

# KD Gantt 2 Reference Manual

[rev.2.0]

Generated by Doxygen 1.5.1

Tue Nov 27 14:29:14 2007



# Contents



# Chapter 1

## KD Gantt 2 Main Page

Welcome to the [KDGantt](#) API reference documentation. Good places to start are [KDGantt::View](#) and [KDGantt::GraphicsView](#)



## Chapter 2

# KD Gantt 2 Module Index

### 2.1 KD Gantt 2 Modules

Here is a list of all modules:

KDGantt . . . . . ??



## Chapter 3

# KD Gantt 2 Directory Hierarchy

### 3.1 KD Gantt 2 Directories

This directory hierarchy is sorted roughly, but not completely, alphabetically:

src . . . . .	??
unittest . . . . .	??



# Chapter 4

## KD Gantt 2 Namespace Index

### 4.1 KD Gantt 2 Namespace List

Here is a list of all namespaces with brief descriptions:

<a href="#">KDAB</a> .....	??
<a href="#">KDAB::UnitTest</a> .....	??
<a href="#">KDGantt</a> .....	??



# Chapter 5

## KD Gantt 2 Hierarchical Index

### 5.1 KD Gantt 2 Class Hierarchy

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

KDAB::UnitTest::Runner . . . . .	??
KDAB::UnitTest::Test . . . . .	??
KDAB::UnitTest::TestFactory . . . . .	??
KDAB::UnitTest::GenericFactory< T_Test > . . . . .	??
KDAB::UnitTest::TestRegistry . . . . .	??
KDGantt::AbstractRowController . . . . .	??
KDGantt::ListViewRowController . . . . .	??
KDGantt::TreeViewRowController . . . . .	??
KDGantt::Constraint . . . . .	??
KDGantt::DateTimeSpan . . . . .	??
KDGantt::Span . . . . .	??
QAbstractItemView . . . . .	??
KDGantt::Legend . . . . .	??
QAbstractProxyModel . . . . .	??
KDGantt::ForwardingProxyModel . . . . .	??
KDGantt::ProxyModel . . . . .	??
KDGantt::SummaryHandlingProxyModel . . . . .	??
QGraphicsItem . . . . .	??
KDGantt::ConstraintGraphicsItem . . . . .	??
KDGantt::GraphicsItem . . . . .	??
QGraphicsScene . . . . .	??
KDGantt::GraphicsScene . . . . .	??
QGraphicsView . . . . .	??
KDGantt::GraphicsView . . . . .	??
QItemDelegate . . . . .	??
KDGantt::ItemDelegate . . . . .	??
QObject . . . . .	??
KDGantt::AbstractGrid . . . . .	??
KDGantt::DateTimeGrid . . . . .	??
KDGantt::ConstraintModel . . . . .	??

KDGantt::ConstraintProxy . . . . .	??
QStyleOptionViewItem . . . . .	??
KDGantt::StyleOptionGanttItem . . . . .	??
QWidget . . . . .	??
KDGantt::View . . . . .	??

# Chapter 6

## KD Gantt 2 Class Index

### 6.1 KD Gantt 2 Class List

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

<a href="#">KDAB::UnitTest::GenericFactory&lt; T_Test &gt;</a>	??
<a href="#">KDAB::UnitTest::Runner</a>	??
<a href="#">KDAB::UnitTest::Test</a>	??
<a href="#">KDAB::UnitTest::TestFactory</a>	??
<a href="#">KDAB::UnitTest::TestRegistry</a>	??
<a href="#">KDGantt::AbstractGrid</a> (Abstract baseclass for grids. A grid is used to convert between QModelIndex'es and gantt chart values (doubles) and to paint the background and header of the view )	??
<a href="#">KDGantt::AbstractRowController</a> (Abstract baseclass for row controllers. A row controller is used by the <a href="#">GraphicsView</a> to navigate the model and to determine the row geometries )	??
<a href="#">KDGantt::Constraint</a> (A class used to represent a dependency )	??
<a href="#">KDGantt::ConstraintGraphicsItem</a>	??
<a href="#">KDGantt::ConstraintModel</a>	??
<a href="#">KDGantt::ConstraintProxy</a>	??
<a href="#">KDGantt::DateTimeGrid</a>	??
<a href="#">KDGantt::DateTimeSpan</a>	??
<a href="#">KDGantt::ForwardingProxyModel</a>	??
<a href="#">KDGantt::GraphicsItem</a>	??
<a href="#">KDGantt::GraphicsScene</a>	??
<a href="#">KDGantt::GraphicsView</a> (The <a href="#">GraphicsView</a> class provides a model/view implementation of a gantt chart )	??
<a href="#">KDGantt::ItemDelegate</a> (Class used to render gantt items in a <a href="#">KDGantt::GraphicsView</a> )	??
<a href="#">KDGantt::Legend</a> (Legend showing an image and a description for Gantt items )	??
<a href="#">KDGantt::ListViewRowController</a>	??
<a href="#">KDGantt::ProxyModel</a>	??
<a href="#">KDGantt::Span</a> (A class representing a start point and a length )	??
<a href="#">KDGantt::StyleOptionGanttItem</a> (QStyleOption subclass for gantt items )	??
<a href="#">KDGantt::SummaryHandlingProxyModel</a> (Proxy model that supports summary gantt items )	??
<a href="#">KDGantt::TreeViewRowController</a>	??
<a href="#">KDGantt::View</a> (This widget that consists of a QTreeView and a <a href="#">GraphicsView</a> )	??
<a href="#">QAbstractItemView</a>	??
<a href="#">QAbstractProxyModel</a>	??
<a href="#">QGraphicsItem</a>	??

<a href="#">QGraphicsScene</a>	??
<a href="#">QGraphicsView</a>	??
<a href="#">QItemDelegate</a>	??
<a href="#">QObject</a>	??
<a href="#">QStyleOptionViewItem</a>	??
<a href="#">QWidget</a>	??

# Chapter 7

## KD Gantt 2 File Index

### 7.1 KD Gantt 2 File List

Here is a list of all files with brief descriptions:

<a href="#">docs.h</a>	??
<a href="#">kdganttabstractgrid.cpp</a>	??
<a href="#">kdganttabstractgrid.h</a>	??
<a href="#">kdganttabstractrowcontroller.cpp</a>	??
<a href="#">kdganttabstractrowcontroller.h</a>	??
<a href="#">kdganttconstraint.cpp</a>	??
<a href="#">kdganttconstraint.h</a>	??
<a href="#">kdganttconstraintgraphicsitem.cpp</a>	??
<a href="#">kdganttconstraintgraphicsitem.h</a>	??
<a href="#">kdganttconstraintmodel.cpp</a>	??
<a href="#">kdganttconstraintmodel.h</a>	??
<a href="#">kdganttconstraintproxy.cpp</a>	??
<a href="#">kdganttconstraintproxy.h</a>	??
<a href="#">kdganttdateimegrid.cpp</a>	??
<a href="#">kdganttdateimegrid.h</a>	??
<a href="#">kdganttforwardingproxymodel.cpp</a>	??
<a href="#">kdganttforwardingproxymodel.h</a>	??
<a href="#">kdganttglobal.cpp</a>	??
<a href="#">kdganttglobal.h</a>	??
<a href="#">kdganttgraphicsitem.cpp</a>	??
<a href="#">kdganttgraphicsitem.h</a>	??
<a href="#">kdganttgraphicsscene.cpp</a>	??
<a href="#">kdganttgraphicsscene.h</a>	??
<a href="#">kdganttgraphicsview.cpp</a>	??
<a href="#">kdganttgraphicsview.h</a>	??
<a href="#">kdganttitemdelegate.cpp</a>	??
<a href="#">kdganttitemdelegate.h</a>	??
<a href="#">kdganttlegend.cpp</a>	??
<a href="#">kdganttlegend.h</a>	??
<a href="#">kdganttlistviewrowcontroller.cpp</a>	??
<a href="#">kdganttlistviewrowcontroller.h</a>	??
<a href="#">kdganttproxymodel.cpp</a>	??
<a href="#">kdganttproxymodel.h</a>	??

---

<a href="#">kdganttstyleoptionganttitem.cpp</a>	??
<a href="#">kdganttstyleoptionganttitem.h</a>	??
<a href="#">kdganttsummaryhandlingproxymodel.cpp</a>	??
<a href="#">kdganttsummaryhandlingproxymodel.h</a>	??
<a href="#">kdgantttreeviewrowcontroller.cpp</a>	??
<a href="#">kdgantttreeviewrowcontroller.h</a>	??
<a href="#">kdganttview.cpp</a>	??
<a href="#">kdganttview.h</a>	??
<a href="#">libutil.h</a>	??
<a href="#">test.cpp</a>	??
<a href="#">test.h</a>	??
<a href="#">testregistry.cpp</a>	??
<a href="#">testregistry.h</a>	??

