateStatusMsg](#) (QString message)
Обновляет сообщение со статусом
- void [Radiooooo::forcePause](#) ()
Обновляет состояние кнопки "play/pause" в QML.

6.8.1 Подробное описание

Сигналы класса [Radiooooo](#).

6.8.2 Сигналы

6.8.2.1 forcePause

```
void Radiooooo::forcePause ( ) [signal]
```

Обновляет состояние кнопки "play/pause" в QML.

[Необходимо сделать](#) Переписать без вызова QML из C++ (например при помощи проверки флага в QML)

6.8.2.2 updateProgressBar

```
void Radiooooo::updateProgressBar (
    double progress ) [signal]
```

Обновляет прогресс бар

Аргументы

progress	- процент завершения
----------	----------------------

6.8.2.3 updateStatusMsg

```
void Radiooooo::updateStatusMsg (
    QString message ) [signal]
```

Обновляет сообщение со статусом

Аргументы

message	- строка с сообщением
---------	-----------------------

6.9 Слоты Radiooooo

Слоты класса [Radiooooo](#).

Открытые слоты

- `QString Radiooooo::loadConfig ()`
Слот для загрузки настроек
- `void Radiooooo::updateConfig (QString param, QString value, bool enable)`
Слот для обновления настроек
- `void Radiooooo::playTrigger (bool play)`
Изменяет состояние аудиоплеера
- `void Radiooooo::skipTrigger ()`
Пропускает трек
- `void Radiooooo::setVolume (int volume)`
Устанавливает громкость
- `void Radiooooo::stateChanged (QMediaPlayer::State playerState)`
Запускает следующий трек, когда текущий заканчивается
- `void Radiooooo::durationChanged (qint64 newDuration)`
Обновляет длительность трека
- `void Radiooooo::updateProgress (qint64 pos)`
Считает прогресс трека и вызывает `updateProgressBar`.
- `QString Radiooooo::getQuickCountries ()`
Получение стран для быстрой настройки
- `QList< QString > Radiooooo::getMoods ()`
Получает список настроений
- `int Radiooooo::getMinDecade ()`
геттер для `minDecade`.
- `int Radiooooo::getMaxDeacde ()`
геттер для `maxDecade`.

6.9.1 Подробное описание

Слоты класса [Radiooooo](#).

6.9.2 Открытые слоты

6.9.2.1 durationChanged

```
void Radiooooo::durationChanged (
    qint64 newDuration ) [slot]
```

Обновляет длительность трека

Аргументы

newDuration	- новая длительность трека
-------------	----------------------------

6.9.2.2 getMaxDeacde

```
int Radiooooo::getMaxDeacde ( ) [slot]
```

геттер для maxDecade.

Возвращает

maxDecade из [Configurator](#)

6.9.2.3 getMinDecade

```
int Radiooooo::getMinDecade ( ) [slot]
```

геттер для minDecade.

Возвращает

minDecade из [Configurator](#)

6.9.2.4 getMoods

```
QList< QString > Radiooooo::getMoods ( ) [slot]
```

Получает список настроек

Возвращает

QList строк

6.9.2.5 getQuickCountries

QString Radiooooo::getQuickCountries () [slot]

Получение стран для быстрой настройки

Возвращает

QString - строка со списком

6.9.2.6 loadConfig

QString Radiooooo::loadConfig () [slot]

Слот для загрузки настроек

Возвращает

QString с настройками

6.9.2.7 playTrigger

void Radiooooo::playTrigger (
 bool play) [slot]

Изменяет состояние аудиоплеера

Аргументы

play	- запустить (true) или остановить (false)
------	-------------------------------------------

6.9.2.8 setVolume

void Radiooooo::setVolume (
 int volume) [slot]

Устанавливает громкость

Аргументы

volume	- громкость (0-100)
--------	---------------------

6.9.2.9 stateChanged

```
void Radiooooo::stateChanged (  
    QMediaPlayer::State playerState ) [slot]
```

Запускает следующий трек, когда текущий заканчивается

Аргументы

playerState	- состояние плеера
-------------	--------------------

6.9.2.10 updateConfig

```
void Radiooooo::updateConfig (  
    QString param,  
    QString value,  
    bool enable ) [slot]
```

Слот для обновления настроек

Аргументы

param	- имя параметра
value	- значение
enable	- флаг, если true, то добавить настройку, иначе удалить

6.9.2.11 updateProgress

```
void Radiooooo::updateProgress (  
    qint64 pos ) [slot]
```

Считает прогресс трека и вызывает updateProgressBar.

