Project

Generated by Doxygen 1.8.13

Contents

Chapter 1

Hierarchical Index

1.1 Class Hierarchy

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

Chapter 2

Class Index

2.1 Class List

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

Chapter 3

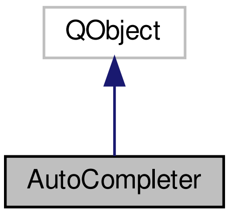
Class Documentation

3.1 AutoCompleter Class Reference

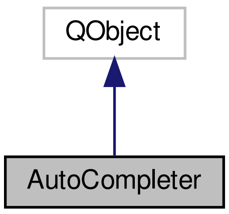
AutoCompleter is a class that provides completion suggestions based on given strings. It also allows updating database of suggestions.

```
#include <autocompleter.h>
```

Inheritance diagram for AutoCompleter:



Collaboration diagram for AutoCompleter:



Public Member Functions

- AutoCompleter (QObject ∗parent=nullptr)

  Constructor.
- void openDictionary (const QString &filePath)

  Reads given file to internal memory and uses values in file as suggestion database.
- QString getSuggestion (const QString &string) const

  Returns a suggestion based on parameter string. Suggestion must start with given string and is then one with lowest levenshtein distance to given string.
- void addSuggestion (const QString &suggestion)

  Adds given suggestion to dictionary opened prior with openDictionary().

Private Member Functions

- size_t levenshteinDist (const QString &string1, const QString &string2) const

  Calculates Levenshtein Distance between given strings.

Private Attributes

- QFile **m_dictionaryFile**
- QSet< QString > **m_dictionary**

3.1.1 Detailed Description

AutoCompleter is a class that provides completion suggestions based on given strings. It also allows updating database of suggestions.

3.1.2 Constructor & Destructor Documentation

3.1.2.1 AutoCompleter()

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

Constructor.

Parameters

parent	Pointer to parent QObject.

3.1.3 Member Function Documentation

3.1.3.1 addSuggestion()

```
void AutoCompleter::addSuggestion (
             const QString & suggestion )
```

Adds given suggestion to dictionary opened prior with openDictionary().

Parameters

suggestion	Suggestion to be added to suggestions.

3.1.3.2 getSuggestion()

```
QString AutoCompleter::getSuggestion (
             const QString & string ) const
```

Returns a suggestion based on parameter string. Suggestion must start with given string and is then one with lowest levenshtein distance to given string.

Parameters

string	Basis for suggestion.

Returns

QString: Suggestion.

3.1.3.3 levenshteinDist()

```
size_t AutoCompleter::levenshteinDist (
             const QString & string1,
             const QString & string2 ) const  [private]
```

Calculates Levenshtein Distance between given strings.

Parameters

string1	First string.
strgin2	Second string.

Returns

size_t: Levenshtein distance between string1 and string2.

3.1.3.4 openDictionary()

```
void AutoCompleter::openDictionary (
             const QString & filePath )
```

Reads given file to internal memory and uses values in file as suggestion database.

Parameters

filePath	Path to csv file of suggestions.

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

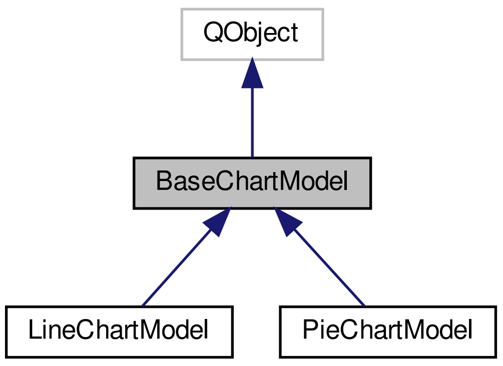
- autocompleter.h
- autocompleter.cpp

3.2 BaseChartModel Class Reference

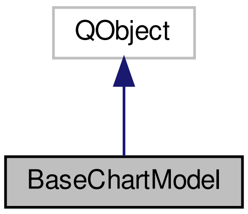
BaseChartModel is a implementation of common features of for different chart models.

```
#include <basechartmodel.h>
```

Inheritance diagram for BaseChartModel:



Collaboration diagram for BaseChartModel:



Signals

- void **titleChanged** () const
- void **axisLabelsChanged** () const
- void **graphsChanged** () const
- void **errorChanged** () const

Public Member Functions

- BaseChartModel (QObject ∗parent=nullptr)

  Constructor.
- void setTitle (const QString &title)

  Sets title to chart model and emits an signal of title changed.
- QString title () const

  returns title of chart model
- void setAxisLabels (const QString &labelX, const QString &labelY)

  Sets axis labels.
- QString labelX () const

  Returns x axis label.
- QString labelY () const

  Return y axis label.
- void publish () const

  Emits a signal that models internal state is ready for displaying in view.
- void setError (const QString &error)

  Sets model to be in error mode and adds a error description.
- virtual quint64 graphCount () const =0

  Pure virtual function that shall returns how many graphs model has.
- virtual void clear ()=0

  Pure virtual function that shall clears all graphs and sets error state to false.

Properties

- QString **title**
- QString **labelX**
- QString **labelY**
- QString **error**
- quint64 **graphCount**

Private Attributes

- QString **m_title**
- QString **m_labelX**
- QString **m_labelY**
- QString **m_error**

3.2.1 Detailed Description

BaseChartModel is a implementation of common features of for different chart models.

3.2.2 Constructor & Destructor Documentation

3.2.2.1 BaseChartModel()

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

Constructor.

Parameters

parent	Pointer to parent QObject.

3.2.3 Member Function Documentation

3.2.3.1 labelX()

```
QString BaseChartModel::labelX ( ) const
```

Returns x axis label.

Returns

QString: X axis label.

3.2.3.2 labelY()

```
QString BaseChartModel::labelY ( ) const
```

Return y axis label.

Returns

QString: Y axis label.

3.2.3.3 setAxisLabels()

```
void BaseChartModel::setAxisLabels (
             const QString & labelX,
             const QString & labelY )
```

Sets axis labels.

Parameters

LabelX	X axis label.
labelY	Y axis label.

3.2.3.4 setError()

```
void BaseChartModel::setError (
             const QString & error )
```

Sets model to be in error mode and adds a error description.

Parameters

error	Error discription.

3.2.3.5 title()

```
QString BaseChartModel::title ( ) const
```

returns title of chart model

Returns

QString: Title.

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

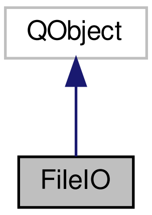
- basechartmodel.h
- basechartmodel.cpp

3.3 FileIO Class Reference

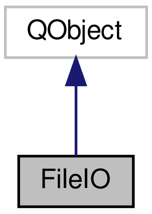
FileIO provides reading and writing data types to and from file. File format is JSON and class also provides list of all saved files.

```
#include <fileio.h>
```

Inheritance diagram for FileIO:



Collaboration diagram for FileIO:



Signals

- void **newFileCreated** ()

Public Member Functions

- FileIO (QObject ∗parent=nullptr)

  Constructor.
- bool saveDataset (QString name, LineChart::LineChartGraph dataset)

  Saves a dataset into a file.
- LineChart::LineChartGraph readDataset (QString name)

  Reads dataset with given name from a file.
- QStringList savedDatasets ()

  Returns all saved dataset names.
- bool savePreset (const QString &name, const PresetController::Preset &preset)

  Saves given preset with given name.
- PresetController::Preset readPreset (const QString &name)

  Returns a saved preset with given name.
- QStringList savedPresets ()

  Returns all saved preset names.

