

# Weather station client

1.0

Generated by Doxygen 1.8.20



<b>1 Module Index</b>	<b>1</b>
1.1 Modules	1
<b>2 Namespace Index</b>	<b>3</b>
2.1 Namespace List	3
<b>3 Hierarchical Index</b>	<b>5</b>
3.1 Class Hierarchy	5
<b>4 Class Index</b>	<b>7</b>
4.1 Class List	7
<b>5 File Index</b>	<b>9</b>
5.1 File List	9
<b>6 Module Documentation</b>	<b>11</b>
6.1 Ui	11
6.1.1 Detailed Description	11
<b>7 Namespace Documentation</b>	<b>13</b>
7.1 Ui Namespace Reference	13
7.1.1 Detailed Description	13
<b>8 Class Documentation</b>	<b>15</b>
8.1 DBase Class Reference	15
8.1.1 Detailed Description	16
8.1.2 Constructor & Destructor Documentation	16
8.1.2.1 DBase()	16
8.1.2.2 ~DBase()	17
8.1.3 Member Function Documentation	17
8.1.3.1 fRand()	17
8.1.3.2 generateSampleData()	17
8.1.3.3 getRecentMeasurements()	18
8.1.3.4 isOpen()	18
8.1.3.5 round()	18
8.1.4 Member Data Documentation	19
8.1.4.1 measurements	19
8.1.4.2 page	19
8.2 MainWindow Class Reference	20
8.2.1 Detailed Description	22
8.2.2 Constructor & Destructor Documentation	22
8.2.2.1 MainWindow()	22
8.2.2.2 ~MainWindow()	22
8.2.3 Member Function Documentation	22
8.2.3.1 addItemToGridLayout()	22

8.2.3.2 generateSampleData()	23
8.2.3.3 getDate()	23
8.2.3.4 nextButton	23
8.2.3.5 prevButton	24
8.2.3.6 refreshButton	24
8.2.3.7 removeItemFromGridLayout()	24
8.2.3.8 setDate()	24
8.2.3.9 setMeasurements()	25
8.2.3.10 setupButton()	25
8.2.3.11 setupChart()	26
8.2.3.12 setupDatabase()	26
8.2.3.13 setupWindow()	27
8.2.3.14 updateButtons()	27
8.2.3.15 updateChart()	27
8.2.4 Member Data Documentation	28
8.2.4.1 dateInputField	28
8.2.4.2 dbase	28
8.2.4.3 ui	28
8.3 DBase::measurement Struct Reference	28
8.4 WChart Class Reference	29
8.4.1 Detailed Description	30
8.4.2 Constructor & Destructor Documentation	31
8.4.2.1 WChart()	31
8.4.2.2 ~WChart()	31
8.4.3 Member Function Documentation	31
8.4.3.1 getMaxRangeYAxis()	32
8.4.3.2 getMinRangeYAxis()	32
8.4.3.3 getTitleChart()	32
8.4.3.4 getTitleXAxis()	32
8.4.3.5 getTitleYAxis()	33
8.4.3.6 getType()	33
8.4.3.7 render()	33
8.4.3.8 setAxesTickCount()	33
8.4.3.9 setData()	34
8.4.3.10 setRangeYAxis()	34
8.4.3.11 setType()	34
<b>9 File Documentation</b>	<b>37</b>
9.1 weerstation/dbase.h File Reference	37
9.1.1 Detailed Description	38
9.2 weerstation/mainwindow.h File Reference	38
9.2.1 Detailed Description	39

---

9.3 weerstation/wchart.h File Reference . . . . .	39
9.3.1 Detailed Description . . . . .	39
<b>Index</b>	<b>41</b>



# Chapter 1

## Module Index

### 1.1 Modules

Here is a list of all modules:

Ui .....	11
----------	----





## Chapter 2

# Namespace Index

### 2.1 Namespace List

Here is a list of all documented namespaces with brief descriptions:

<a href="#">Ui</a>	Used to differentiate between the ui class from the designer and the class that implements the functionality . . . . .	<a href="#">13</a>
--------------------	--	--------------------



## Chapter 3

# Hierarchical Index

### 3.1 Class Hierarchy

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

DBase . . . . .	15
DBase::measurement . . . . .	28
QMainWindow	
MainWindow . . . . .	20
QWidget	
WChart . . . . .	29



## Chapter 4

# Class Index

### 4.1 Class List

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

<a href="#">DBase</a>	The data layer class . . . . .	15
<a href="#">MainWindow</a>	The presentation layer class . . . . .	20
<a href="#">DBase::measurement</a>	. . . . .	28
<a href="#">WChart</a>	The logic layer class . . . . .	29



## Chapter 5

# File Index

### 5.1 File List

Here is a list of all documented files with brief descriptions:

weerstation/ <a href="#">dbase.h</a>	
The data layer . . . . .	37
weerstation/ <a href="#">mainwindow.h</a>	
The presentation layer . . . . .	38
weerstation/ <a href="#">wchart.h</a>	
The logic layer . . . . .	39





## Chapter 6

# Module Documentation

### 6.1 Ui

#### Namespaces

- [Ui](#)

*Used to differentiate between the ui class from the designer and the class that implements the functionality.*

#### 6.1.1 Detailed Description



## Chapter 7

# Namespace Documentation

### 7.1 Ui Namespace Reference

Used to differentiate between the ui class from the designer and the class that implements the functionality.

#### 7.1.1 Detailed Description

Used to differentiate between the ui class from the designer and the class that implements the functionality.



## Chapter 8

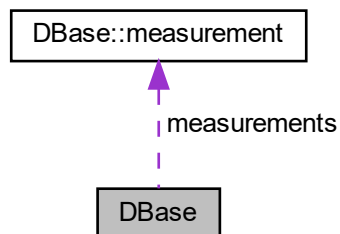
# Class Documentation

### 8.1 DBase Class Reference

The data layer class.

```
#include "dbase.h"
```

Collaboration diagram for DBase:



#### Classes

- struct [measurement](#)

#### Public Member Functions

- [DBase](#) (const QString &hostname, const QString &username, const QString &password, const QString &name)  
*Creates the database connection.*
- [~DBase](#) ()  
*Destroys the database connection.*
- bool [isOpen](#) () const

- *Checks if the connection is made with the database server.*
- void [getRecentMeasurements](#) (const QString &filter, const QString &date)  
*Gets the measurements from the database based on the type and date values.*
- void [generateSampleData](#) (unsigned amount)  
*Generates fake sample data to work with some data.*
- double [fRand](#) (double fMin, double fMax)  
*Generates random value between fMin and fMax.*
- double [round](#) (double var)  
*Round-up a double value.*

## Public Attributes

- struct [DBase::measurement measurements](#) [60]  
*Holds 60 samples of measurement data.*
- int [totalMeasurements](#)  
*Holds the data of the total amount of measurements that is currently available.*
- int [page](#)  
*Holds the current page the view layer has to show.*
- int [range](#)  
*As the data shrinks less measurements are available. The page calculates how many measurements on the page are allowed to show.*

## Private Attributes

- QSqlDatabase [m\\_db](#)  
*The database variable which deals with the database server.*

### 8.1.1 Detailed Description

The data layer class.

In the data layer every aspect of communication with the database is managed by this class.

### 8.1.2 Constructor & Destructor Documentation

#### 8.1.2.1 DBase()

```
DBase::DBase (
    const QString & hostname,
    const QString & username,
    const QString & password,
    const QString & name )
```

Creates the database connection.

The constructor.

Returns

void

### 8.1.2.2 ~DBase()

```
DBase::~~DBase ( )
```

Destroys the database connection.

The destructor.

#### Returns

void

## 8.1.3 Member Function Documentation

### 8.1.3.1 fRand()

```
double DBase::fRand (
    double fMin,
    double fMax )
```

Generates random value between fMin and fMax.