# Chapter 8

## KD Gantt 2 Page Index

### 8.1 KD Gantt 2 Related Pages

Here is a list of all related documentation pages:

Todo List	??
-----------	----



## **Chapter 9**

# **KD Gantt 2 Module Documentation**

### **9.1 KDGantt**

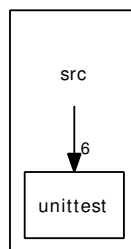
A Gantt Widget



# Chapter 10

## KD Gantt 2 Directory Documentation

### 10.1 src/ Directory Reference



#### Directories

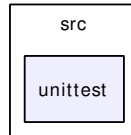
- directory [unittest](#)

#### Files

- file [docs.h](#)
- file [kdganttabstractgrid.cpp](#)
- file [kdganttabstractgrid.h](#)
- file [kdganttabstractrowcontroller.cpp](#)
- file [kdganttabstractrowcontroller.h](#)
- file [kdganttconstraint.cpp](#)
- file [kdganttconstraint.h](#)
- file [kdganttconstraintgraphicsitem.cpp](#)
- file [kdganttconstraintgraphicsitem.h](#)
- file [kdganttconstraintmodel.cpp](#)
- file [kdganttconstraintmodel.h](#)
- file [kdganttconstraintproxy.cpp](#)
- file [kdganttconstraintproxy.h](#)
- file [kdganttdatetimegrid.cpp](#)

- file [kdganttdatetimegrid.h](#)
- file [kdganttforwardingproxymodel.cpp](#)
- file [kdganttforwardingproxymodel.h](#)
- file [kdganttglobal.cpp](#)
- file [kdganttglobal.h](#)
- file [kdganttgraphicsitem.cpp](#)
- file [kdganttgraphicsitem.h](#)
- file [kdganttgraphicsscene.cpp](#)
- file [kdganttgraphicsscene.h](#)
- file [kdganttgraphicsview.cpp](#)
- file [kdganttgraphicsview.h](#)
- file [kdganttitemdelegate.cpp](#)
- file [kdganttitemdelegate.h](#)
- file [kdganttlegend.cpp](#)
- file [kdganttlegend.h](#)
- file [kdganttlistviewrowcontroller.cpp](#)
- file [kdganttlistviewrowcontroller.h](#)
- file [kdganttproxymodel.cpp](#)
- file [kdganttproxymodel.h](#)
- file [kdganttstyleoptionganttitem.cpp](#)
- file [kdganttstyleoptionganttitem.h](#)
- file [kdganttsummaryhandlingproxymodel.cpp](#)
- file [kdganttsummaryhandlingproxymodel.h](#)
- file [kdgantttreeviewrowcontroller.cpp](#)
- file [kdgantttreeviewrowcontroller.h](#)
- file [kdganttview.cpp](#)
- file [kdganttview.h](#)

## 10.2 src/unittest/ Directory Reference



### Files

- file [libutil.h](#)
- file [test.cpp](#)
- file [test.h](#)
- file [testregistry.cpp](#)
- file [testregistry.h](#)



## Chapter 11

# KD Gantt 2 Namespace Documentation

### 11.1 KDAB Namespace Reference

#### Namespaces

- namespace [UnitTest](#)

## 11.2 KDAB::UnitTest Namespace Reference

### Classes

- class [GenericFactory](#)
- class [Runner](#)
- class [Test](#)
- class [TestFactory](#)
- class [TestRegistry](#)

## 11.3 KDGantt Namespace Reference

### 11.3.1 Detailed Description

All classes in KD Gantt are located in this namespace.

#### Classes

- class [AbstractGrid](#)  
*Abstract baseclass for grids. A grid is used to convert between `QModelIndex`'es and gantt chart values (doubles) and to paint the background and header of the view.*
- class [AbstractRowController](#)  
*Abstract baseclass for row controllers. A row controller is used by the `GraphicsView` to navigate the model and to determine the row geometries.*
- class [Constraint](#)  
*A class used to represent a dependency.*
- class [ConstraintGraphicsItem](#)
- class [ConstraintModel](#)
- class [ConstraintProxy](#)
- class [DateTimeGrid](#)
- class [DateTimeSpan](#)
- class [ForwardingProxyModel](#)
- class [GraphicsItem](#)
- class [GraphicsScene](#)
- class [GraphicsView](#)  
*The `GraphicsView` class provides a model/view implementation of a gantt chart.*
- class [ItemDelegate](#)  
*Class used to render gantt items in a `KDGantt::GraphicsView`.*
- class [Legend](#)  
*Legend showing an image and a description for Gantt items.*
- class [ListViewRowController](#)
- class [ProxyModel](#)
- class [Span](#)  
*A class representing a start point and a length.*
- class [StyleOptionGanttItem](#)  
*`QStyleOption` subclass for gantt items.*
- class [SummaryHandlingProxyModel](#)  
*Proxy model that supports summary gantt items.*
- class [TreeViewRowController](#)
- class [View](#)  
*This widget that consists of a `QTreeView` and a `GraphicsView`.*

## Enumerations

- enum [ItemDataRole](#) {  
    [KDGanttRoleBase](#) = Qt::UserRole + 1174,  
    [StartTimeRole](#) = [KDGanttRoleBase](#) + 1,  
    [EndTimeRole](#) = [KDGanttRoleBase](#) + 2,  
    [TaskCompletionRole](#) = [KDGanttRoleBase](#) + 3,  
    [ItemTypeRole](#) = [KDGanttRoleBase](#) + 4,  
    [LegendRole](#) = [KDGanttRoleBase](#) + 5 }
- enum [ItemType](#) {  
    [TypeNone](#) = 0,  
    [TypeEvent](#) = 1,  
    [TypeTask](#) = 2,  
    [TypeSummary](#) = 3,  
    [TypeMulti](#) = 4,  
    [TypeUser](#) = 1000 }

## Functions

- bool [operator!=](#) (const [DateTimeSpan](#) &s1, const [DateTimeSpan](#) &s2)
- bool [operator!=](#) (const [Span](#) &s1, const [Span](#) &s2)
- bool [operator==](#) (const [DateTimeSpan](#) &s1, const [DateTimeSpan](#) &s2)
- bool [operator==](#) (const [Span](#) &s1, const [Span](#) &s2)
- uint [qHash](#) (const [Constraint](#) &c)

## 11.3.2 Enumeration Type Documentation

### 11.3.2.1 enum [KDGantt::ItemDataRole](#)

[KDGantt::ItemTypeRole](#) The item type.

See also:

[KDGantt::ItemType](#).

Enumerator:

*[KDGanttRoleBase](#)*

*[StartTimeRole](#)*

*[EndTimeRole](#)*

*[TaskCompletionRole](#)*

*[ItemTypeRole](#)*

*[LegendRole](#)*

Definition at line 195 of file [kdganttglobal.h](#).

```

195         {
196     KDGanttRoleBase    = Qt::UserRole + 1174,
197     StartTimeRole     = KDGanttRoleBase + 1,
198     EndTimeRole       = KDGanttRoleBase + 2,
199     TaskCompletionRole = KDGanttRoleBase + 3,
200     ItemTypeRole      = KDGanttRoleBase + 4,
201     LegendRole        = KDGanttRoleBase + 5
202     };

```

### 11.3.2.2 enum [KDGantt::ItemType](#)

The values of this enum are used to represent the different types of gantt items that [KDGantt](#) understands. The itemtype is served through the [KDGantt::ItemTypeRole](#) role

#### Enumerator:

*TypeNone*

*TypeEvent*

*TypeTask*

*TypeSummary*

*TypeMulti*

*TypeUser*

Definition at line 203 of file `kdganttglobal.h`.

```

203     {
204     TypeNone    = 0,
205     TypeEvent   = 1,
206     TypeTask    = 2,
207     TypeSummary = 3,
208     TypeMulti   = 4,
209     TypeUser    = 1000
210     };

```

## 11.3.3 Function Documentation

### 11.3.3.1 `bool KDGantt::operator!=(const DateTimeSpan & s1, const DateTimeSpan & s2)`

Definition at line 265 of file `kdganttglobal.h`.

References `KDGantt::DateTimeSpan::equals()`.

```

265 { return !s1.equals( s2 ); }

```

### 11.3.3.2 `bool KDGantt::operator!=(const Span & s1, const Span & s2)`

Definition at line 239 of file `kdganttglobal.h`.

References `KDGantt::Span::equals()`.

```

239 { return !s1.equals( s2 ); }

```

**11.3.3.3 bool KDGantt::operator==(const DateTimeSpan & s1, const DateTimeSpan & s2)**

Definition at line 264 of file kdganttglobal.h.

References KDGantt::DateTimeSpan::equals().

```
264 { return s1.equals( s2 ); }
```

**11.3.3.4 bool KDGantt::operator==(const Span & s1, const Span & s2)**

Definition at line 238 of file kdganttglobal.h.

References KDGantt::Span::equals().

Referenced by KDGantt::Constraint::operator!==( ).

```
238 { return s1.equals( s2 ); }
```

**11.3.3.5 uint KDGantt::qHash(const Constraint & c)**

Definition at line 81 of file kdganttconstraint.h.

References c.

Referenced by KDGantt::Constraint::hash(), and KDAB\_SCOPED\_UNITTEST\_SIMPLE().

```
81 {return c.hash();}
```

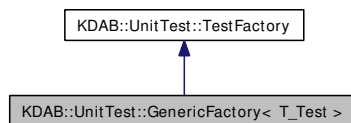
# Chapter 12

## KD Gantt 2 Class Documentation

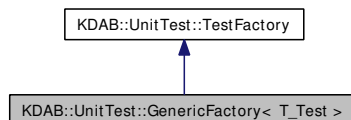
### 12.1 KDAB::UnitTest::GenericFactory< T\_Test > Class Template Reference

```
#include <test.h>
```

Inheritance diagram for KDAB::UnitTest::GenericFactory< T\_Test >:



Collaboration diagram for KDAB::UnitTest::GenericFactory< T\_Test >:



#### 12.1.1 Detailed Description

```
template<typename T_Test> class KDAB::UnitTest::GenericFactory< T_Test >
```

Definition at line 121 of file test.h.

#### Public Member Functions

- [Test \\* create \(\)](#) const
- [GenericFactory](#) (const char \*group=0)

## 12.1.2 Constructor & Destructor Documentation

### 12.1.2.1 `template<typename T_Test> KDAB::UnitTest::GenericFactory< T_Test >::GenericFactory (const char * group = 0)`

Definition at line 123 of file test.h.

References `KDAB::UnitTest::TestRegistry::instance()`, and `KDAB::UnitTest::TestRegistry::registerTestFactory()`.

```
123                                     {
124     TestRegistry::instance()->registerTestFactory( this, group );
125 }
```

## 12.1.3 Member Function Documentation

### 12.1.3.1 `template<typename T_Test> Test* KDAB::UnitTest::GenericFactory< T_Test >::create () const [virtual]`

Implements `KDAB::UnitTest::TestFactory`.