Private Attributes

- QDir **m_datasetDir**
- QDir **m_presetDir**

3.3.1 Detailed Description

FileIO provides reading and writing data types to and from file. File format is JSON and class also provides list of all saved files.

3.3.2 Constructor & Destructor Documentation

3.3.2.1 FileIO()

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

Constructor.

Parameters

parent	Pointer to parent QObject.

3.3.3 Member Function Documentation

3.3.3.1 readDataset()

```
LineChart::LineChartGraph FileIO::readDataset (
            QString name )
```

Reads dataset with given name from a file.

Parameters

name	Name of the dataset.

Returns

LineChart::LineChartGraph: Graph parsed from a file.

3.3.3.2 readPreset()

```
PresetController::Preset FileIO::readPreset (
            const QString & name )
```

Returns a saved preset with given name.

Parameters

name	Name of the preset.

Returns

PresetController::Preset: Preset parsed from file.

3.3.3.3 saveDataset()

```
bool FileIO::saveDataset (
            QString name,
            LineChart::LineChartGraph dataset )
```

Saves a dataset into a file.

Parameters

name	Name for the saved file.
dataset	Data for the saved file.

Returns

bool: True if saving a dateset was successful, False if it couldn't be saved.

3.3.3.4 savedDatasets()

```
QStringList FileIO::savedDatasets ( )
```

Returns all saved dataset names.

Returns

QStringList: List of names.

3.3.3.5 savedPresets()

```
QStringList FileIO::savedPresets ( )
```

Returns all saved preset names.

Returns

QStringList: List of names.

3.3.3.6 savePreset()

```
bool FileIO::savePreset (
             const QString & name,
             const PresetController::Preset & preset )
```

Saves given preset with given name.

Parameters

name	Name of the preset.
preset	Preset to save.

Returns

bool: True if succeess else false.

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

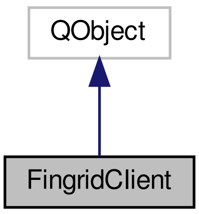
- fileio.h
- fileio.cpp

3.4 FingridClient Class Reference

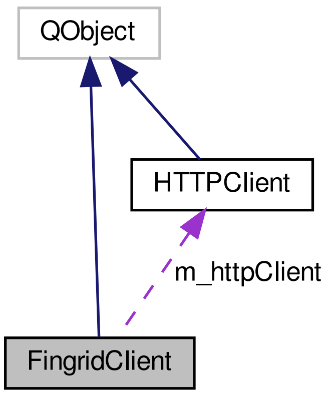
The FingridClient class fetches and parses electricity market and power system data from Fingrid api `https://data.fingrid.fi/en/`.

```
#include <fingridclient.h>
```

Inheritance diagram for FingridClient:



Collaboration diagram for FingridClient:



Classes

- struct Response

  Response format for client. Allows error messages to be passed upstream.

Public Member Functions

- FingridClient (QObject ∗parent=nullptr)

  Constructor.
- Response energyThroughputForecast24h (bool production=true, bool consuption=true)

  Fetches and parses electricity consumption and production forecast for next 24 hours.
- Response energyThroughputHistory (const QDate &startDate, const QDate &endDate, bool production=true, bool consuption=true)

  Fetches and parses energy production and consumption history for given time range.
- Response renewableEnergyProductionForecast24h (bool wind=true, bool solar=true)

  Fetches and parses solar and wind electricity production forecast for next 24 hours.
- Response energyProductionMethods (const QDate &startDate, const QDate &endDate, bool nuclear=true, bool hydro=true, bool wind=true)

  Fetches and parses nuclear, hydro and wind power production history for given time range.
- Response energyProductionDistribution (const QDate &startDate, const QDate &endDate)

  Fetches and parses pie chart from data fetched from renewableEnergyProductionForecast24h.
- Response realTimeFrequency (std::optional< LineChart::LineChartGraph > previous)

  Provides real time monitoring for electric grid frequency. This application fetches data from api that is updated every 3 minutes.
- Response realTimeConsumption (std::optional< LineChart::LineChartGraph > previous)

  Provides real time monitoring for electricity consumption. data from api that is updated every 3 minutes.
- Response realTimeEnergyImport (std::optional< LineChart::LineChartGraph > previous)

  Provides real time monitoring for electricity exrport/import. data from api that is updated every 3 minutes.

Private Member Functions

- QString combineQuery (const QString &id, const QDateTime &startTime, const QDateTime &endTime, const QString &event="events")

  combineQuery formulates api query.
- LineChart parseData (const QJsonArray &data)

  Parses json received from api to LineChart.
- QString parseError (const QJsonDocument &data) const

  Parses error message from json received from api ro error description.
- Response realTimeData (const QString &variableId, const QString &lineName, const QString &unit, std::optional< LineChart::LineChartGraph > previous)

  Fetches data from real time apis. If no previous Graph is given, fetches values from last 20 min Is previoud is given fetches realtime value and adds it to previous and returns previous.

Private Attributes

- HTTPClient **m_httpClient**
- const QString **m_baseAdress**

3.4.1 Detailed Description

The FingridClient class fetches and parses electricity market and power system data from Fingrid api `https://data.fingrid.fi/en/`.

3.4.2 Constructor & Destructor Documentation

3.4.2.1 FingridClient()

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

Constructor.

Parameters

parent	Pointer to parent QObject.

3.4.3 Member Function Documentation

3.4.3.1 combineQuery()

```
QString FingridClient::combineQuery (
            const QString & id,
            const QDateTime & startTime,
            const QDateTime & endTime,
            const QString & event = "events" )  [private]
```

combineQuery formulates api query.

Parameters

id	api specific id for data set.
startDate	Beginning of time range.
endDate	End of time range.
event	Type of request (event or events).

Returns

QString: Parsed query.

3.4.3.2 energyProductionDistribution()

```
FingridClient::Response FingridClient::energyProductionDistribution (
            const QDate & startDate,
            const QDate & endDate )
```

Fetches and parses pie chart from data fetched from renewableEnergyProductionForecast24h.

Parameters

startDate	Beginning of time range.
endDate	End of time range.

Returns

Response: Parsed Response with pieGraph or error state and message.

3.4.3.3 energyProductionMethods()

```
FingridClient::Response FingridClient::energyProductionMethods (
             const QDate & startDate,
             const QDate & endDate,
             bool nuclear = true,
             bool hydro = true,
             bool wind = true )
```

Fetches and parses nuclear, hydro and wind power production history for given time range.