#### Warning

Only for testing purposes.

#### Precondition

Date must be a valid date in the format of for e.g.: 2020-10-25.

#### Parameters

<i>fMin</i>	
<i>fMax</i>	

#### Returns

void

### 8.1.3.2 generateSampleData()

```
void DBase::generateSampleData (
    unsigned amount )
```

Generates fake sample data to work with some data.

**Warning**

Only for testing purposes.

**Precondition**

Date must be a valid date in the format ex: 2020-10-25.

**Parameters**

<i>amount</i>	
---------------	--

**Returns**

void

**8.1.3.3 getRecentMeasurements()**

```
void DBase::getRecentMeasurements (
    const QString & filter,
    const QString & date )
```

Gets the measurements from the database based on the type and date values.

**Returns**

void

**8.1.3.4 isOpen()**

```
bool DBase::isOpen ( ) const
```

Checks if the connection is made with the database server.

**Returns**

boolean

**8.1.3.5 round()**

```
double DBase::round (
    double var )
```

Round-up a double value.



### Parameters

<i>var</i>	
------------	--

### Returns

double

### Example:

```
17.66666 * 100 = 1766.66  
1766.66 + .5 = 1767.16 for rounding off value  
then type cast to int so value is 1767  
then divided by 100 so the value converted into 17.67
```

## 8.1.4 Member Data Documentation

### 8.1.4.1 measurements

```
struct measurement QString type value date DBase::measurements[60]
```

Holds 60 samples of measurement data.

The data is stored in this array before it is passed down to the chart widget.

### 8.1.4.2 page

```
int DBase::page
```

Holds the current page the view layer has to show.

The page indicates that 60 measurements may be shown by the view layer.

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

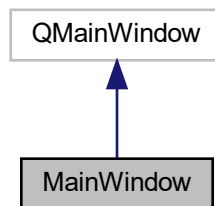
- [weerstation/dbase.h](#)
- [weerstation/dbase.cpp](#)

## 8.2 MainWindow Class Reference

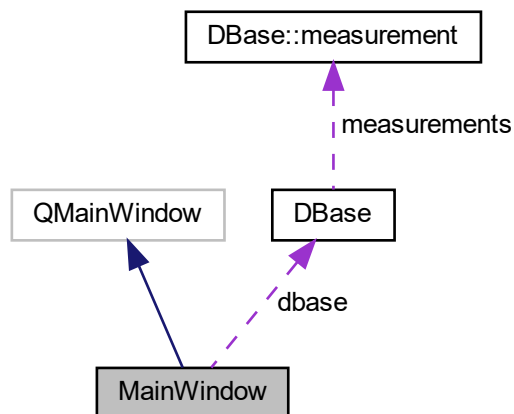
The presentation layer class.

```
#include "mainwindow.h"
```

Inheritance diagram for MainWindow:



Collaboration diagram for MainWindow:



### Public Member Functions

- [MainWindow](#) (`QWidget *parent=nullptr`)  
*Creates the main window.*
- [~MainWindow](#) ()  
*Destroys the main window.*

## Private Slots

- void [refreshButton](#) ()  
*Updates the charts, buttons and the gridlayout based on the page value.*
- void [prevButton](#) ()  
*Updates the charts, buttons and the gridlayout based on the page value.*
- void [nextButton](#) ()  
*Updates the charts, buttons and the gridlayout based on the page value.*