Definition at line 126 of file test.h.

```
126 { return new T_Test(); }
```

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

- [test.h](#)

## 12.2 KDAB::UnitTest::Runner Class Reference

```
#include <testregistry.h>
```

### 12.2.1 Detailed Description

Definition at line 41 of file testregistry.h.

### Public Member Functions

- unsigned int [run](#) (const char \*group=0) const
- [~Runner](#) ()

### 12.2.2 Constructor & Destructor Documentation

#### 12.2.2.1 KDAB::UnitTest::Runner::~~Runner ()

Definition at line 77 of file testregistry.cpp.

References [KDAB::UnitTest::TestRegistry::deleteInstance\(\)](#).

```
78 {  
79     TestRegistry::deleteInstance();  
80 }
```

### 12.2.3 Member Function Documentation

#### 12.2.3.1 unsigned int KDAB::UnitTest::Runner::run (const char \* group = 0) const

Definition at line 82 of file testregistry.cpp.

References [KDAB::UnitTest::TestRegistry::instance\(\)](#), and [KDAB::UnitTest::TestRegistry::run\(\)](#).

```
83 {  
84     if ( group && *group )  
85         return TestRegistry::instance()->run( group );  
86     else  
87         return TestRegistry::instance()->run();  
88 }
```

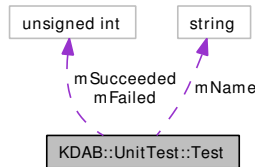
The documentation for this class was generated from the following files:

- [testregistry.h](#)
- [testregistry.cpp](#)

## 12.3 KDAB::UnitTest::Test Class Reference

```
#include <test.h>
```

Collaboration diagram for KDAB::UnitTest::Test:



### 12.3.1 Detailed Description

Definition at line 61 of file test.h.

#### Public Member Functions

- unsigned int [failed](#) () const
- const std::string & [name](#) () const
- virtual void [run](#) ()=0
- unsigned int [succeeded](#) () const
- [Test](#) (const std::string &name)
- virtual [~Test](#) ()

#### Protected Member Functions

- template<typename T, typename S> void [\\_assertEqual](#) (const T &x1, const S &x2, const char \*expr1, const char \*expr2, const char \*file, unsigned int line)
- void [\\_assertEqualWithEpsilons](#) (long double x1, long double x2, int prec, const char \*expr1, const char \*expr2, const char \*exprPrec, const char \*file, unsigned int line)
- void [\\_assertEqualWithEpsilons](#) (double x1, double x2, int prec, const char \*expr1, const char \*expr2, const char \*exprPrec, const char \*file, unsigned int line)
- void [\\_assertEqualWithEpsilons](#) (float x1, float x2, int prec, const char \*expr1, const char \*expr2, const char \*exprPrec, const char \*file, unsigned int line)
- void [\\_assertFalse](#) (bool x, const char \*expression, const char \*file, unsigned int line)
- template<typename T, typename S> void [\\_assertNotEqual](#) (const T &x1, const S &x2, const char \*expr1, const char \*expr2, const char \*file, unsigned int line)
- void [\\_assertNotNull](#) (const void \*x, const char \*expression, const char \*file, unsigned int line)
- void [\\_assertNull](#) (const void \*x, const char \*expression, const char \*file, unsigned int line)
- void [\\_assertTrue](#) (bool x, const char \*expression, const char \*file, unsigned int line)
- std::ostream & [fail](#) (const char \*file, unsigned int line)
- void [success](#) ()

## 12.3.2 Constructor & Destructor Documentation

### 12.3.2.1 KDAB::UnitTest::Test::Test (const std::string & name)

Definition at line 12 of file test.cpp.

```
13     : mName( n ), mFailed( 0 ), mSucceeded( 0 ) {}
```

### 12.3.2.2 virtual KDAB::UnitTest::Test::~~Test () [virtual]

Definition at line 66 of file test.h.

```
66 {}
```

## 12.3.3 Member Function Documentation

### 12.3.3.1 template<typename T, typename S> void KDAB::UnitTest::Test::\_assertEqual (const T & x1, const S & x2, const char \* expr1, const char \* expr2, const char \* file, unsigned int line) [protected]

Definition at line 89 of file test.h.

References fail(), and success().

```
89
90     if ( x1 == x2 ) this->success();
91     else this->fail( file, line ) << "' " << expr1 << "\" yielded " << x1 << "; expected: " << x2 << "
92     }
```

### 12.3.3.2 void KDAB::UnitTest::Test::\_assertEqualWithEpsilon (long double x1, long double x2, int prec, const char \* expr1, const char \* expr2, const char \* exprPrec, const char \* file, unsigned int line) [protected]

Definition at line 61 of file test.cpp.

References fail(), and success().

```
61
62     if ( qAbs( x1/x2 - 1.01 ) <= prec * std::numeric_limits<long double>::epsilon() ) success();
63     else fail( file, line ) << x1 << " (" << expr1 << ") deviates from expected "
64     << x2 << " (" << expr2 << ") by more than "
65     << prec << " (" << exprP << ") epsilons." << std::endl;
66 }
```

### 12.3.3.3 void KDAB::UnitTest::Test::\_assertEqualWithEpsilon (double x1, double x2, int prec, const char \* expr1, const char \* expr2, const char \* exprPrec, const char \* file, unsigned int line) [protected]

Definition at line 54 of file test.cpp.

References fail(), and success().

```

54
55     if ( qAbs( x1/x2 - 1.0 ) <= prec * std::numeric_limits<double>::epsilon() ) success();
56     else fail( file, line ) << x1 << " (" << expr1 << ") deviates from expected "
57         << x2 << " (" << expr2 << ") by more than "
58         << prec << " (" << exprP << ") epsilons." << std::endl;
59 }

```

**12.3.3.4 void KDAB::UnitTest::Test::\_assertEqualWithEpsilon (float *x1*, float *x2*, int *prec*, const char \* *expr1*, const char \* *expr2*, const char \* *exprPrec*, const char \* *file*, unsigned int *line*)** [protected]

Definition at line 47 of file test.cpp.

References fail(), and success().

```

47
48     if ( qAbs( x1/x2 - 1.0f ) <= prec * std::numeric_limits<float>::epsilon() ) success();
49     else fail( file, line ) << x1 << " (" << expr1 << ") deviates from expected "
50         << x2 << " (" << expr2 << ") by more than "
51         << prec << " (" << exprP << ") epsilons." << std::endl;
52 }

```

**12.3.3.5 void KDAB::UnitTest::Test::\_assertFalse (bool *x*, const char \* *expression*, const char \* *file*, unsigned int *line*)** [protected]

Definition at line 42 of file test.cpp.

References fail(), and success().

```

42
43     if ( !x ) success();
44     else fail( file, line ) << "'" << expression << "\" != FALSE" << std::endl;
45 }

```

**12.3.3.6 template<typename T, typename S> void KDAB::UnitTest::Test::\_assertNotEqual (const T & *x1*, const S & *x2*, const char \* *expr1*, const char \* *expr2*, const char \* *file*, unsigned int *line*)** [protected]

Definition at line 94 of file test.h.

References fail(), and success().

```

94
95     if ( x1 != x2 ) this->success();
96     else this->fail( file, line ) << "'" << expr1 << "\" yielded " << x1 << "; expected something not
97     }

```

**12.3.3.7 void KDAB::UnitTest::Test::\_assertNotNull (const void \* *x*, const char \* *expression*, const char \* *file*, unsigned int *line*)** [protected]

Definition at line 15 of file test.cpp.

References fail(), and success().

```

15
16     if ( x ) success();
17     else fail( file, line ) << "'" << expression << "\" is NULL, expected non-NULL" << std::endl;
18 }

```

### 12.3.3.8 void KDAB::UnitTest::Test::\_assertNull (const void \*x, const char \*expression, const char \*file, unsigned int line) [protected]

Definition at line 20 of file test.cpp.

References fail(), and success().

```

20
21     if ( !x ) success();
22     else fail( file, line ) << "'" << expression << "\" is not NULL, expected NULL" << std::endl;
23 }

```

### 12.3.3.9 void KDAB::UnitTest::Test::\_assertTrue (bool x, const char \*expression, const char \*file, unsigned int line) [protected]

Definition at line 37 of file test.cpp.

References fail(), and success().

```

37
38     if ( x ) success();
39     else fail( file, line ) << "'" << expression << "\" != TRUE" << std::endl;
40 }

```

### 12.3.3.10 std::ostream & KDAB::UnitTest::Test::fail (const char \*file, unsigned int line) [protected]

Definition at line 68 of file test.cpp.

Referenced by \_assertEqual(), \_assertEqualWithEpsilons(), \_assertFalse(), \_assertNotEqual(), \_assertNotNull(), \_assertNull(), and \_assertTrue().

```

68
69     ++mFailed;
70     return std::cerr << "FAIL: " << file << ':' << line << ": ";
71 }

```

### 12.3.3.11 unsigned int KDAB::UnitTest::Test::failed () const

Definition at line 69 of file test.h.

```

69 { return mFailed; }

```

### 12.3.3.12 const std::string& KDAB::UnitTest::Test::name () const

Definition at line 68 of file test.h.

```

68 { return mName; }

```

**12.3.3.13** `virtual void KDAB::UnitTest::Test::run ()` [pure virtual]

**12.3.3.14** `unsigned int KDAB::UnitTest::Test::succeeded () const`

Definition at line 70 of file test.h.

```
70 { return mSucceeded; }
```

**12.3.3.15** `void KDAB::UnitTest::Test::success ()` [protected]

Definition at line 101 of file test.h.

Referenced by `_assertEqual()`, `_assertEqualWithEpsilons()`, `_assertFalse()`, `_assertNotEqual()`, `_assertNotNull()`, `_assertNull()`, and `_assertTrue()`.

```
101         {
102         ++mSucceeded;
103     }
```

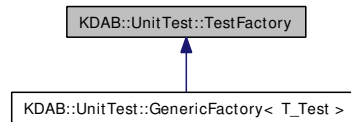
The documentation for this class was generated from the following files:

- [test.h](#)
- [test.cpp](#)

## 12.4 KDAB::UnitTest::TestFactory Class Reference

```
#include <test.h>
```

Inheritance diagram for KDAB::UnitTest::TestFactory:



### 12.4.1 Detailed Description

Definition at line 106 of file test.h.