Parameters

startDate	Beginning of time range.
endDate	End of time range.
nuclear	Is nuclear graph fetched and parsed to response.
hydro	Is hydro graph fetched and parsed to response.
wind	Is wind graph fetched and parsed to response.

Returns

Response: Parsed Response with lineGraph or error state and message.

3.4.3.4 energyThroughputForecast24h()

```
FingridClient::Response FingridClient::energyThroughputForecast24h (
             bool production = true,
             bool consuption = true )
```

Fetches and parses electricity consumption and production forecast for next 24 hours.

Parameters

production	Is production graph fetched and parsed to response.
consumption	Is consumption graph fetched and parsed to response.

Returns

Response: Parsed Response with lineGraph or error state and message

3.4.3.5 energyThroughputHistory()

```
FingridClient::Response FingridClient::energyThroughputHistory (
             const QDate & startDate,
             const QDate & endDate,
             bool production = true,
             bool consuption = true )
```

Fetches and parses energy production and consumption history for given time range.

Parameters

startDate	Beginning of time range.
endDate	End of time range.
production	Is production graph fetched and parsed to response.
consumption	Is consumption graph fetched and parsed to response.

Returns

Response: Parsed Response with lineGraph or error state and message

3.4.3.6 parseData()

```
LineChart FingridClient::parseData (
             const QJsonArray & data )  [private]
```

Parses json received from api to LineChart.

Parameters

data	Json array received from api query.

Returns

LineChart: Parsed data.

3.4.3.7 parseError()

```
QString FingridClient::parseError (
             const QJsonDocument & data ) const  [private]
```

Parses error message from json received from api ro error description.

Parameters

data	Json array received from api query.

Returns

QString: Description of error

3.4.3.8 realTimeConsumption()

```
FingridClient::Response FingridClient::realTimeConsumption (
            std::optional< LineChart::LineChartGraph > previous )
```

Provides real time monitoring for electricity consumption. data from api that is updated every 3 minutes.

Parameters

previous	When previous == std::nullopt new LineChart::LineChartGraph is created and populated with last ∼20 minutes of consumption data else if api provides newer data than previous contains, new data is appended to previous either way graph is returned in Response.

Returns

Response: Parsed Response with pieGraph or error state and message.

3.4.3.9 realTimeData()

```
FingridClient::Response FingridClient::realTimeData (
            const QString & variableId,
            const QString & lineName,
            const QString & unit,
            std::optional< LineChart::LineChartGraph > previous )  [private]
```

Fetches data from real time apis. If no previous Graph is given, fetches values from last 20 min Is previoud is given fetches realtime value and adds it to previous and returns previous.

Parameters

variable←↩ Id	Api Variable Id.
lineName	Name of the line.
unit	Unit of the line.
previous	If not std::nullopt new value is added to this graph.

Returns

Response: Contains lineGraphs with last 20 minutes of values or with previoud graphs with new value appended.

3.4.3.10 realTimeEnergyImport()

```
FingridClient::Response FingridClient::realTimeEnergyImport (
            std::optional< LineChart::LineChartGraph > previous )
```

Provides real time monitoring for electricity exrport/import. data from api that is updated every 3 minutes.

Parameters

previous	When previous == std::nullopt new LineChart::LineChartGraph is created and populated with last ∼20 minutes of exrport/import data else if api provides newer data than previous contains, new data is appended to previous either way graph is returned in Response.

Returns

Response: Parsed Response with pieGraph or error state and message.

3.4.3.11 realTimeFrequency()

```
FingridClient::Response FingridClient::realTimeFrequency (
            std::optional< LineChart::LineChartGraph > previous )
```

Provides real time monitoring for electric grid frequency. This application fetches data from api that is updated every 3 minutes.

Parameters

previous	When previous == std::nullopt new LineChart::LineChartGraph is created and populated with last ∼20 minutes of frequency data else if api provides newer data than previous contains, new data is appended to previous either way graph is returned in Response.

Returns

Response: Parsed Response with pieGraph or error state and message.

3.4.3.12 renewableEnergyProductionForecast24h()

```
FingridClient::Response FingridClient::renewableEnergyProductionForecast24h (
            bool wind = true,
            bool solar = true )
```

Fetches and parses solar and wind electricity production forecast for next 24 hours.

Parameters

wind	Is wind energy production graph fetched and parsed to response.
solar	Is solar energy produciton graph fetched and parsed to response.

Returns

Response: Parsed Response with lineGraph or error state and message.

3.4.4 Member Data Documentation

3.4.4.1 m_baseAdress

```
const QString FingridClient::m_baseAdress  [private]
```

Initial value:

```
= "https://api.fingrid.fi/v1/variable/"
                                     "%1/%2/json?start_time=%3Z&end_time=%4Z"
```

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

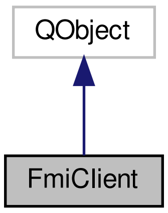
- fingridclient.h
- fingridclient.cpp

3.5 FmiClient Class Reference

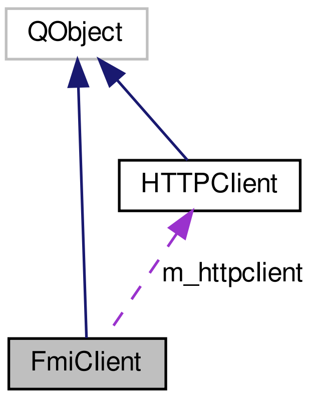
The FmiClient class fetches and parses weather data from The Finnish Meteorological Institute api `https://en.ilmatieteenlaitos.fi/open-data-manual`.

```
#include <fmiclient.h>
```

Inheritance diagram for FmiClient:



Collaboration diagram for FmiClient:



Classes

- struct Response

  Response format for client. Allows error messages to be passed upstream.

Public Member Functions

- FmiClient (QObject ∗parent=nullptr)

  Constructor.
- Response weatherForecast24h (const QString &location, bool temperature=true, bool windSpeed=true)

  Fetches and parses temperature and wind speed forecast for next 24 hours.
- Response weatherHistory (const QDate &start, const QDate &end, const QString &location, bool temperature=true, bool windSpeed=true, bool cloudiness=true)

  Fetches and parses temperature, wind speed and cloudiness history for given time range.
- Response monthlyTemperatureAverages (const unsigned int month, const unsigned int year, const QString &location, bool avg=true, bool min=true, bool max=true)

  Fetches and parses min, max and avg temperature for given month.

Private Member Functions

- QString get24hWeatherForecastQuery (const QString &location)

  Parses query string for weather forecast.
- QString getWeatherHistoryQuery (const QDate &startDate, const QDate &endDate, const QString &location)

  Parses query string for weather history.
- QString getMonthlyTempStatsQuery (const unsigned int month, const unsigned int year, const QString &location)

  Parses query string for monthly temperature query.
- LineChart::LineChartGraph createWeatherCharts (const QJsonArray &array, const QStringList &parameters, const QStringList &chartNames)

  Parses json received from api in to LineChart::LineChartGraph. Takes lists of parametes and chart names as parameter. These lists must have same size. Each element in parameters and lineNames describe one chart to be parsed. Charts are added to return Graph in same order as they are defined in parameter and chartNames lists.
- QString parseErrorMessage (const QJsonDocument &message) const

  parses error received from api to error string.