## Private Member Functions

- const QString [getDate](#) ()  
*Gets the date from the dateInputField field.*
- void [setDate](#) (const QString &date)  
*Sets the date from a given string.*
- void [setupDatabase](#) (const QString &hostname, const QString &username, const QString &password, const QString &name)  
*Initializes a database connection.*
- void [generateSampleData](#) (unsigned amount)  
*Generates fake sample data for the measurement tbl.*
- void [setMeasurements](#) (WChart &chart, const QString &date)  
*Gets data from the measurement tbl and puts the data into the chart.*
- void [setupChart](#) (const QString &titleChart, const QString &type, const QString &titleXAxis, const QString &titleYAxis, qint16 minRangeY, qint16 maxRangeY)  
*Initializes a database connection.*
- void [updateChart](#) (WChart \*chart)  
*Updates the chart with new data and renews the gridlayout.*
- void [setupButton](#) (const QString &name, const char \*slot)  
*Creates a button with an event trigger.*
- void [updateButtons](#) ()  
*Updates the buttons depending on some states.*
- void [addItemToGridLayout](#) (QWidget \*item, unsigned posX, unsigned posY)  
*Adds a new widget to the grid layout.*
- void [removeItemFromGridLayout](#) (unsigned posX, unsigned posY)  
*Removes a widget from the grid layout based on the row and column of that item.*
- void [setupWindow](#) (const QString &>windowTitle, QWidget \*centralWidget)  
*Sets a window title and sets the central widget of the main window.*

## Private Attributes

- QString [date](#)  
*Stores the current date or date that is coming from the input field.*
- QLineEdit \* [dateInputField](#)  
*The widget which behaves as a user input field.*
- QVector< WChart \* > [charts](#)  
*A vector type data stores the view of all charts and manages the life-cycle.*
- QVector< QPushButton \* > [buttons](#)  
*A vector type data stores all buttons present and manages the life-cycle.*
- DBase \* [dbase](#)  
*The database class instance.*
- Ui::MainWindow \* [ui](#)  
*The [MainWindow](#) class is used to store the mainwindow.ui class generated by the Qt.*

## 8.2.1 Detailed Description

The presentation layer class.

In the presentation layer every aspect of the lifecycle of a widget is managed by this class.

## 8.2.2 Constructor & Destructor Documentation

### 8.2.2.1 MainWindow()

```
MainWindow::MainWindow (
    QWidget * parent = nullptr )
```

Creates the main window.

The constructor.

#### Returns

void

### 8.2.2.2 ~MainWindow()

```
MainWindow::~MainWindow ( )
```

Destroys the main window.

The destructor.

#### Returns

void

## 8.2.3 Member Function Documentation

### 8.2.3.1 addItemToGridLayout()

```
void MainWindow::addItemToGridLayout (
    QWidget * item,
    unsigned posX,
    unsigned posY ) [private]
```

Adds a new widget to the grid layout.

**Parameters**

<i>item</i>	
<i>posX</i>	
<i>posY</i>	

**Returns**

void

**8.2.3.2 generateSampleData()**

```
void MainWindow::generateSampleData (
    unsigned amount ) [private]
```

Generates fake sample data for the measurement tbl.

**Parameters**

<i>amount</i>	
---------------	--

**Returns**

void

**8.2.3.3 getDate()**

```
const QString MainWindow::getDate ( ) [private]
```

Gets the date from the dateInputField field.

**Returns**

const QString

**8.2.3.4 nextButton**

```
void MainWindow::nextButton ( ) [private], [slot]
```

Updates the charts, buttons and the gridlayout based on the page value.

**Returns**

void

### 8.2.3.5 prevButton

```
void MainWindow::prevButton ( ) [private], [slot]
```

Updates the charts, buttons and the gridlayout based on the page value.

#### Returns

void

### 8.2.3.6 refreshButton

```
MainWindow::refreshButton ( ) [private], [slot]
```

Updates the charts, buttons and the gridlayout based on the page value.

#### Returns

void

### 8.2.3.7 removeItemFromGridLayout()

```
void MainWindow::removeItemFromGridLayout (
    unsigned posX,
    unsigned posY ) [private]
```

Removes a widget from the grid layout based on the row and column of that item.

#### Parameters

<i>posX</i>	
<i>posY</i>	

#### Returns

void

### 8.2.3.8 setDate()

```
void MainWindow::setDate (
    const QString & date ) [private]
```

Sets the date from a given string.