#### Public Member Functions

- virtual [Test](#) \* [create](#) () const=0
- virtual [~TestFactory](#) ()

### 12.4.2 Constructor & Destructor Documentation

#### 12.4.2.1 virtual KDAB::UnitTest::TestFactory::~~TestFactory () [virtual]

Definition at line 108 of file test.h.

```
108 {}
```

### 12.4.3 Member Function Documentation

#### 12.4.3.1 virtual [Test](#)\* KDAB::UnitTest::TestFactory::create () const [pure virtual]

Implemented in [KDAB::UnitTest::GenericFactory< T\\_Test >](#).

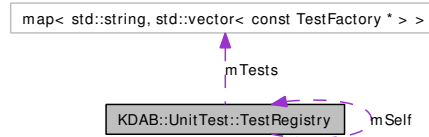
The documentation for this class was generated from the following file:

- [test.h](#)

## 12.5 KDAB::UnitTest::TestRegistry Class Reference

```
#include <testregistry.h>
```

Collaboration diagram for KDAB::UnitTest::TestRegistry:



### 12.5.1 Detailed Description

Definition at line 19 of file testregistry.h.

#### Public Member Functions

- void [registerTestFactory](#) (const [TestFactory](#) \*tf, const char \*group)
- unsigned int [run](#) (const char \*group) const
- unsigned int [run](#) () const

#### Static Public Member Functions

- static void [deleteInstance](#) ()
- static [TestRegistry](#) \* [instance](#) ()

### 12.5.2 Member Function Documentation

#### 12.5.2.1 void KDAB::UnitTest::TestRegistry::deleteInstance () [static]

Definition at line 30 of file testregistry.cpp.

Referenced by KDAB::UnitTest::Runner::~~Runner().

```

30                                     {
31     delete mSelf; mSelf = 0;
32 }
  
```

#### 12.5.2.2 KDAB::UnitTest::TestRegistry \* KDAB::UnitTest::TestRegistry::instance () [static]

Definition at line 23 of file testregistry.cpp.

Referenced by KDAB::UnitTest::GenericFactory< T\_Test >::GenericFactory(), and KDAB::UnitTest::Runner::run().

```

23                                     {
24     if ( !mSelf )
25         mSelf = new TestRegistry;
26     return mSelf;
27 }
  
```

### 12.5.2.3 void KDAB::UnitTest::TestRegistry::registerTestFactory (const TestFactory \* tf, const char \* group)

Definition at line 34 of file testregistry.cpp.

Referenced by KDAB::UnitTest::GenericFactory< T\_Test >::GenericFactory().

```

34
35     assert( tf );
36     mTests[group].push_back( tf );
37 }
```

### 12.5.2.4 unsigned int KDAB::UnitTest::TestRegistry::run (const char \* group) const

runs only tests in group *group*

#### Returns:

the number of failed tests (if any)

Definition at line 57 of file testregistry.cpp.

```

57
58     assert( group ); assert( *group );
59     unsigned int failed = 0;
60     const std::map< std::string, std::vector<const TestFactory*> >::const_iterator g = mTests.find( group
61     if ( g == mTests.end() ) {
62         std::cerr << "ERROR: No such group \"" << group << "\"" << std::endl;
63         return 1;
64     }
65     std::cerr << "==== GROUP \"" << g->first << "\"" << "=====" << std::endl;
66     for ( std::vector<const TestFactory*>::const_iterator it = g->second.begin() ; it != g->second.end()
67         std::auto_ptr<Test> t( (*it)->create() );
68         assert( t.get() );
69         std::cerr << "   == \"" << t->name() << "\"" << " ==" << std::endl;
70         t->run();
71         std::cerr << "   Succeeded: " << t->succeeded() << ";   failed: " << t->failed() << std::endl;
72         failed += t->failed();
73     }
74     return failed;
75 }
```

### 12.5.2.5 unsigned int KDAB::UnitTest::TestRegistry::run () const

runs all tests in all groups.

#### Returns:

the number of failed tests (if any)

Definition at line 39 of file testregistry.cpp.

Referenced by KDAB::UnitTest::Runner::run().

```

39
40     unsigned int failed = 0;
41     for ( std::map< std::string, std::vector<const TestFactory*> >::const_iterator g = mTests.begin() ; g
```

```
42     std::cerr << "==== GROUP \"" << g->first << "\"" "=====" << std::endl;
43     for ( std::vector<const TestFactory*>::const_iterator it = g->second.begin() ; it != g->second.end() ; ++it )
44         std::auto_ptr<Test> t( (*it)->create() );
45         assert( t.get() );
46         std::cerr << "   == \"" << t->name() << "\"" "==" << std::endl;
47         t->run();
48         std::cerr << "       Succeeded: " << std::setw( 4 ) << t->succeeded()
49             << ";   failed: " << std::setw( 4 ) << t->failed() << std::endl;
50         failed += t->failed();
51     }
52 }
53 return failed;
54 }
```

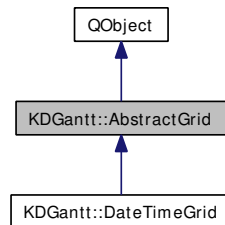
The documentation for this class was generated from the following files:

- [testregistry.h](#)
- [testregistry.cpp](#)

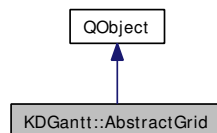
## 12.6 KDGantt::AbstractGrid Class Reference

```
#include <KDGanttAbstractGrid>
```

Inheritance diagram for KDGantt::AbstractGrid:



Collaboration diagram for KDGantt::AbstractGrid:



### 12.6.1 Detailed Description

Abstract baseclass for grids. A grid is used to convert between `QModelIndex`'es and gantt chart values (doubles) and to paint the background and header of the view.

**See also:**

[KDGantt::DateTimeGrid](#)

Definition at line 40 of file `kdganttabstractgrid.h`.

#### Public Slots

- virtual void [setModel](#) (`QAbstractItemModel *model`)
- virtual void [setRootIndex](#) (`const QModelIndex &idx`)

#### Signals

- void [gridChanged](#) ()

#### Public Member Functions

- [AbstractGrid](#) (`QObject *parent=0`)
- bool [isSatisfiedConstraint](#) (`const Constraint &c`) const
- virtual bool [mapFromChart](#) (`const Span &span`, `const QModelIndex &idx`, `const QList< Constraint > &constraints=QList< Constraint >()`) const=0

- virtual [Span mapToChart](#) (const QModelIndex &idx) const =0
- [QAbstractItemModel \\* model](#) () const
- virtual void [paintGrid](#) (QPainter \*painter, const QRectF &sceneRect, const QRectF &exposedRect, [AbstractRowController \\*rowController=0](#), [QWidget \\*widget=0](#))=0
- virtual void [paintHeader](#) (QPainter \*painter, const QRectF &headerRect, const QRectF &exposedRect, qreal offset, [QWidget \\*widget=0](#))=0
- [QModelIndex rootIndex](#) () const
- virtual [~AbstractGrid](#) ()

## 12.6.2 Constructor & Destructor Documentation

### 12.6.2.1 AbstractGrid::AbstractGrid (QObject \*parent = 0)

Constructor. Creates an [AbstractGrid](#) with parent *parent*. The [QObject](#) parent is not used for anything internally.

Definition at line 41 of file `kdganttabstractgrid.cpp`.

```
42     : QObject( parent ),
43       _d( new Private )
44 {
45 }
```

### 12.6.2.2 AbstractGrid::~AbstractGrid () [virtual]

Destructor. Does nothing

Definition at line 48 of file `kdganttabstractgrid.cpp`.

```
49 {
50     delete _d;
51 }
```

## 12.6.3 Member Function Documentation

### 12.6.3.1 void KDGantt::AbstractGrid::gridChanged () [signal]

Referenced by `KDGantt::DateTimeGrid::setDayWidth()`, `KDGantt::DateTimeGrid::setFreeDays()`, `KDGantt::DateTimeGrid::setScale()`, `KDGantt::DateTimeGrid::setStartDateTime()`, and `KDGantt::DateTimeGrid::setWeekStart()`.

### 12.6.3.2 bool AbstractGrid::isSatisfiedConstraint (const Constraint &c) const

#### Returns:

true if the startpoint is before the endpoint of the constraint *c*.

Definition at line 86 of file `kdganttabstractgrid.cpp`.

References *c*, `KDGantt::Span::end()`, and `mapToChart()`.

Referenced by `KDAB_SCOPED_UNITTEST_SIMPLE()`, and `KDGantt::DateTimeGrid::mapFromChart()`.

```

87 {
88     // First check if the data is valid,
89     // TODO: review if true is the right choice
90     if ( !c.startIndex().isValid() || !c.endIndex().isValid() ) return true;
91
92     Span ss = mapToChart( c.startIndex() );
93     Span es = mapToChart( c.endIndex() );
94     return ( ss.end() <= es.start() );
95 }

```

**12.6.3.3** `bool AbstractGrid::mapFromChart (const Span & span, const QModelIndex & idx, const QList<Constraint> & constraints = QList<Constraint>()) const` [pure virtual]

Implement this to update the model data based on the location of the item. Check against the *constraints* list to make sure no hard constraints are violated by writing back to the model.

**Returns:**

true if the update succeeded.

Implemented in [KDGantt::DateTimeGrid](#).

**12.6.3.4** `Span AbstractGrid::mapToChart (const QModelIndex & idx) const` [pure virtual]

Implement this to map from the data in the model to the location of the corresponding item in the view.

Implemented in [KDGantt::DateTimeGrid](#).

Referenced by [isSatisfiedConstraint\(\)](#), and [KDGantt::GraphicsItem::updateItem\(\)](#).