Private Attributes

- HTTPClient **m_httpclient**
- const unsigned int **m_timestep** = 60
- const QString **m_apiaddress** = "https://opendata.fmi.fi/wfs"

3.5.1 Detailed Description

The FmiClient class fetches and parses weather data from The Finnish Meteorological Institute api `https↩ ://en.ilmatieteenlaitos.fi/open-data-manual`.

3.5.2 Constructor & Destructor Documentation

3.5.2.1 FmiClient()

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

Constructor.

Parameters

parent	Pointer to parent QObject.

3.5.3 Member Function Documentation

3.5.3.1 createWeatherCharts()

```
LineChart::LineChartGraph FmiClient::createWeatherCharts (
              const QJsonArray & array,
              const QStringList & parameters,
              const QStringList & chartNames )   [private]
```

Parses json received from api in to LineChart::LineChartGraph. Takes lists of parametes and chart names as parameter. These lists must have same size. Each element in parameters and lineNames describe one chart to be parsed. Charts are added to return Graph in same order as they are defined in parameter and chartNames lists.

Parameters

array	a json array received from api.
parameters	List of search parameter names (must be as long as chartNames).
chartNames	List of chart names (must be as long as parameters).

Returns

LineChart::LineChartGraph: Graph parsed from given data

3.5.3.2 get24hWeatherForecastQuery()

```
QString FmiClient::get24hWeatherForecastQuery (
             const QString & location )  [private]
```

Parses query string for weather forecast.

Parameters

location	Name of the city/town.

Returns

QString: Query.

3.5.3.3 getMonthlyTempStatsQuery()

```
QString FmiClient::getMonthlyTempStatsQuery (
             const unsigned int month,
             const unsigned int year,
             const QString & location )  [private]
```

Parses query string for monthly temperature query.

Parameters

month	Month of query.
year	Year of query.
location	Name of the city/town.

Returns

QString: Query.

3.5.3.4 getWeatherHistoryQuery()

```
QString FmiClient::getWeatherHistoryQuery (
             const QDate & startDate,
             const QDate & endDate,
             const QString & location )  [private]
```

Parses query string for weather history.

Parameters

startDate	Start of time range.
endDate	End of timerange.
location	Name of the city/town.

Returns

QString: Query.

3.5.3.5 monthlyTemperatureAverages()

```
FmiClient::Response FmiClient::monthlyTemperatureAverages (
            const unsigned int month,
            const unsigned int year,
            const QString & location,
            bool avg = true,
            bool min = true,
            bool max = true )
```

Fetches and parses min, max and avg temperature for given month.

Parameters

month	Month.
year	Year.
location	Location of queried forecast.
avg	Is avg graph fetched and parsed to response.
min	Is min graph fetched and parsed to response.
max	Is max graph fetched and parsed to response.

Returns

Response: Response with Parsed lineGraph or error state and message.

3.5.3.6 parseErrorMessage()

```
QString FmiClient::parseErrorMessage (
            const QJsonDocument & message ) const  [private]
```

parses error received from api to error string.

Parameters

message	json received from api query.

Returns

QString: Description of the error.

3.5.3.7 weatherForecast24h()

```
FmiClient::Response FmiClient::weatherForecast24h (
             const QString & location,
             bool temperature = true,
             bool windSpeed = true )
```

Fetches and parses temperature and wind speed forecast for next 24 hours.

Parameters

location	Location of queried forecast.
temperature	Is temperature graph fetched and parsed to response.
windSpeed	Is windSpeed graph fetched and parsed to response.

Returns

Response: Response with Parsed lineGraph or error state and message.

3.5.3.8 weatherHistory()

```
FmiClient::Response FmiClient::weatherHistory (
             const QDate & start,
             const QDate & end,
             const QString & location,
             bool temperature = true,
             bool windSpeed = true,
             bool cloudiness = true )
```

Fetches and parses temperature, wind speed and cloudiness history for given time range.

Parameters

start	Start date of the time range.
end	End date of the time range.
location	Location of queried forecast.
temperature	Is temperature graph fetched and parsed to response.
windSpeed	Is windSpeed graph fetched and parsed to response.
cloudiness	Is cloudiness graph fetched and parsed to response.

Returns

Response: Response with Parsed lineGraph or error state and message.

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

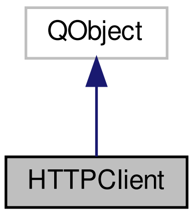
- fmiclient.h
- fmiclient.cpp

3.6 HTTPClient Class Reference

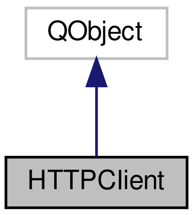
HTTPClient is a synchronous http client that supports get request and passing http headers.

```
#include <httpclient.h>
```

Inheritance diagram for HTTPClient:



Collaboration diagram for HTTPClient:



Classes

- struct Response

  Response format for client. Allows error state to be passed upstream.

Public Member Functions

- HTTPClient (QObject ∗parent=nullptr)

  Constructor.
- Response get (const QString &query)

  Makes http get request and returns received data and error state. Added headers are added to request.
- void addHeader (const QByteArray &headerName, const QByteArray &value)

  Adds a header, these headers are included in request made after addind.
- void clearHeaders ()

  Clears all headers.

Private Attributes

- QNetworkAccessManager **m_networkManager**
- QEventLoop **m_eventLoop**
- QList< QPair< QByteArray, QByteArray > > **m_headers**

3.6.1 Detailed Description

HTTPClient is a synchronous http client that supports get request and passing http headers.

3.6.2 Constructor & Destructor Documentation

3.6.2.1 HTTPClient()

```
HTTPClient::HTTPClient (
            QObject * parent = nullptr )
```

Constructor.

Parameters

parent	Pointer to parent QObject.

3.6.3 Member Function Documentation

3.6.3.1 addHeader()

```
void HTTPClient::addHeader (
            const QByteArray & headerName,
            const QByteArray & value )
```

Adds a header, these headers are included in request made after addind.

Parameters

headerName	Name of the header.
value	Value of the header.

3.6.3.2 get()

```
HTTPClient::Response HTTPClient::get (
            const QString & query )
```

Makes http get request and returns received data and error state. Added headers are added to request.

Parameters

query	Http(s) query for get request.

Returns

Response: Data and error state from response to query made.

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

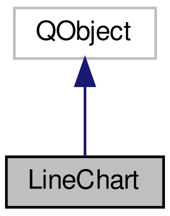
- httpclient.h
- httpclient.cpp

3.7 LineChart Class Reference

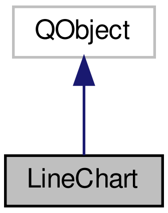
LineChart is a abstraction for 2d line chart that uses timeline x axis and real y axis. Internally x axis values are stored as milliseconds since epoch.

```
#include <linechart.h>
```

Inheritance diagram for LineChart:



Collaboration diagram for LineChart:



Classes

- struct LineChartGraph

  LineChartGraph is a abstraction for 2d graph with multiple line charts.