Аргументы

pos	- прошедшее время трека
-----	-------------------------

Глава 7

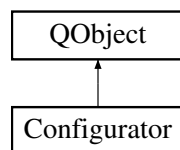
Классы

7.1 Класс Configurator

Используется для загрузки и сохранения настроек приложения.

```
#include <configurator.h>
```

Граф наследования:Configurator:



Открытые типы

- `typedef QMap< QString, QList< QString > > ConfigStorage`
Тип данных для хранения настроек

Открытые члены

- `Configurator (QObject *parent=nullptr)`
`Configurator` - дефолтный конструктор класса `QObject`.
- `void initDirs ()`
Создаёт папки в домашней директории приложения
- `void initConfig ()`
Заполняет настройки со значениями по умолчанию
- `void saveConfig ()`
Сохраняет настройки в json файл
- `void loadConfig ()`
Загружает настройки из json файла
- `void updateConfig (QString param, QString value, bool enable)`
Обновляет настройки приложения
- `ConfigStorage getConfig ()`

- Геттер для настроек приложения
- QString [getConfigStr](#) ()
- Геттер для настроек приложения
- QJsonDocument [configToJson](#) (ConfigStorage c)
- Конвертирует настройки в json документ
- QString [getQuickCountries](#) ()
- Возвращает список стран для быстрой настройки в виде строки (для QML)

Открытые атрибуты

- const QList< QString > [moods](#) = defMoods
- Все доступные "настроения".
- const QList< QString > [allCountries](#) = defCountries
- Все доступные страны
- const QMap< QString, QString > [defaultConfig](#) = [defConfig](#)
- Настройки по умолчанию
- const int [minDecade](#) = 1910
- Минимальный год
- const int [maxDecade](#) = 2020
- Максимальный год
- const QMap< QString, QString > [quickCountries](#) = [quickSetupCountries](#)
- Список стран для быстрой настройки
- const QString [appDir](#) = ".radiooooo-qt"
- Имя для домашней директории приложения
- const QString [appDirPath](#) = QDir::homePath() + "/" + [appDir](#)
- Путь к домашней директории приложения
- const QString [configFilePath](#) = [appDirPath](#) + "/config.json"
- Путь к json файлу для настроек
- const QString [audioDirPath](#) = [appDirPath](#) + "/" + "audio-files"
- Путь к папке для хранения аудиофайлов

Закрытые данные

- [ConfigStorage config](#)
- Переменная для хранения настроек

7.1.1 Подробное описание

Используется для загрузки и сохранения настроек приложения.

[Необходимо сделать](#) Переписать взаимодействие с QML придерживаясь MVP

Объявления и описания членов классов находятся в файлах:

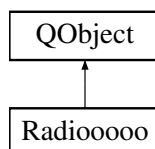
- configurator.h
- configurator.cpp

7.2 Класс Radiooooo

Реализует взаимодействие с <https://radiooooo.app/> и логику аудиоплеера

```
#include <radiooooo.h>
```

Граф наследования:Radiooooo:



Открытые типы

- enum { IDLE , PLAYING , PAUSED , DOWNLOADING }
Все состояния аудиоплеера

Открытые слоты

- QString [loadConfig](#) ()
Слот для загрузки настроек
- void [updateConfig](#) (QString param, QString value, bool enable)
Слот для обновления настроек
- void [playTrigger](#) (bool play)
Изменяет состояние аудиоплеера
- void [skipTrigger](#) ()
Пропускает трек
- void [setVolume](#) (int volume)
Устанавливает громкость
- void [stateChanged](#) (QMediaPlayer::State playerState)
Запускает следующий трек, когда текущий заканчивается
- void [durationChanged](#) (qint64 newDuration)
Обновляет длительность трека
- void [updateProgress](#) (qint64 pos)
Считает прогресс трека и вызывает updateProgressBar.
- QString [getQuickCountries](#) ()
Получение стран для быстрой настройки
- QList< QString > [getMoods](#) ()
Получает список настроений
- int [getMinDecade](#) ()
геттер для minDecade.
- int [getMaxDeacde](#) ()
геттер для maxDecade.

Сигналы

- void `updateProgressBar` (double progress)
Обновляет прогресс бар
- void `updateStatusMsg` (QString message)
Обновляет сообщение со статусом
- void `forcePause` ()
Обновляет состояние кнопки "play/pause" в QML.

Открытые члены

- `Radiooooo` (QObject *parent=nullptr)
`Radiooooo` - дефолтный конструктор класса QObject.

Открытые атрибуты

- int `state` = IDLE
Текущее состояние аудиоплеера

Закрытые члены

- QObject `getSongInfo` ()
Получает данные через API `Radiooooo`.
- QObject `getCountries` (QString decade)
Получает доступные исокоды
- QString `downloadFile` (QString path, QString url)
Скачивает файл по URL.
- bool `saveFile` (QString path, QByteArray data)
Сохраняет файл
- void `playNext` ()
Сохраняет файл и запускает его через QMediaPlayer.