**Parameters**

<i>date</i>	
-------------	--

**Returns**

void

**8.2.3.9 setMeasurements()**

```
void MainWindow::setMeasurements (
    WChart & chart,
    const QString & date ) [private]
```

Gets data from the measurement tbl and puts the data into the chart.

**Precondition**

The value of amount must not be negative

**Parameters**

<i>chart</i>	
<i>date</i>	

**Returns**

void

**8.2.3.10 setupButton()**

```
void MainWindow::setupButton (
    const QString & name,
    const char * slot ) [private]
```

Creates a button with an event trigger.

**Parameters**

<i>name</i>	
<i>slot</i>	

**Returns**

void

**8.2.3.11 setupChart()**

```
void MainWindow::setupChart (
    const QString & titleChart,
    const QString & type,
    const QString & titleXAxis,
    const QString & titleYAxis,
    quint16 minRangeY,
    quint16 maxRangeY ) [private]
```

Initializes a database connection.

**Parameters**

<i>titleChart</i>	
<i>type</i>	
<i>titleXAxis</i>	
<i>titleYAxis</i>	
<i>minRangeY</i>	
<i>maxRangeY</i>	

**Returns**

void

**8.2.3.12 setupDatabase()**

```
void MainWindow::setupDatabase (
    const QString & hostname,
    const QString & username,
    const QString & password,
    const QString & name ) [private]
```

Initializes a database connection.

**Parameters**

<i>hostname</i>	
<i>username</i>	
<i>password</i>	
<i>name</i>	



**Returns**

void

**8.2.3.13 setupWindow()**

```
void MainWindow::setupWindow (
    const QString & windowTitle,
    QWidget * centralWidget ) [private]
```

Sets a window title and sets the central widget of the main window.

**Parameters**

<i>windowTitle</i>	
<i>centralWidget</i>	

**Returns**

void

**8.2.3.14 updateButtons()**

```
void MainWindow::updateButtons ( ) [private]
```

Updates the buttons depending on some states.

**Returns**

void

**8.2.3.15 updateChart()**

```
void MainWindow::updateChart (
    WChart * chart ) [private]
```

Updates the chart with new data and renews the gridlayout.

**Parameters**

<i>chart</i>	
--------------	--

#### Returns

void

## 8.2.4 Member Data Documentation

### 8.2.4.1 dateInputField

```
QLineEdit * MainWindow::dateInputField [private]
```

The widget which behaves as a user input field.

The user input field where a date could be specified.

### 8.2.4.2 dbase

```
DBase * MainWindow::dbase [private]
```

The database class instance.

Manages all income/outcome traffic with the database server.

### 8.2.4.3 ui

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

The [MainWindow](#) class is used to store the mainwindow.ui class generated by the Qt.

Manages the design view of the application window.

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

- weestation/[mainwindow.h](#)
- weestation/mainwindow.cpp

## 8.3 DBase::measurement Struct Reference

### Public Attributes

- QString [type](#)  
*Stores the type of the measurement data.*
- QString [value](#)  
*Stores the value of the measurement data.*
- QString [date](#)  
*Stores the date of the measurement data.*

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

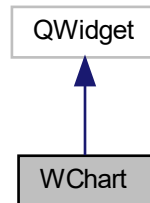
- weestation/[dbase.h](#)

## 8.4 WChart Class Reference

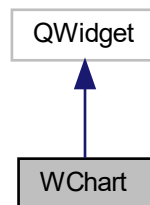
The logic layer class.

```
#include "wchart.h"
```

Inheritance diagram for WChart:



Collaboration diagram for WChart:



### Public Member Functions

- [WChart](#) (QWidget \*parent=nullptr, const QString &[titleChart](#)="", const QString &[titleXAxis](#)="", const QString &[titleYAxis](#)="")  
*Creates a basic chart.*
- [~WChart](#) ()  
*Destroys the chart widget.*
- void [render](#) ()  
*Renders the widget.*
- void [setRangeYAxis](#) (unsigned min, unsigned max)  
*Sets the range of Y-axis.*
- void [setAxesTickCount](#) (unsigned amount)  
*Sets the amount of steps in the x-axis and y-axis.*
- void [setData](#) (const qreal &x, const qreal &y)