Public Member Functions

- LineChart (QObject ∗parent=nullptr)

  Constructor.
- LineChart (const LineChart &obj)

  Copy Constructor.
- LineChart & operator= (const LineChart &obj)

  Copy Constructor.
- void setName (const QString &name)

  Sets name to the chart.
- QString name () const

  Returns the name of the chart.
- void addPoint (const QDateTime &x, const double &y)

  Adds a point to LineChart.
- QDateTime xMax () const

  Returns largest X value held by linechart.
- QDateTime xMin () const

  Returns smalles X value held by linechart.
- double yMax () const

  Returns largest Y value held by linechart.
- double yMin () const

  Returns smalles Y value held by linechart.
- quint64 length () const

  Returns number of points held by linechart.
- QList< QPointF > values () const

  Returns list of points. Points are in format QPointF<MsecSinceEpoch, value>.

Private Attributes

- QString **m_name**
- QList< QPointF > **m_values**

3.7.1 Detailed Description

LineChart is a abstraction for 2d line chart that uses timeline x axis and real y axis. Internally x axis values are stored as milliseconds since epoch.

3.7.2 Constructor & Destructor Documentation

3.7.2.1 LineChart() [1/2]

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

Constructor.

Parameters

parent	Pointer to parent QObject.

3.7.2.2 LineChart() [2/2]

```
LineChart::LineChart (
            const LineChart & obj )
```

Copy Constructor.

Parameters

obj	Object to copy.

3.7.3 Member Function Documentation

3.7.3.1 addPoint()

```
void LineChart::addPoint (
            const QDateTime & x,
            const double & y )
```

Adds a point to LineChart.

Parameters

x	X value of the point.
y	Y value of the point.

3.7.3.2 length()

```
quint64 LineChart::length ( ) const
```

Returns number of points held by linechart.

Returns

quint64: Number of points.

3.7.3.3 name()

```
QString LineChart::name ( ) const
```

Returns the name of the chart.

Returns

QString: Name of the linechart.

3.7.3.4 operator=()

```
LineChart & LineChart::operator= (
            const LineChart & obj )
```

Copy Constructor.

Parameters

obj	Object to copy.

3.7.3.5 setName()

```
void LineChart::setName (
            const QString & name )
```

Sets name to the chart.

Parameters

name	name to be set.

3.7.3.6 values()

```
QList< QPointF > LineChart::values ( ) const
```

Returns list of points. Points are in format QPointF<MsecSinceEpoch, value>.

Returns

QList<QPointF>>: List of added points.

3.7.3.7 xMax()

```
QDateTime LineChart::xMax ( ) const
```

Returns largest X value held by linechart.

Returns

QDateTime: Maximum X value.

3.7.3.8 xMin()

```
QDateTime LineChart::xMin ( ) const
```

Returns smalles X value held by linechart.

Returns

QDateTime: Minimum X Value.

3.7.3.9 yMax()

```
double LineChart::yMax ( ) const
```

Returns largest Y value held by linechart.

Returns

QDateTime: Maximum Y value.

3.7.3.10 yMin()

```
double LineChart::yMin ( ) const
```

Returns smalles Y value held by linechart.

Returns

double: Minimum Y Value.

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

- linechart.h
- linechart.cpp

3.8 LineChart::LineChartGraph Struct Reference

LineChartGraph is a abstraction for 2d graph with multiple line charts.

```
#include <linechart.h>
```

Public Member Functions

- void addLine (const LineChart &line)

  Adds line to graph if given line is not empty.

Public Attributes

- QString **title**
- QString **xLabel**
- QString **yLabel**
- std::vector< LineChart > **lines**

3.8.1 Detailed Description

LineChartGraph is a abstraction for 2d graph with multiple line charts.

3.8.2 Member Function Documentation

3.8.2.1 addLine()

```
void LineChart::LineChartGraph::addLine (
            const LineChart & line )  [inline]
```

Adds line to graph if given line is not empty.

Parameters

line	LineChart to be added.

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

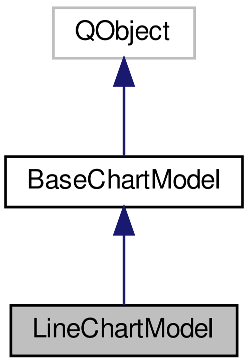
- linechart.h

3.9 LineChartModel Class Reference

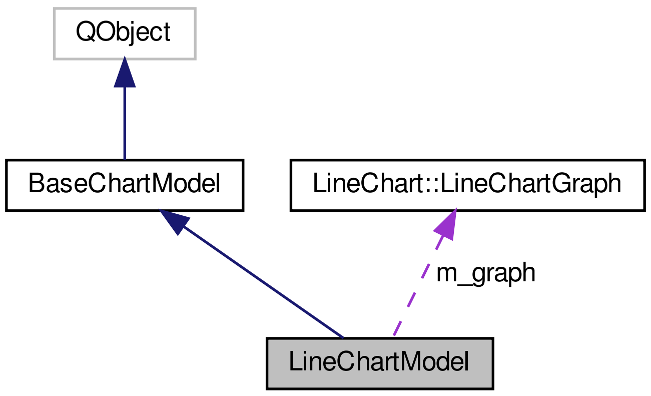
LineChartModel is used display LineChart::LineChartGraph in view.

```
#include <linechartmodel.h>
```

Inheritance diagram for LineChartModel:



Collaboration diagram for LineChartModel:



Public Member Functions

- LineChartModel (QObject ∗parent=nullptr)

  Constructor.
- void addGraph (const LineChart::LineChartGraph &graph)

  Adds graph to model and updates title and axis labels.
- LineChart::LineChartGraph getGraph () const

  Returns currently held graph.
- void clear () override

  Removes graph and resets all attributes to default.

- quint64 graphCount () const override

  Returns how many lines currently held graph has.
- Q_INVOKABLE void transferSeries (QtCharts::QLineSeries ∗series, int index)

  Transfer LineChart that is stored in currently held graph with given index to series given as parameter.
- QDateTime xAxisMax () const

  Returns maximum X value of any line in currently held graph.
- QDateTime xAxisMin () const

  Returns minimum X value of any line in currently held graph.
- double yAxisMax () const

  Returns maximum Y value of any line in currently held graph.
- double yAxisMin () const

  Returns minimum Y value of any line in currently held graph.

Properties

- QVariant **xAxisMax**
- QVariant **xAxisMin**
- QVariant **yAxisMax**
- QVariant **yAxisMin**

Private Attributes

- LineChart::LineChartGraph **m_graph**

Additional Inherited Members

3.9.1 Detailed Description

LineChartModel is used display LineChart::LineChartGraph in view.

3.9.2 Constructor & Destructor Documentation

3.9.2.1 LineChartModel()

```
LineChartModel::LineChartModel (
             QObject ∗ parent = nullptr )  [explicit]
```

Constructor.