Закрытые данные

- const QString `baseAPI` = "https://radiooooo.app"
Корневой эндпоинт для API `Radiooooo`.
- const QString `getCodesUrl` = `baseAPI` + "/country/mood?decade="
- const QString `getSongUrl` = `baseAPI` + "/play"
- QString `nowPlaying`
Строка, которая хранит название трека и название группы
- qint64 `audioDuration` = 0
Продолжительность трека
- `Configurator` * `cfg`
Указатель на `Configurator`.
- QMediaPlayer * `mediaPlayer`
Указатель на QMediaPlayer (класс Qt для работы с медиафайлами)
- QNetworkAccessManager * `netManager`
Указатель на QNetworkAccessManager (класс Qt для работы с сетью)

7.2.1 Подробное описание

Реализует взаимодействие с <https://radiooooo.app/> и логику аудиоплеера

Заметки

Функции слоты класса [Radiooooo](#) доступны из QML

Взаимодействие с объектом класса [Configurator](#) происходит при помощи сигналов

Объявления и описания членов классов находятся в файлах:

- radiooooo.h
- radiooooo.cpp

Предметный указатель

- API, [17](#)
 - [downloadFile](#), [17](#)
 - [getCountries](#), [18](#)
 - [getSongInfo](#), [18](#)
 - [saveFile](#), [18](#)
- [configToJson](#)
 - [Настройки](#), [13](#)
- [Configurator](#), [25](#)
 - [Основа Configurator](#), [11](#)
- [defConfig](#)
 - [Константы](#), [15](#)
- [downloadFile](#)
 - [API](#), [17](#)
- [durationChanged](#)
 - [Слоты Radiooooo](#), [21](#)
- [forcePause](#)
 - [Сигналы Radiooooo](#), [20](#)
- [getConfig](#)
 - [Настройки](#), [13](#)
- [getConfigStr](#)
 - [Настройки](#), [14](#)
- [getCountries](#)
 - [API](#), [18](#)
- [getMaxDeacde](#)
 - [Слоты Radiooooo](#), [22](#)
- [getMinDecade](#)
 - [Слоты Radiooooo](#), [22](#)
- [getMoods](#)
 - [Слоты Radiooooo](#), [22](#)
- [getQuickCountries](#)
 - [Настройки](#), [14](#)
 - [Слоты Radiooooo](#), [22](#)
- [getSongInfo](#)
 - [API](#), [18](#)
- [loadConfig](#)
 - [Слоты Radiooooo](#), [23](#)
- [playTrigger](#)
 - [Слоты Radiooooo](#), [23](#)
- [quickSetupCountries](#)
 - [Константы](#), [15](#)
- [Radiooooo](#), [27](#)
 - [Основа Radiooooo](#), [16](#)
- [saveFile](#)
 - [API](#), [18](#)
- [setVolume](#)
 - [Слоты Radiooooo](#), [23](#)
- [stateChanged](#)
 - [Слоты Radiooooo](#), [24](#)
- [updateConfig](#)
 - [Настройки](#), [14](#)
 - [Слоты Radiooooo](#), [24](#)
- [updateProgress](#)
 - [Слоты Radiooooo](#), [24](#)
- [updateProgressBar](#)
 - [Сигналы Radiooooo](#), [20](#)
- [updateStatusMsg](#)
 - [Сигналы Radiooooo](#), [20](#)
- [Константы](#), [15](#)
 - [defConfig](#), [15](#)
 - [quickSetupCountries](#), [15](#)
- [Медиаплеер](#), [19](#)
- [Настройки](#), [12](#)
 - [configToJson](#), [13](#)
 - [getConfig](#), [13](#)
 - [getConfigStr](#), [14](#)
 - [getQuickCountries](#), [14](#)
 - [updateConfig](#), [14](#)
- [Основа Configurator](#), [11](#)
 - [Configurator](#), [11](#)
- [Основа Radiooooo](#), [16](#)
 - [Radiooooo](#), [16](#)
- [Сигналы Radiooooo](#), [19](#)
 - [forcePause](#), [20](#)
 - [updateProgressBar](#), [20](#)
 - [updateStatusMsg](#), [20](#)
- [Слоты Radiooooo](#), [21](#)
 - [durationChanged](#), [21](#)
 - [getMaxDeacde](#), [22](#)
 - [getMinDecade](#), [22](#)
 - [getMoods](#), [22](#)
 - [getQuickCountries](#), [22](#)
 - [loadConfig](#), [23](#)
 - [playTrigger](#), [23](#)
 - [setVolume](#), [23](#)
 - [stateChanged](#), [24](#)
 - [updateConfig](#), [24](#)
 - [updateProgress](#), [24](#)
- [Файловая система](#), [12](#)