- Sets the data points on the graph.*
- void [setType](#) (const QString &[type](#))  
*Sets the type.*
- const QString & [getType](#) ()  
*Gets the type.*
- const QString & [getTitleChart](#) ()  
*Gets the main title.*
- const QString & [getTitleXAxis](#) ()  
*Gets the title of x-axis.*
- const QString & [getTitleYAxis](#) ()  
*Gets the title of y-axis.*
- const qint16 & [getMinRangeYAxis](#) ()  
*Gets the starting range of the y-axis.*
- const qint16 & [getMaxRangeYAxis](#) ()  
*Gets the ending range of the y-axis.*

## Public Attributes

- QtCharts::QChartView \* [chartView](#)  
*The layer of the widget that is going to be rendered on the screen.*

## Private Attributes

- QtCharts::QValueAxis \* [valueAxisY](#)  
*The layer that is responsible for showing the data on the Y-axis.*
- QtCharts::QDateTimeAxis \* [valueAxisX](#)  
*The layer that is responsible for showing the data on the X-axis.*
- QtCharts::QChart \* [chart](#)  
*The chart stores and keeps track of all data.*
- QtCharts::QLineSeries \* [series](#)  
*Series stores all data points that is going to be presented by valueAxisX and valueAxisY.*
- QString [type](#)  
*Stores the type.*
- QString [titleChart](#)  
*Stores the main title.*
- QString [titleXAxis](#)  
*Stores the title of the x-axis.*
- QString [titleYAxis](#)  
*Stores the title of the y-axis.*
- qint16 [minRange](#)  
*Stores the minimum range of the y-axis.*
- qint16 [maxRange](#)  
*Stores the maximum range of the y-axis.*

### 8.4.1 Detailed Description

The logic layer class.

In the logic layer all logic of the created charts and life-cycle are managed by this class.

## 8.4.2 Constructor & Destructor Documentation

### 8.4.2.1 WChart()

```
WChart::WChart (
    QWidget * parent = nullptr,
    const QString & titleChart = "",
    const QString & titleXAxis = "",
    const QString & titleYAxis = "" ) [explicit]
```

Creates a basic chart.

The constructor.

#### Parameters

<i>parent</i>	
<i>titleChart</i>	
<i>titleXAxis</i>	
<i>titleYAxis</i>	

#### Returns

void

### 8.4.2.2 ~WChart()

```
WChart::~WChart ( )
```

Destroys the chart widget.

The destructor.

#### Returns

void

## 8.4.3 Member Function Documentation

#### 8.4.3.1 getMaxRangeYAxis()

```
const qint16 & WChart::getMaxRangeYAxis ( )
```

Gets the ending range of the y-axis.

##### Returns

const qint16&

#### 8.4.3.2 getMinRangeYAxis()

```
const qint16 & WChart::getMinRangeYAxis ( )
```

Gets the starting range of the y-axis.

##### Returns

const qint16&

#### 8.4.3.3 getTitleChart()

```
const QString & WChart::getTitleChart ( )
```

Gets the main title.

##### Returns

const QString&

#### 8.4.3.4 getTitleXAxis()

```
const QString & WChart::getTitleXAxis ( )
```

Gets the title of x-axis.

##### Returns

const QString&

#### 8.4.3.5 getTitleYAxis()

```
const QString & WChart::getTitleYAxis ( )
```

Gets the title of y-axis.

##### Returns

const QString&

#### 8.4.3.6 getType()

```
const QString & WChart::getType ( )
```

Gets the type.

##### Returns

const QString&

#### 8.4.3.7 render()

```
void WChart::render ( )
```

Renders the widget.

All additional part of the widget like updates are managed my this method.