Parameters

parent	Pointer to parent QObject.

3.9.3 Member Function Documentation

3.9.3.1 addGraph()

```
void LineChartModel::addGraph (
            const LineChart::LineChartGraph & graph )
```

Adds graph to model and updates title and axis labels.

Parameters

graph	LineChartGraph to be added.

3.9.3.2 getGraph()

```
LineChart::LineChartGraph LineChartModel::getGraph ( ) const
```

Returns currently held graph.

Returns

LineChartGraph: Current LineChartGraph.

3.9.3.3 graphCount()

```
quint64 LineChartModel::graphCount ( ) const  [override], [virtual]
```

Returns how many lines currently held graph has.

Returns

quint64: Line count of currently held graph.

Implements BaseChartModel.

3.9.3.4 transferSeries()

```
void LineChartModel::transferSeries (
            QtCharts::QLineSeries * series,
            int index )
```

Transfer LineChart that is stored in currently held graph with given index to series given as parameter.

Parameters

series	Pointer to QLineSeries.
index	Index of LineChart in currently held graph.

3.9.3.5 xAxisMax()

```
QDateTime LineChartModel::xAxisMax ( ) const
```

Returns maximum X value of any line in currently held graph.

Returns

QDateTime: Maximum value for X-axis.

3.9.3.6 xAxisMin()

```
QDateTime LineChartModel::xAxisMin ( ) const
```

Returns minimum X value of any line in currently held graph.

Returns

QDateTime: Minimum value for X-axis.

3.9.3.7 yAxisMax()

```
double LineChartModel::yAxisMax ( ) const
```

Returns maximum Y value of any line in currently held graph.

Returns

double: Maximum value for Y-axis.

3.9.3.8 yAxisMin()

```
double LineChartModel::yAxisMin ( ) const
```

Returns minimum Y value of any line in currently held graph.

Returns

double: Minimum value for Y-axis.

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

- linechartmodel.h
- linechartmodel.cpp

3.10 PieChartModel::PieChartGraph Struct Reference

Abstraction for pie graph.

```
#include <piechartmodel.h>
```

Public Attributes

- QString **title**
- QList< Slice > **slices**

3.10.1 Detailed Description

Abstraction for pie graph.

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

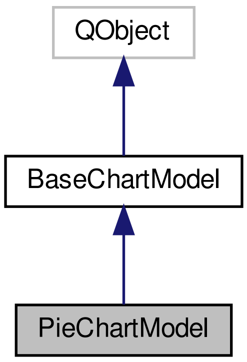
- piechartmodel.h

3.11 PieChartModel Class Reference

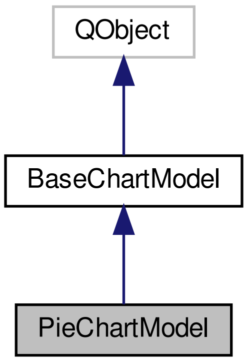
The PieChart is used display a pie chart in a view.

```
#include <piechartmodel.h>
```

Inheritance diagram for PieChartModel:



Collaboration diagram for PieChartModel:



Classes

- struct PieChartGraph

  Abstraction for pie graph.
- struct Slice

  Abstraction for pie slice.

Public Member Functions

- PieChartModel (QObject ∗parent=nullptr)

  Constructor.
- void addSlice (const Slice &slice)

  adds a pie slice to pie chart.
- QList< Slice > slices () const

  returns list of Slices in pie chart.
- Q_INVOKABLE void transferSeries (QtCharts::QPieSeries ∗series)

  transfers contained pieslices to series given ad parameter.
- quint64 graphCount () const override

  Returns the slice count of the model.
- void clear () override

  Removes all slices and resets all parameters to default.

Private Attributes

- QList< Slice > **m_slices**

Additional Inherited Members

3.11.1 Detailed Description

The PieChart is used display a pie chart in a view.

3.11.2 Constructor & Destructor Documentation

3.11.2.1 PieChartModel()

```
PieChartModel::PieChartModel (
            QObject ∗ parent = nullptr )  [explicit]
```

Constructor.

Parameters

parent	Pointer to parent QObject.

3.11.3 Member Function Documentation

3.11.3.1 addSlice()

```
void PieChartModel::addSlice (
             const Slice & slice )
```

adds a pie slice to pie chart.

Parameters

Slice	Slice to be added.

3.11.3.2 graphCount()

```
quint64 PieChartModel::graphCount ( ) const  [override], [virtual]
```

Returns the slice count of the model.

Returns

quint64: Slice count of the model.

Implements BaseChartModel.

3.11.3.3 transferSeries()

```
void PieChartModel::transferSeries (
             QtCharts::QPieSeries * series )
```

transfers contained pieslices to series given ad parameter.

Parameters

series	Pointer to a QPieSeries to be filled.

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

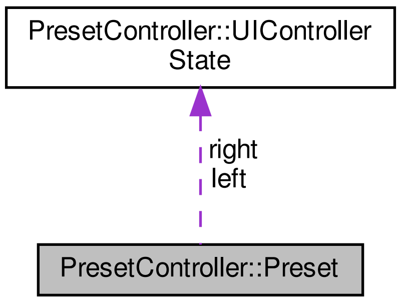
- piechartmodel.h
- piechartmodel.cpp

3.12 PresetController::Preset Struct Reference

Abstraction preset with states for each UIController.

```
#include <presetcontroller.h>
```

Collaboration diagram for PresetController::Preset:



Public Attributes

- UIControllerState **left**
- UIControllerState **right**

3.12.1 Detailed Description

Abstraction preset with states for each UIController.

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

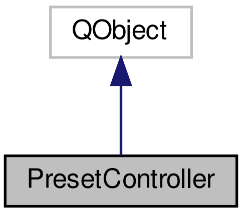
- presetcontroller.h

3.13 PresetController Class Reference

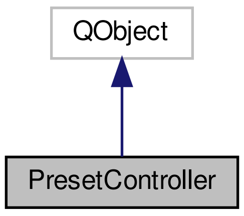
The Preset handles reading and writing state of UIControllers as presets.

```
#include <presetcontroller.h>
```

Inheritance diagram for PresetController:



Collaboration diagram for PresetController:



Classes

- struct Preset

  Abstraction preset with states for each UIController.
- struct UIControllerState

  Abstraction for the state of the user interface state defined by UIController.

Public Member Functions

- PresetController (std::shared_ptr< UIController > leftUIcontroller, std::shared_ptr< UIController > rightU←Icontroller, std::shared_ptr< FileIO > fileIO, QObject ∗parent=nullptr)

  Constructor.
- Q_INVOKABLE void savePreset (const QString &name)

  Saves UI preset based on current state of UI controllers.
- Q_INVOKABLE void loadPreset (const QString &name)

  Loads preset and sets UIController to correspoinding state.
- Q_INVOKABLE QStringListModel ∗ savedPresetsModel ()

  Returns pointer to model with names of saved data sets.