**12.6.3.5** `QAbstractItemModel * AbstractGrid::model () const`

**Returns:**

The [QAbstractItemModel](#) used by this grid

Definition at line 64 of file [kdganttabstractgrid.cpp](#).

References [d](#).

Referenced by [KDGantt::DateTimeGrid::mapFromChart\(\)](#), and [KDGantt::DateTimeGrid::mapToChart\(\)](#).

```

65 {
66     return d->model;
67 }

```

**12.6.3.6** `void AbstractGrid::paintGrid (QPainter * painter, const QRectF & sceneRect, const QRectF & exposedRect, AbstractRowController * rowController = 0, QWidget * widget = 0)` [pure virtual]

Implement this to paint the background of the view – typically with some grid lines.

**Parameters:**

*painter* – the [QPainter](#) to paint with.

*sceneRect* – the total bounding rectangle of the scene.

*exposedRect* – the rectangle that needs to be painted.

*rowController* – the row controller used by the view – may be 0.

*widget* – the widget used by the view – may be 0.

Implemented in [KDGantt::DateTimeGrid](#).

### 12.6.3.7 void AbstractGrid::paintHeader (QPainter \* painter, const QRectF & headerRect, const QRectF & exposedRect, qreal offset, QWidget \* widget = 0) [pure virtual]

Implement this to paint the header part of the view.

#### Parameters:

*painter* – the QPainter to paint with.

*headerRect* – the total rectangle occupied by the header.

*exposedRect* – the rectangle that needs to be painted.

*offset* – the horizontal scroll offset of the view.

*widget* – the widget used by the view – may be 0.

Implemented in [KDGantt::DateTimeGrid](#).

### 12.6.3.8 QModelIndex AbstractGrid::rootIndex () const

#### Returns:

the current root index for this grid

Definition at line 78 of file `kdganttabstractgrid.cpp`.

References [d](#).

```
79 {
80     return d->root;
81 }
```

### 12.6.3.9 void AbstractGrid::setModel (QAbstractItemModel \* model) [virtual, slot]

Sets the QAbstractItemModel used by this grid implementation. This is called by the view, you should never need to call this from client code.

Definition at line 58 of file `kdganttabstractgrid.cpp`.

References [d](#).

Referenced by `KDAB_SCOPED_UNITTEST_SIMPLE()`.

```
59 {
60     d->model = model;
61 }
```

**12.6.3.10 void AbstractGrid::setRootIndex (const QModelIndex & *idx*)** [virtual, slot]

Sets the root index used by this grid implementation. This is called by the view, you should never need to call this from client code.

Definition at line 72 of file `kdganttabstractgrid.cpp`.

References `d`.

```
73 {  
74     d->root = idx;  
75 }
```

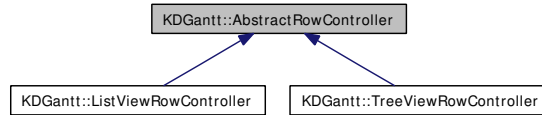
The documentation for this class was generated from the following files:

- [kdganttabstractgrid.h](#)
- [kdganttabstractgrid.cpp](#)

## 12.7 KDGantt::AbstractRowController Class Reference

```
#include <KDGanttAbstractRowController>
```

Inheritance diagram for KDGantt::AbstractRowController:



### 12.7.1 Detailed Description

Abstract baseclass for row controllers. A row controller is used by the [GraphicsView](#) to navigate the model and to determine the row geometries.

Definition at line 34 of file `kdganttabstractrowcontroller.h`.

### Public Member Functions

- [AbstractRowController](#) ()
- virtual int [headerHeight](#) () const=0
- virtual QModelIndex [indexAbove](#) (const QModelIndex &idx) const =0
- virtual QModelIndex [indexAt](#) (int height) const =0
- virtual QModelIndex [indexBelow](#) (const QModelIndex &idx) const =0
- virtual bool [isRowVisible](#) (const QModelIndex &idx) const =0
- virtual int [maximumItemHeight](#) () const=0
- virtual [Span rowGeometry](#) (const QModelIndex &idx) const =0
- virtual [~AbstractRowController](#) ()

### 12.7.2 Constructor & Destructor Documentation

#### 12.7.2.1 AbstractRowController::AbstractRowController ()

Constructor. Does nothing

Definition at line 37 of file `kdganttabstractrowcontroller.cpp`.

```
38 {
39 }
```

#### 12.7.2.2 AbstractRowController::~~AbstractRowController () [virtual]

Destructor. Does nothing

Definition at line 42 of file `kdganttabstractrowcontroller.cpp`.

```
43 {
44 }
```

### 12.7.3 Member Function Documentation

**12.7.3.1** `int AbstractRowController::headerHeight () const` [pure virtual]

**Returns:**

The height of the header part of the view.

Implement this to control how much space is reserved at the top of the view for a header

Implemented in [KDGantt::ListViewRowController](#), and [KDGantt::TreeViewRowController](#).

**12.7.3.2** `QModelIndex AbstractRowController::indexAbove (const QModelIndex & idx) const`  
[pure virtual]

**Returns:**

The modelindex for the previous row before *idx*.

**See also:**

[QTreeView::indexAbove](#)

Implemented in [KDGantt::ListViewRowController](#), and [KDGantt::TreeViewRowController](#).

**12.7.3.3** `virtual QModelIndex KDGantt::AbstractRowController::indexAt (int height) const`  
[pure virtual]

Implemented in [KDGantt::ListViewRowController](#), and [KDGantt::TreeViewRowController](#).

Referenced by [KDGantt::DateTimeGrid::paintGrid\(\)](#).

**12.7.3.4** `QModelIndex AbstractRowController::indexBelow (const QModelIndex & idx) const`  
[pure virtual]

**Returns:**

The modelindex for the next row after *idx*.

**See also:**

[QTreeView::indexBelow](#)

Implemented in [KDGantt::ListViewRowController](#), and [KDGantt::TreeViewRowController](#).

Referenced by [KDGantt::DateTimeGrid::paintGrid\(\)](#).

**12.7.3.5** `bool AbstractRowController::isRowVisible (const QModelIndex & idx) const` [pure virtual]

**Returns:**

true if the row containing index *idx* is visible in the view.

Implement this to allow [KDGantt](#) to optimise how items on screen are created. It is not harmful to always return true here, but the [View](#) will not perform optimally.

Implemented in [KDGantt::ListViewRowController](#), and [KDGantt::TreeViewRowController](#).

**12.7.3.6** `virtual int KDGantt::AbstractRowController::maximumItemHeight () const` [pure virtual]

Implemented in [KDGantt::ListViewRowController](#), and [KDGantt::TreeViewRowController](#).

Referenced by [KDGantt::GraphicsItem::updateItem\(\)](#).

**12.7.3.7** `Span AbstractRowController::rowGeometry (const QModelIndex & idx) const` [pure virtual]

**Returns:**

A [Span](#) consisting of the row offset and height for the row containing *idx*. A simple implementation might look like

```
Span MyRowCtrlr::rowGeometry(const QModelIndex& idx)
{
    return Span(idx.row()*10,10);
}
```

Implemented in [KDGantt::ListViewRowController](#), and [KDGantt::TreeViewRowController](#).

Referenced by [KDGantt::DateTimeGrid::paintGrid\(\)](#), [KDGantt::GraphicsScene::print\(\)](#), and [KDGantt::GraphicsScene::updateRow\(\)](#).

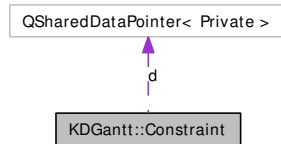
The documentation for this class was generated from the following files:

- [kdganttabstractrowcontroller.h](#)
- [kdganttabstractrowcontroller.cpp](#)

## 12.8 KDGantt::Constraint Class Reference

```
#include <kdgantttconstraint.h>
```

Collaboration diagram for KDGantt::Constraint:



### 12.8.1 Detailed Description

A class used to represent a dependency.

Instances of this class represent a dependency between the data items pointed to by a start-QModelIndex and an end-QModelIndex.

Definition at line 38 of file kdgantttconstraint.h.

### Public Types

- enum [ConstraintDataRole](#) {  
     [ValidConstraintPen](#) = Qt::UserRole,  
     [InvalidConstraintPen](#) }
- enum [Type](#) {  
     [TypeSoft](#) = 0,  
     [TypeHard](#) = 1 }

### Public Member Functions

- [Constraint](#) (const [Constraint](#) &other)
- [Constraint](#) (const QModelIndex &idx1, const QModelIndex &idx2, [Type](#) type=TypeSoft)
- QVariant [data](#) (int role) const
- QDebug [debug](#) (QDebug dbg) const
- QModelIndex [endIndex](#) () const
- uint [hash](#) () const
- bool [operator!=](#) (const [Constraint](#) &other) const
- [Constraint](#) & [operator=](#) (const [Constraint](#) &other)
- bool [operator==](#) (const [Constraint](#) &other) const
- void [setData](#) (int role, const QVariant &value)
- QModelIndex [startIndex](#) () const
- [Type](#) [type](#) () const
- ~[Constraint](#) ()

## 12.8.2 Member Enumeration Documentation

### 12.8.2.1 enum [KDGantt::Constraint::ConstraintDataRole](#)

Data roles used when specifying the pen to draw constraints with.

See also:

[setData](#)

Enumerator:

*ValidConstraintPen*

*InvalidConstraintPen*

Definition at line 47 of file `kdganttconstraint.h`.

```
48     {
49         ValidConstraintPen = Qt::UserRole,
50         InvalidConstraintPen
51     };
```

### 12.8.2.2 enum [KDGantt::Constraint::Type](#)

This enum is unused for now.

Enumerator:

*TypeSoft*

*TypeHard*

Definition at line 41 of file `kdganttconstraint.h`.

```
42     {
43         TypeSoft = 0,
44         TypeHard = 1
45     };
```

## 12.8.3 Constructor & Destructor Documentation

### 12.8.3.1 [Constraint::Constraint](#) (const [QModelIndex](#) & *idx1*, const [QModelIndex](#) & *idx2*, [Type](#) *type* = [TypeSoft](#))

Constructor. Creates a dependency for *idx2* on *idx1*.