##### Returns

void

#### 8.4.3.8 setAxesTickCount()

```
void WChart::setAxesTickCount (
    unsigned amount )
```

Sets the amount of steps in the x-axis and y-axis.

##### Returns

void

#### 8.4.3.9 setData()

```
void WChart::setData (
    const qreal & x,
    const qreal & y )
```

Sets the data points on the graph.

##### Returns

void

#### 8.4.3.10 setRangeYAxis()

```
void WChart::setRangeYAxis (
    unsigned min,
    unsigned max )
```

Sets the range of Y-axis.

##### Returns

void

#### 8.4.3.11 setType()

```
void WChart::setType (
    const QString & type )
```

Sets the type.

This type is important because the method `getRecentMeasurements(const QString& filter, const QString& date)` database needs this as a filter input to get the corresponding data.

##### Note

Needs a valid type which occurs in the table of the database in order to retrieve data from the database.

##### Parameters

<i>type</i>	
-------------	--

##### Returns

void



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

- [weerstation/wchart.h](#)
- [weerstation/wchart.cpp](#)



## Chapter 9

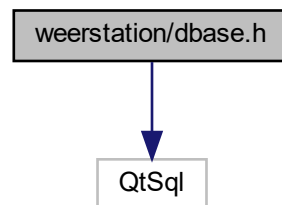
# File Documentation

### 9.1 weerstation/dbase.h File Reference

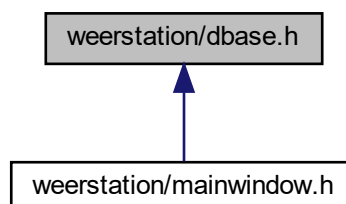
The data layer.

```
#include <QtSql>
```

Include dependency graph for dbase.h:



This graph shows which files directly or indirectly include this file:



## Classes

- class [DBase](#)  
*The data layer class.*
- struct [DBase::measurement](#)

### 9.1.1 Detailed Description

The data layer.

#### Author

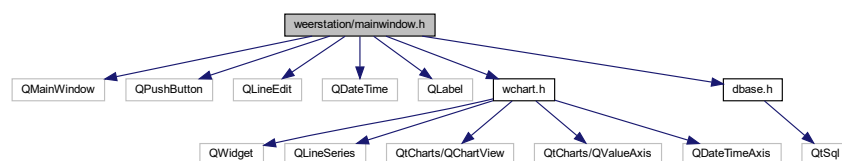
Bedirhan Dincer

## 9.2 weerstation/mainwindow.h File Reference

The presentation layer.

```
#include <QMainWindow>
#include <QPushButton>
#include <QLineEdit>
#include <QDateTime>
#include <QLabel>
#include <wchart.h>
#include <dbase.h>
```

Include dependency graph for mainwindow.h:



## Classes

- class [MainWindow](#)  
*The presentation layer class.*

## Namespaces

- [Ui](#)  
*Used to differentiate between the ui class from the designer and the class that implements the functionality.*

### 9.2.1 Detailed Description

The presentation layer.

Author

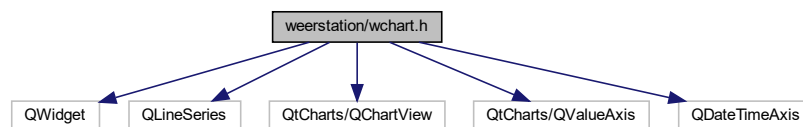
Bedirhan Dincer

## 9.3 weerstation/wchart.h File Reference

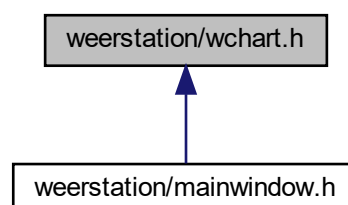
The logic layer.

```
#include <QWidget>
#include <QLineSeries>
#include <QtCharts/QChartView>
#include <QtCharts/QValueAxis>
#include <QDateTimeAxis>
```

Include dependency graph for wchart.h:



This graph shows which files directly or indirectly include this file:



### Classes

- class [WChart](#)  
*The logic layer class.*

### 9.3.1 Detailed Description

The logic layer.