Private Member Functions

- UIControllerState readUiControllerState (std::shared_ptr< UIController > controller) const

  Reads the state of given UIController to UICotrollerState struct.
- void setUiControllerState (std::shared_ptr< UIController > controller, UIControllerState state)

  Sets the state of given UIController to given state.
- void createModels ()

  Creates models that class PresetController uses.
- void updateSavedPresetsModel ()

  Updates presetFilesModel with current files.

Private Attributes

- std::shared_ptr< UIController > **m_leftController** = nullptr
- std::shared_ptr< UIController > **m_rightController** = nullptr
- std::shared_ptr< FileIO > **m_fileIO** = nullptr
- std::unique_ptr< QStringListModel > **m_savedPresetsModel** = nullptr

3.13.1 Detailed Description

The Preset handles reading and writing state of UIControllers as presets.

3.13.2 Constructor & Destructor Documentation

3.13.2.1 PresetController()

```
PresetController::PresetController (
            std::shared_ptr< UIController > leftUIcontroller,
            std::shared_ptr< UIController > rightUIcontroller,
            std::shared_ptr< FileIO > fileIO,
            QObject * parent = nullptr )  [explicit]
```

Constructor.

Parameters

leftUIController	Shared pointer to left UIController.
rightUIController	Shared pointer to right UIController.
fileIO	Shared pointer to FileIO.
parent	Pointer to parent QObject.

3.13.3 Member Function Documentation

3.13.3.1 loadPreset()

```
void PresetController::loadPreset (
            const QString & name )
```

Loads preset and sets UIController to correspoinding state.

Parameters

name	Name of the preset to load.

3.13.3.2 readUiControllerState()

```
PresetController::UIControllerState PresetController::readUiControllerState (
            std::shared_ptr< UIController > controller ) const  [private]
```

Reads the state of given UIController to UICotrollerState struct.

Parameters

controller	Pointer to UIController.

Returns

UIControllerState: State of the given UIController.

3.13.3.3 savedPresetsModel()

```
QStringListModel * PresetController::savedPresetsModel ( )
```

Returns pointer to model with names of saved data sets.

Returns

QStringListModel: Model of saved data set names.

3.13.3.4 savePreset()

```
void PresetController::savePreset (
             const QString & name )
```

Saves UI preset based on current state of UI controllers.

Parameters

name	Name to save preset with.

3.13.3.5 setUiControllerState()

```
void PresetController::setUiControllerState (
             std::shared_ptr< UIController > controller,
             UIControllerState state ) [private]
```

Sets the state of given UIController to given state.

Parameters

controller	Pointer to UIController.
state	State to put UIController in.

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

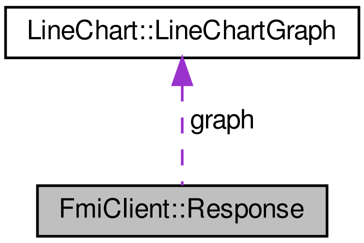
- presetcontroller.h
- presetcontroller.cpp

3.14 FmiClient::Response Struct Reference

Response format for client. Allows error messages to be passed upstream.

```
#include <fmiclient.h>
```

Collaboration diagram for FmiClient::Response:



Public Attributes

- bool **error**
- LineChart::LineChartGraph **graph**
- QString **errorMessage**

3.14.1 Detailed Description

Response format for client. Allows error messages to be passed upstream.

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

- fmiclient.h

3.15 HTTPClient::Response Struct Reference

Response format for client. Allows error state to be passed upstream.

```
#include <httpclient.h>
```

Public Attributes

- bool **error**
- QByteArray **data**

3.15.1 Detailed Description

Response format for client. Allows error state to be passed upstream.

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

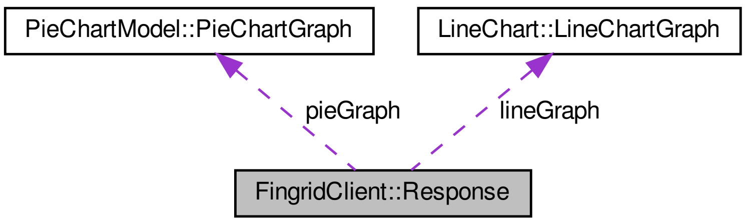
- httpclient.h

3.16 FingridClient::Response Struct Reference

Response format for client. Allows error messages to be passed upstream.

```
#include <fingridclient.h>
```

Collaboration diagram for FingridClient::Response:



Public Attributes

- bool **error** = false
- LineChart::LineChartGraph **lineGraph**
- PieChartModel::PieChartGraph **pieGraph**
- QString **errorMessage**

3.16.1 Detailed Description

Response format for client. Allows error messages to be passed upstream.

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

- fingridclient.h

3.17 PieChartModel::Slice Struct Reference

Abstraction for pie slice.

```
#include <piechartmodel.h>
```

Public Attributes

- QString **name**
- double **value**

3.17.1 Detailed Description

Abstraction for pie slice.

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

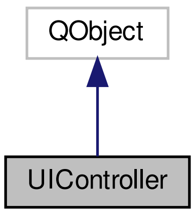
- piechartmodel.h

3.18 UIController Class Reference

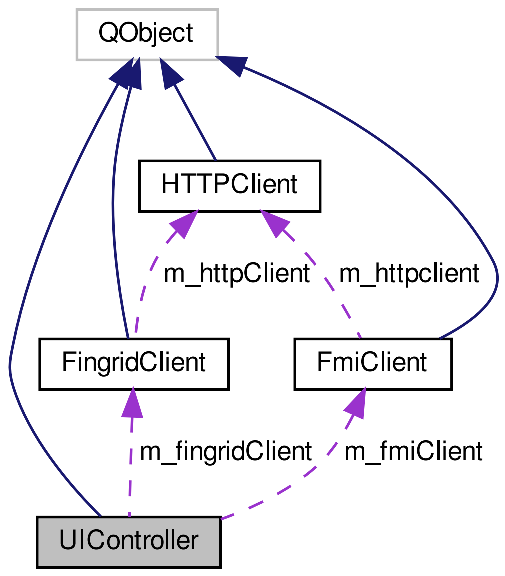
UIController tracks the state of user interface elements ans mediates the searches made by user.

```
#include <uicontroller.h>
```

Inheritance diagram for UIController:



Collaboration diagram for UIController:



Public Slots

- void resetControls ()

  Resets all ui element variables, called when chart type is changed.
- void updateLocationAutoComp ()

  Updates the autocompletion quess based on current location string.
- void updateSavedDatasetsModel ()

  Updates saved data sets model.

Signals

- void **chartTypeChanged** () const
- void **locationChanged** () const
- void **fileNameChanged** () const
- void **dateChanged** () const
- void **checkBoxesChanged** () const
- void **locationAutoCompChanged** () const
- void **busyIndicatorChanged** () const
- void **errorChanged** () const

Public Member Functions

- UIController (std::shared_ptr< FileIO > fileIO, QObject *parent=nullptr)

  Constructor.
- Q_INVOKABLE void search ()

  Queries data from clients and fills and published models based on ui state.
- Q_INVOKABLE LineChartModel * lineChartModel () const

  Returns pointer to LineChartModel this controller contontrols.
- Q_INVOKABLE PieChartModel * pieChartModel () const

  Returns pointer to PieChartModel this controller contontrols.
- Q_INVOKABLE QStringListModel * savedFilesModel ()

  Returns pointer to model with names of saved data sets.
- Q_INVOKABLE void saveDataSet (const QString &name)

  Saves currently displayed linechart with given name.