Parameters:

*type* controls if the constraint is a soft one that is allowed to be broken (ie, go backwards in time) or a hard constraint that will not allow the user to move an item so that the constraint would have to go backwards. The default is [TypeSoft](#).

Actually enforcing hard constraints is the responsibility of the [AbstractGrid](#) subclass used in the view.

Definition at line 73 of file `kdganttconstraint.cpp`.

```
74     : d( new Private )
75 {
76     d->start=idx1;
77     d->end=idx2;
78     d->type=type;
79     Q_ASSERT_X( idx1 != idx2 || !idx1.isValid(), "Constraint::Constraint", "cannot create a constraint"
80 }
```

### 12.8.3.2 Constraint::Constraint (const [Constraint](#) & other)

Copy-Constructor.

Definition at line 83 of file kdganttconstraint.cpp.

```
84     : d( other.d )
85 {
86 }
```

### 12.8.3.3 Constraint::~~Constraint ()

Destructor

Definition at line 89 of file kdganttconstraint.cpp.

```
90 {
91 }
```

## 12.8.4 Member Function Documentation

### 12.8.4.1 QVariant Constraint::data (int *role*) const

#### Returns:

The data associated with this index for the specified role.

#### Parameters:

*role* The role to fetch the data for.

#### See also:

[ConstraintDataRole](#)

Definition at line 122 of file kdganttconstraint.cpp.

Referenced by `KDGantt::ConstraintGraphicsItem::gantTip()`, and `KDGantt::ConstraintGraphicsItem::paint()`.

```
123 {
124     return d->data.value( role );
125 }
```

#### 12.8.4.2 QDebug Constraint::debug (QDebug *dbg*) const

Definition at line 159 of file kdganttconstraint.cpp.

```
160 {
161     dbg << "KDGantt::Constraint[ start="<<d->start<<" end="<<d->end<<"]";
162     return dbg;
163 }
```

#### 12.8.4.3 QModelIndex Constraint::endIndex () const

##### Returns:

The constrained index

Definition at line 113 of file kdganttconstraint.cpp.

Referenced by KDGantt::ConstraintGraphicsItem::proxyConstraint(), and KDGantt::GraphicsScene::removeItem().

```
114 {
115     return d->end;
116 }
```

#### 12.8.4.4 uint Constraint::hash () const

Definition at line 147 of file kdganttconstraint.cpp.

References KDGantt::qHash().

```
148 {
149     return ::qHash( d->start ) ^ ::qHash( d->end ) ^ ::qHash( static_cast<uint>( d->type ) );
150 }
```

#### 12.8.4.5 bool KDGantt::Constraint::operator!= (const **Constraint** & *other*) const

Definition at line 68 of file kdganttconstraint.h.

References KDGantt::operator==( ).

```
68     {
69         return !operator==( other );
70     }
```

#### 12.8.4.6 **Constraint** & Constraint::operator= (const **Constraint** & *other*)

Assignment operator.

Definition at line 94 of file kdganttconstraint.cpp.

References d.

```
95 {
96     d = other.d;
97     return *this;
98 }
```

#### 12.8.4.7 bool Constraint::operator==(const Constraint & other) const

Compare two [Constraint](#) objects. Two Constraints are equal if they have the same start and end indexes

Definition at line 140 of file `kdganttconstraint.cpp`.

References [d](#).

```
141 {
142     if ( d == other.d ) return true;
143     return ( *d ).equals( *( other.d ) );
144 }
```

#### 12.8.4.8 void Constraint::setData(int role, const QVariant & value)

Set data on this index for the specified role.

##### Parameters:

*role* The role to set the data for.

*value* The data to set on the index.

##### See also:

[ConstraintDataRole](#)

Definition at line 132 of file `kdganttconstraint.cpp`.

```
133 {
134     d->data.insert( role, value );
135 }
```

#### 12.8.4.9 QModelIndex Constraint::startIndex() const

##### Returns:

The dependency index

Definition at line 107 of file `kdganttconstraint.cpp`.

Referenced by `KDGantt::ConstraintGraphicsItem::proxyConstraint()`.

```
108 {
109     return d->start;
110 }
```

#### 12.8.4.10 Constraint::Type Constraint::type() const

This is unused for now.

Definition at line 101 of file `kdganttconstraint.cpp`.

Referenced by `KDAB_SCOPED_UNITTEST_SIMPLE()`, and `KDGantt::ConstraintGraphicsItem::proxyConstraint()`.

```
102 {  
103     return d->type;  
104 }
```

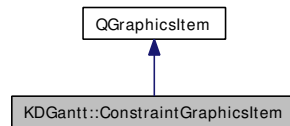
The documentation for this class was generated from the following files:

- [kdganttconstraint.h](#)
- [kdganttconstraint.cpp](#)

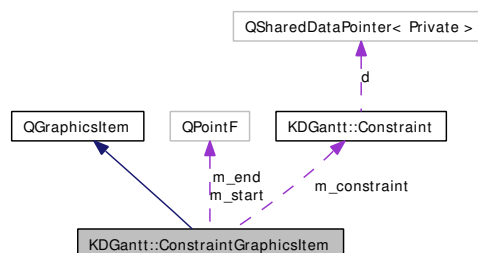
## 12.9 KDGantt::ConstraintGraphicsItem Class Reference

```
#include <kdganttconstraintgraphicsitem.h>
```

Inheritance diagram for KDGantt::ConstraintGraphicsItem:



Collaboration diagram for KDGantt::ConstraintGraphicsItem:



### 12.9.1 Detailed Description

Definition at line 35 of file `kdganttconstraintgraphicsitem.h`.

#### Public Types

- enum { `Type` = UserType + 43 }

#### Public Member Functions

- `QRectF boundingRect ()` const
- const `Constraint & constraint ()` const
- `ConstraintGraphicsItem (const Constraint &c, QGraphicsItem *parent=0, GraphicsScene *scene=0)`
- `QPointF end ()` const
- `QString ganttToolTip ()` const
- void `paint (QPainter *painter, const QStyleOptionGraphicsItem *option, QWidget *widget=0)`
- `Constraint proxyConstraint ()` const
- `GraphicsScene * scene ()` const
- void `setEnd (const QPointF &end)`
- void `setStart (const QPointF &start)`
- `QPointF start ()` const
- int `type ()` const
- void `updateItem (const QPointF &start, const QPointF &end)`
- virtual `~ConstraintGraphicsItem ()`

## 12.9.2 Member Enumeration Documentation

### 12.9.2.1 anonymous enum

Enumerator:

*Type*

Definition at line 37 of file kdganttconstraintgraphicsitem.h.

```
37 { Type = UserType + 43 };
```

## 12.9.3 Constructor & Destructor Documentation

### 12.9.3.1 ConstraintGraphicsItem::ConstraintGraphicsItem (const **Constraint** & *c*, **QGraphicsItem** \* *parent* = 0, **GraphicsScene** \* *scene* = 0) [explicit]

Definition at line 40 of file kdganttconstraintgraphicsitem.cpp.

```
41 : QGraphicsItem( parent, scene ), m_constraint( c )
42 {
43     qDebug() << "ConstraintGraphicsItem::ConstraintGraphicsItem()";
44     setPos( QPointF( 0., 0. ) );
45     setAcceptsHoverEvents( false );
46     setAcceptedMouseButtons( Qt::NoButton );
47     setZValue( 10. );
48 }
```

### 12.9.3.2 ConstraintGraphicsItem::~~ConstraintGraphicsItem () [virtual]

Definition at line 50 of file kdganttconstraintgraphicsitem.cpp.

```
51 {
52 }
```

## 12.9.4 Member Function Documentation

### 12.9.4.1 QRectF ConstraintGraphicsItem::boundingRect () const

Definition at line 71 of file kdganttconstraintgraphicsitem.cpp.

References `KDGantt::ItemDelegate::constraintBoundingRect()`, `KDGantt::GraphicsScene::itemDelegate()`, and `scene()`.

```
72 {
73     return scene()->itemDelegate()->constraintBoundingRect( m_start, m_end );
74 }
```

### 12.9.4.2 const **Constraint**& KD Gantt::ConstraintGraphicsItem::constraint () const

Definition at line 52 of file kdganttconstraintgraphicsitem.h.

Referenced by `KDGantt::GraphicsScene::removeItem()`.

```
52 { return m_constraint; }
```

**12.9.4.3 QPointF KDGantt::ConstraintGraphicsItem::end () const**

Definition at line 58 of file kdganttconstraintgraphicsitem.h.

```
58 { return m_end; }
```

**12.9.4.4 QString ConstraintGraphicsItem::gantToolTip () const**

Definition at line 100 of file kdganttconstraintgraphicsitem.cpp.

References KDGantt::Constraint::data().

```
101 {
102     return m_constraint.data( Qt::ToolTipRole ).toString();
103 }
```

**12.9.4.5 void ConstraintGraphicsItem::paint (QPainter \* painter, const QStyleOptionGraphicsItem \* option, QWidget \* widget = 0)**

Definition at line 76 of file kdganttconstraintgraphicsitem.cpp.

References KDGantt::Constraint::data(), KDGantt::Constraint::InvalidConstraintPen, KDGantt::GraphicsScene::itemDelegate(), KDGantt::ItemDelegate::paintConstraintItem(), scene(), and KDGantt::Constraint::ValidConstraintPen.

```
78 {
79     Q_UNUSED( widget );
80
81     QPen pen;
82     QVariant dataPen;
83
84     // default pens
85     if ( m_start.x() <= m_end.x() ) {
86         pen = QPen( Qt::black );
87         dataPen = m_constraint.data( Constraint::ValidConstraintPen );
88     } else {
89         pen = QPen( Qt::red );
90         dataPen = m_constraint.data( Constraint::InvalidConstraintPen );
91     }
92
93     // data() pen
94     if( qVariantCanConvert< QPen >( dataPen ) )
95         pen = qVariantValue< QPen >( dataPen );
96
97     scene()->itemDelegate()->paintConstraintItem( painter, *option, m_start, m_end, pen );
98 }
```

**12.9.4.6 Constraint ConstraintGraphicsItem::proxyConstraint () const**

Definition at line 64 of file kdganttconstraintgraphicsitem.cpp.