Author

Bedirhan Dincer



# Index

- ~DBase
  - DBase, [16](#)
- ~MainWindow
  - MainWindow, [22](#)
- ~WChart
  - WChart, [31](#)
- addItemToGridLayout
  - MainWindow, [22](#)
- dateInputField
  - MainWindow, [28](#)
- DBase, [15](#)
  - ~DBase, [16](#)
  - DBase, [16](#)
  - fRand, [17](#)
  - generateSampleData, [17](#)
  - getRecentMeasurements, [18](#)
  - isOpen, [18](#)
  - measurements, [19](#)
  - page, [19](#)
  - round, [18](#)
- dbase
  - MainWindow, [28](#)
- DBase::measurement, [28](#)
- fRand
  - DBase, [17](#)
- generateSampleData
  - DBase, [17](#)
  - MainWindow, [23](#)
- getDate
  - MainWindow, [23](#)
- getMaxRangeYAxis
  - WChart, [31](#)
- getMinRangeYAxis
  - WChart, [32](#)
- getRecentMeasurements
  - DBase, [18](#)
- getTitleChart
  - WChart, [32](#)
- getTitleXAxis
  - WChart, [32](#)
- getTitleYAxis
  - WChart, [32](#)
- getType
  - WChart, [33](#)
- isOpen
  - DBase, [18](#)
- MainWindow, [20](#)
  - ~MainWindow, [22](#)
  - addItemToGridLayout, [22](#)
  - dateInputField, [28](#)
  - dbase, [28](#)
  - generateSampleData, [23](#)
  - getDate, [23](#)
  - MainWindow, [22](#)
  - nextButton, [23](#)
  - prevButton, [23](#)
  - refreshButton, [24](#)
  - removeItemFromGridLayout, [24](#)
  - setDate, [24](#)
  - setMeasurements, [25](#)
  - setupButton, [25](#)
  - setupChart, [26](#)
  - setupDatabase, [26](#)
  - setupWindow, [27](#)
  - ui, [28](#)
  - updateButtons, [27](#)
  - updateChart, [27](#)
- measurements
  - DBase, [19](#)
- nextButton
  - MainWindow, [23](#)
- page
  - DBase, [19](#)
- prevButton
  - MainWindow, [23](#)
- refreshButton
  - MainWindow, [24](#)
- removeItemFromGridLayout
  - MainWindow, [24](#)
- render
  - WChart, [33](#)
- round
  - DBase, [18](#)
- setAxesTickCount
  - WChart, [33](#)
- setData
  - WChart, [33](#)
- setDate
  - MainWindow, [24](#)
- setMeasurements
  - MainWindow, [25](#)
- setRangeYAxis

- WChart, [34](#)
- setType
  - WChart, [34](#)
- setupButton
  - MainWindow, [25](#)
- setupChart
  - MainWindow, [26](#)
- setupDatabase
  - MainWindow, [26](#)
- setupWindow
  - MainWindow, [27](#)
- Ui, [11](#), [13](#)
- ui
  - MainWindow, [28](#)
- updateButtons
  - MainWindow, [27](#)
- updateChart
  - MainWindow, [27](#)
- WChart, [29](#)
  - ~WChart, [31](#)
  - getMaxRangeYAxis, [31](#)
  - getMinRangeYAxis, [32](#)
  - getTitleChart, [32](#)
  - getTitleXAxis, [32](#)
  - getTitleYAxis, [32](#)
  - getType, [33](#)
  - render, [33](#)
  - setAxesTickCount, [33](#)
  - setData, [33](#)
  - setRangeYAxis, [34](#)
  - setType, [34](#)
  - WChart, [31](#)
- weerstation/dbase.h, [37](#)
- weerstation/mainwindow.h, [38](#)
- weerstation/wchart.h, [39](#)