Static Public Member Functions

- static QStringListModel & chartTypesModel ()

  Returns static model with list of available chart types.

Properties

- QString **chartType**
- int **chartIndex**
- QString **location**
- QString **locationHint**
- QString **locationAutoComp**
- QString **fileName**
- int **fileIndex**
- QDate **startDate**
- QDate **endDate**
- bool **checkBox0**
- bool **checkBox1**
- bool **checkBox2**
- bool **busyIndicator**

Private Member Functions

- void createModels ()

  Creates model objects of UIController.
- void searchWeatherForecast ()

  Replaces line model with model with weather forecast.
- void searchWeatherHistory ()

  Replaces line model with model with weather history.
- void searchMonthlyTemp ()

  Replaces line model with model with monthly temperature data.
- void searchElectrProdCons ()

  Replaces line model with model with electricity troughput.
- void searchElectrForecast ()

  Replaces line model with model with electricity forecast.

- void searchRenewEnergyProd ()

  Replaces line model with model with renewable power produciton forecast.
- void searchElectricityProdMethods ()

  Replaces line model with model with electricity production method data.
- void searchElectrProdDistr ()

  Replaces piechart model with electricity production method distribution.
- void loadSavedDataSet ()

  Replaces line model with model with previously saved data set.
- void startRealTimeFrequency ()

  Starts realtime frequency monitoring.
- void startRealTimeConsumption ()

  Starts realtime consumption monitoring.
- void startRealTimeImport ()

  Starts realtime import/export monitoring.
- void showBusyIndicator (bool show)

  Sets UIController in busy state.
- void updateDictionary (const QString &location)

  If autocomplete came up empty but model was populated, location is added to auto complete suggestions.
- int chartIndex () const

  Returns the index of current chartType.
- int fileIndex () const

  Returns the index of currently selected saved data set.

Private Attributes

- std::shared_ptr< FileIO > **m_fileIO** = nullptr
- std::unique_ptr< LineChartModel > **m_lineModel** = nullptr
- std::unique_ptr< PieChartModel > **m_pieModel** = nullptr
- std::unique_ptr< QStringListModel > **m_savedDatasetsModel** = nullptr
- QString **m_chartType** = m_chartTypes[0]
- QString **m_location**
- QString **m_fileName**
- QString **m_locationHint** = "Location"
- QDate **m_startDate** = QDate::currentDate()
- QDate **m_endDate** = QDate::currentDate()
- unsigned int **m_month** = QDate::currentDate().month()
- unsigned int **m_year** = QDate::currentDate().year()
- bool **m_checkBox0** = true
- bool **m_checkBox1** = true
- bool **m_checkBox2** = true
- bool **m_busyIndicator** = false
- FmiClient **m_fmiClient**
- FingridClient **m_fingridClient**
- std::unique_ptr< AutoCompleter > **m_autoCompleter** = nullptr
- QString **m_finnishPlaceNamesDict** = "data/fmiLocationDictionary.txt"
- QString **m_locationAutoComp** = ""
- QTimer **m_realTimeTimer**

Static Private Attributes

- static QStringList **m_chartTypes**
- static QStringListModel **m_chartTypesModel** = QStringListModel(m_chartTypes)
- static int **m_realTimeInterval_ms** = 30000

Friends

- class **PresetController**

3.18.1 Detailed Description

UIController tracks the state of user interface elements ans mediates the searches made by user.

3.18.2 Constructor & Destructor Documentation

3.18.2.1 UIController()

```
UIController::UIController (
            std::shared_ptr< FileIO > fileIO,
            QObject * parent = nullptr )  [explicit]
```

Constructor.

Parameters

fileIO	Shared Pointer to FileIO.
parent	Pointer to parent QObject.

3.18.3 Member Function Documentation

3.18.3.1 chartIndex()

```
int UIController::chartIndex ( ) const  [private]
```

Returns the index of current chartType.

Returns

int: Index of current chartType.

3.18.3.2 chartTypesModel()

```
QStringListModel & UIController::chartTypesModel ( )  [static]
```

Returns static model with list of available chart types.

Returns

QStringListModel&: Model with names of available chart types.

3.18.3.3 fileIndex()

```
int UIController::fileIndex ( ) const  [private]
```

Returns the index of currently selected saved data set.

Returns

Qint: Index of currently selected saved data set.

3.18.3.4 lineChartModel()

```
LineChartModel * UIController::lineChartModel ( ) const
```

Returns pointer to LineChartModel this controller contontrols.

Returns

LineChartModel∗: Pointer to LineChartModel.

3.18.3.5 pieChartModel()

```
PieChartModel * UIController::pieChartModel ( ) const
```

Returns pointer to PieChartModel this controller contontrols.

Returns

PieChartModel∗: Pointer to PieChartModel.

3.18.3.6 saveDataSet()

```
void UIController::saveDataSet (
             const QString & name )
```

Saves currently displayed linechart with given name.

Parameters

name	Name of the data set.

3.18.3.7 savedFilesModel()

```
QStringListModel * UIController::savedFilesModel ( )
```

Returns pointer to model with names of saved data sets.

Returns

QStringListModel∗: Pointer to model with saved dataset names.

3.18.3.8 showBusyIndicator()

```
void UIController::showBusyIndicator (
             bool show )  [private]
```

Sets UIController in busy state.

Parameters

show	Should busy indicator be shown.

3.18.3.9 updateDictionary()

```
void UIController::updateDictionary (
             const QString & location )  [private]
```

If autocomplete came up empty but model was populated, location is added to auto complete suggestions.

Parameters

3.18.4 Member Data Documentation

3.18.4.1 m_chartTypes

```
QStringList UIController::m_chartTypes  [inline], [static], [private]
```

Initial value:

```
= {"Weather Forecast 24h",
            "Weather History",
            "Monthly Temperature Averages",
            "Electricity Production And Consumption",
            "Electricity Production And Consumption Forecast 24h",
            "Renewable Energy Production Forecast 24h",
            "Electricity Production Methods",
            "Electricity Production Method Distribution",
            "Saved Datasets",
            "Real Time Frequency",
            "Real Time Consumption",
            "Real Time Import/Export"}
```

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

- uicontroller.h
- uicontroller.cpp

3.19 PresetController::UIControllerState Struct Reference

Abstraction for the state of the user interface state defined by UIController.

```
#include <presetcontroller.h>
```

Public Attributes

- QString **chartType**
- QString **location**
- QString **fileName**
- bool **checkBox0**
- bool **checkBox1**
- bool **checkBox2**

3.19.1 Detailed Description

Abstraction for the state of the user interface state defined by UIController.

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

- presetcontroller.h

Index