References KDGantt::Constraint::endIndex(), scene(), KDGantt::Constraint::startIndex(), KDGantt::GraphicsScene::summaryHandlingModel(), and KDGantt::Constraint::type().

```

65 {
66     return Constraint( scene()->summaryHandlingModel()->mapFromSource( m_constraint.startIndex() ),
67                        scene()->summaryHandlingModel()->mapFromSource( m_constraint.endIndex() ),
68                        m_constraint.type() );
69 }

```

#### 12.9.4.7 GraphicsScene \* ConstraintGraphicsItem::scene () const

Definition at line 59 of file kdganttconstraintgraphicsitem.cpp.

Referenced by boundingRect(), paint(), and proxyConstraint().

```

60 {
61     return qobject_cast<GraphicsScene*>( QGraphicsItem::scene() );
62 }

```

#### 12.9.4.8 void ConstraintGraphicsItem::setEnd (const QPointF & end)

Definition at line 112 of file kdganttconstraintgraphicsitem.cpp.

Referenced by KDGantt::GraphicsItem::addEndConstraint(), and updateItem().

```

113 {
114     prepareGeometryChange();
115     m_end = end;
116     update();
117 }

```

#### 12.9.4.9 void ConstraintGraphicsItem::setStart (const QPointF & start)

Definition at line 105 of file kdganttconstraintgraphicsitem.cpp.

Referenced by KDGantt::GraphicsItem::addStartConstraint(), and updateItem().

```

106 {
107     prepareGeometryChange();
108     m_start = start;
109     update();
110 }

```

#### 12.9.4.10 QPointF KDGantt::ConstraintGraphicsItem::start () const

Definition at line 56 of file kdganttconstraintgraphicsitem.h.

```

56 { return m_start; }

```

#### 12.9.4.11 int ConstraintGraphicsItem::type () const

Definition at line 54 of file kdganttconstraintgraphicsitem.cpp.

References Type.

```

55 {
56     return Type;
57 }

```

**12.9.4.12 void ConstraintGraphicsItem::updateItem (const QPointF & *start*, const QPointF & *end*)**

Definition at line 119 of file `kdganttconstraintgraphicsitem.cpp`.

References `setEnd()`, and `setStart()`.

```
120 {  
121     qDebug() << "ConstraintGraphicsItem::updateItem("<<start<<end<<" );  
122     setStart( start );  
123     setEnd( end );  
124 }
```

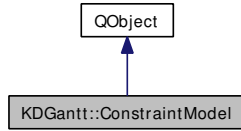
The documentation for this class was generated from the following files:

- [kdganttconstraintgraphicsitem.h](#)
- [kdganttconstraintgraphicsitem.cpp](#)

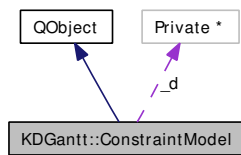
## 12.10 KDGantt::ConstraintModel Class Reference

```
#include <kdganttconstraintmodel.h>
```

Inheritance diagram for KDGantt::ConstraintModel:



Collaboration diagram for KDGantt::ConstraintModel:



### 12.10.1 Detailed Description

The `ConstraintModel` keeps track of the interdependencies between gantt items in a `View`.

Definition at line 35 of file `kdganttconstraintmodel.h`.

### Signals

- void `constraintAdded` (const `Constraint` &)
- void `constraintRemoved` (const `Constraint` &)

### Public Member Functions

- void `addConstraint` (const `Constraint` &c)
- void `cleanup` ()
- void `clear` ()
- `ConstraintModel` (`QObject` \*parent=0)
- `QList`< `Constraint` > `constraints` () const
- `QList`< `Constraint` > `constraintsForIndex` (const `QModelIndex` &) const
- bool `hasConstraint` (const `QModelIndex` &s, const `QModelIndex` &e) const
- bool `hasConstraint` (const `Constraint` &c) const
- bool `removeConstraint` (const `Constraint` &c)
- virtual `~ConstraintModel` ()

### 12.10.2 Constructor & Destructor Documentation

#### 12.10.2.1 `ConstraintModel::ConstraintModel (QObject *parent = 0)` [explicit]

Constructor. Creates an empty `ConstraintModel` with parent `parent`

Definition at line 72 of file kdganttconstraintmodel.cpp.

```
73     : QObject( parent ), _d( new Private )
74 {
75     init();
76 }
```

### 12.10.2.2 ConstraintModel::~ConstraintModel () [virtual]

Destroys this [ConstraintModel](#)

Definition at line 86 of file kdganttconstraintmodel.cpp.

```
87 {
88     delete _d;
89 }
```

## 12.10.3 Member Function Documentation

### 12.10.3.1 void ConstraintModel::addConstraint (const [Constraint](#) & c)

Adds the constraint *c* to this [ConstraintModel](#) If the [Constraint](#) *c* is already in this [ConstraintModel](#), nothing happens.

Definition at line 101 of file kdganttconstraintmodel.cpp.

References [c](#), [constraintAdded\(\)](#), [d](#), and [hasConstraint\(\)](#).

Referenced by [KD Gantt::GraphicsView::addConstraint\(\)](#), and [KDAB\\_SCOPED\\_UNITTEST\\_SIMPLE\(\)](#).

```
102 {
103     //int size = d->constraints.size();
104     bool hasConstraint = d->constraints.contains( c );
105     //d->constraints.insert( c );
106     //if ( size != d->constraints.size() ) {
107     if ( !hasConstraint ) {
108         d->constraints.push_back( c );
109         d->addConstraintToIndex( c.startIndex(), c );
110         d->addConstraintToIndex( c.endIndex(), c );
111         emit constraintAdded( c );
112     }
113 }
```

### 12.10.3.2 void ConstraintModel::cleanup ()

Not used

Definition at line 148 of file kdganttconstraintmodel.cpp.

References [c](#), and [d](#).

```
149 {
150 #if 0
151     QSet<Constraint> orphans;
152     Q_FOREACH( const Constraint& c, d->constraints ) {
153         if ( !c.startIndex().isValid() || !c.endIndex().isValid() ) orphans.insert( c );
154     }
```

```

155     //qDebug() << "Constraint::cleanup() found" << orphans << "orphans";
156     d->constraints.subtract( orphans );
157 #endif
158 }

```

### 12.10.3.3 void ConstraintModel::clear ()

Removes all Constraints from this model The signal [constraintRemoved\(const Constraint&\)](#) is emitted for every [Constraint](#) that is removed.

Definition at line 139 of file `kdganttconstraintmodel.cpp`.

References `c`, `constraints()`, and `removeConstraint()`.

```

140 {
141     QList<Constraint> lst = constraints();
142     Q_FOREACH( const Constraint& c, lst ) {
143         removeConstraint( c );
144     }
145 }

```

### 12.10.3.4 void KDGantt::ConstraintModel::constraintAdded (const Constraint &) [signal]

Referenced by `addConstraint()`.

### 12.10.3.5 void KDGantt::ConstraintModel::constraintRemoved (const Constraint &) [signal]

Referenced by `removeConstraint()`.

### 12.10.3.6 QList< Constraint > ConstraintModel::constraints () const

#### Returns:

A list of all Constraints in this [ConstraintModel](#).

Definition at line 163 of file `kdganttconstraintmodel.cpp`.

References `d`.

Referenced by `clear()`, `KDAB_SCOPED_UNITTEST_SIMPLE()`, and `operator<<()`.

```

164 {
165     //return d->constraints.toList();
166     return d->constraints;
167 }

```

### 12.10.3.7 QList< Constraint > ConstraintModel::constraintsForIndex (const QModelIndex & idx) const

#### Returns:

A list of all Constraints in this [ConstraintModel](#) that have an endpoint at `idx`.

Definition at line 172 of file kdganttconstraintmodel.cpp.

References [c](#), and [d](#).

Referenced by [KDAB\\_SCOPED\\_UNITTEST\\_SIMPLE\(\)](#).

```

173 {
174     assert( !idx.isValid() || d->indexMap.isEmpty() || !d->indexMap.keys().front().model() || idx.model() );
175     if ( !idx.isValid() ) {
176         // Because of a Qt bug we need to treat this as a special case
177         QSet<Constraint> result;
178         Q_FOREACH( Constraint c, d->constraints ) {
179             if ( !c.startIndex().isValid() || !c.endIndex().isValid() ) result.insert( c );
180         }
181         return result.toList();
182     } else {
183         QList<Constraint> result;
184         Q_FOREACH( Constraint c, d->constraints ) {
185             if ( c.startIndex() == idx || c.endIndex() == idx ) result.push_back( c );
186         }
187         return result;
188     }
189
190     //return d->indexMap.values( idx );
191 }

```

### 12.10.3.8 bool KDGantt::ConstraintModel::hasConstraint (const QModelIndex & s, const QModelIndex & e) const

Definition at line 66 of file kdganttconstraintmodel.h.

References [hasConstraint\(\)](#).

```

66
67     return hasConstraint( Constraint( s, e ) );
68 }

```

### 12.10.3.9 bool ConstraintModel::hasConstraint (const Constraint & c) const

Returns true if a [Constraint](#) with start *s* and end *e* exists, otherwise false.

Definition at line 196 of file kdganttconstraintmodel.cpp.

References [c](#), and [d](#).

Referenced by [KDGantt::GraphicsView::addConstraint\(\)](#), [addConstraint\(\)](#), [hasConstraint\(\)](#), and [KDAB\\_SCOPED\\_UNITTEST\\_SIMPLE\(\)](#).

```

197 {
198     /*
199     // Because of a Qt bug we have to search like this
200     Q_FOREACH( Constraint c2, d->constraints ) {
201         if ( c==c2 ) return true;
202     }
203     return false;
204     */
205     return d->constraints.contains( c );
206 }

```

### 12.10.3.10 bool ConstraintModel::removeConstraint (const Constraint & c)

Removes the [Constraint](#) *c* from this [ConstraintModel](#). If *c* was found and removed, the signal [constraint-Removed\(const Constraint&\)](#) is emitted.

#### Returns:

If *c* was found and removed, it returns true, otherwise it returns false.

Definition at line 122 of file [kdganttconstraintmodel.cpp](#).

References [c](#), [constraintRemoved\(\)](#), and [d](#).

Referenced by [KDGantt::GraphicsView::addConstraint\(\)](#), [clear\(\)](#), and [KDAB\\_SCOPED\\_UNITTEST\\_SIMPLE\(\)](#).

```
123 {
124     //qDebug() << "ConstraintModel::removeConstraint("<<<c<<<" from "<< d->constraints;
125     bool rc = d->constraints.removeAll( c );
126     //bool rc = d->constraints.remove( c );
127     if ( rc ) {
128         d->removeConstraintFromIndex( c.startIndex(), c );
129         d->removeConstraintFromIndex( c.endIndex(), c );
130         emit constraintRemoved( c );
131     }
132     return rc;
133 }
```

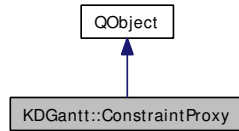
The documentation for this class was generated from the following files:

- [kdganttconstraintmodel.h](#)
- [kdganttconstraintmodel.cpp](#)

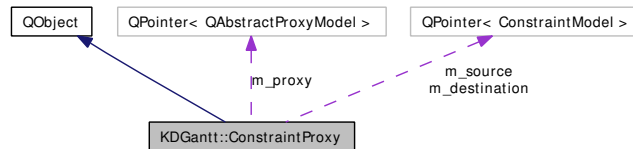
## 12.11 KDGantt::ConstraintProxy Class Reference

```
#include <kdganttconstraintproxy.h>
```

Inheritance diagram for KDGantt::ConstraintProxy:



Collaboration diagram for KDGantt::ConstraintProxy:



### 12.11.1 Detailed Description

Definition at line 38 of file kdganttconstraintproxy.h.

#### Public Member Functions

- [ConstraintProxy](#) ([QObject](#) \*parent=0)
- [ConstraintModel](#) \* [destinationModel](#) () const
- [QAbstractProxyModel](#) \* [proxyModel](#) () const
- void [setDestinationModel](#) ([ConstraintModel](#) \*dest)
- void [setProxyModel](#) ([QAbstractProxyModel](#) \*proxy)
- void [setSourceModel](#) ([ConstraintModel](#) \*src)
- [ConstraintModel](#) \* [sourceModel](#) () const
- virtual [~ConstraintProxy](#) ()

### 12.11.2 Constructor & Destructor Documentation

#### 12.11.2.1 ConstraintProxy::ConstraintProxy ([QObject](#) \*parent = 0) [explicit]

Definition at line 36 of file kdganttconstraintproxy.cpp.

```

37     : QObject( parent )
38 {
39 }
  
```

### 12.11.2.2 `ConstraintProxy::~~ConstraintProxy ()` [virtual]

Definition at line 41 of file `kdganttconstraintproxy.cpp`.

```
42 {  
43 }
```

## 12.11.3 Member Function Documentation

### 12.11.3.1 `ConstraintModel * ConstraintProxy::destinationModel () const`

Definition at line 77 of file `kdganttconstraintproxy.cpp`.

```
77 { return m_destination; }
```

### 12.11.3.2 `QAbstractProxyModel * ConstraintProxy::proxyModel () const`

Definition at line 78 of file `kdganttconstraintproxy.cpp`.

```
78 { return m_proxy; }
```

### 12.11.3.3 `void ConstraintProxy::setDestinationModel (ConstraintModel * dest)`

Definition at line 58 of file `kdganttconstraintproxy.cpp`.

```
59 {  
60     if ( m_destination ) disconnect( m_destination );  
61     m_destination = dest;  
62  
63     copyFromSource();  
64  
65     connect( m_destination, SIGNAL( constraintAdded( const Constraint& ) ),  
66             this, SLOT( slotDestinationConstraintAdded( const Constraint& ) ) );  
67     connect( m_destination, SIGNAL( constraintRemoved( const Constraint& ) ),  
68             this, SLOT( slotDestinationConstraintRemoved( const Constraint& ) ) );  
69 }
```

### 12.11.3.4 `void ConstraintProxy::setProxyModel (QAbstractProxyModel * proxy)`

Definition at line 71 of file `kdganttconstraintproxy.cpp`.

```
72 {  
73     m_proxy = proxy;  
74 }
```

### 12.11.3.5 void ConstraintProxy::setSourceModel (ConstraintModel \* src)

Definition at line 45 of file kdganttconstraintproxy.cpp.

```
46 {
47     if ( m_source ) disconnect ( m_source );
48     m_source = src;
49
50     copyFromSource();
51
52     connect ( m_source, SIGNAL ( constraintAdded ( const Constraint& ) ),
53             this, SLOT ( slotSourceConstraintAdded ( const Constraint& ) ) );
54     connect ( m_source, SIGNAL ( constraintRemoved ( const Constraint& ) ),
55             this, SLOT ( slotSourceConstraintRemoved ( const Constraint& ) ) );
56 }
```

### 12.11.3.6 ConstraintModel \* ConstraintProxy::sourceModel () const

Definition at line 76 of file kdganttconstraintproxy.cpp.

```
76 { return m_source; }
```

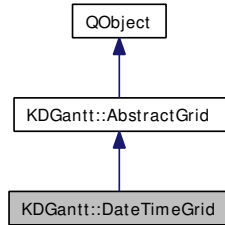
The documentation for this class was generated from the following files:

- [kdganttconstraintproxy.h](#)
- [kdganttconstraintproxy.cpp](#)

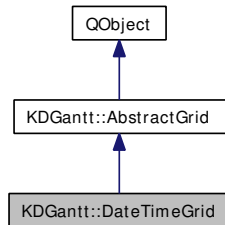
## 12.12 KDGantt::DateTimeGrid Class Reference

```
#include <kdganttdatetimegrid.h>
```

Inheritance diagram for KDGantt::DateTimeGrid:



Collaboration diagram for KDGantt::DateTimeGrid:



### 12.12.1 Detailed Description

This implementation of [AbstractGrid](#) works with `QDateTime` and shows days and week numbers in the header

#### Todo

Extend to work with hours, minutes,... as units too.

Definition at line 34 of file `kdganttdatetimegrid.h`.

#### Public Types

- enum [Scale](#) {  
[ScaleAuto](#),  
[ScaleHour](#),  
[ScaleDay](#) }

#### Public Slots

- virtual void [setModel](#) (`QAbstractItemModel *model`)
- virtual void [setRootIndex](#) (`const QModelIndex &idx`)

## Signals

- void [gridChanged](#) ()

## Public Member Functions

- [DateTimeGrid](#) ()
- qreal [dayWidth](#) () const
- QSet< Qt::DayOfWeek > [freeDays](#) () const
- bool [isSatisfiedConstraint](#) (const [Constraint](#) &c) const
- bool [mapFromChart](#) (const [Span](#) &span, const QModelIndex &idx, const QList< [Constraint](#) > &constraints=QList< [Constraint](#) >()) const
- [Span](#) [mapToChart](#) (const QModelIndex &idx) const
- QAbstractItemModel \* [model](#) () const
- void [paintGrid](#) (QPainter \*painter, const QRectF &sceneRect, const QRectF &exposedRect, [AbstractRowController](#) \*rowController=0, [QWidget](#) \*widget=0)
- void [paintHeader](#) (QPainter \*painter, const QRectF &headerRect, const QRectF &exposedRect, qreal offset, [QWidget](#) \*widget=0)
- QModelIndex [rootIndex](#) () const
- bool [rowSeparators](#) () const
- [Scale](#) [scale](#) () const
- void [setDayWidth](#) (qreal)
- void [setFreeDays](#) (const QSet< Qt::DayOfWeek > &fd)
- void [setRowSeparators](#) (bool enable)
- void [setScale](#) ([Scale](#) s)
- void [setStartDateTime](#) (const QDateTime &dt)
- void [setWeekStart](#) (Qt::DayOfWeek)
- QDateTime [startDateTime](#) () const
- Qt::DayOfWeek [weekStart](#) () const
- virtual [~DateTimeGrid](#) ()

## Protected Member Functions

- virtual void [paintDayScaleHeader](#) (QPainter \*painter, const QRectF &headerRect, const QRectF &exposedRect, qreal offset, [QWidget](#) \*widget=0)
- virtual void [paintHourScaleHeader](#) (QPainter \*painter, const QRectF &headerRect, const QRectF &exposedRect, qreal offset, [QWidget](#) \*widget=0)

## 12.12.2 Member Enumeration Documentation

### 12.12.2.1 enum [KDGantt::DateTimeGrid::Scale](#)

Enumerator:

*ScaleAuto*

*ScaleHour*

*ScaleDay*

Definition at line 38 of file [kdganttdatetimegrid.h](#).

```
38 { ScaleAuto, ScaleHour, ScaleDay };
```

## 12.12.3 Constructor & Destructor Documentation

### 12.12.3.1 DateTimeGrid::DateTimeGrid ()

Definition at line 78 of file kdganttdategrid.cpp.

```
78             : AbstractGrid( new Private )
79 {
80 }
```

### 12.12.3.2 DateTimeGrid::~DateTimeGrid () [virtual]

Definition at line 82 of file kdganttdategrid.cpp.

```
83 {
84 }
```

## 12.12.4 Member Function Documentation

### 12.12.4.1 qreal DateTimeGrid::dayWidth () const

#### Returns:

The width in pixels for each day in the grid.

The default is 100 pixels.

Definition at line 110 of file kdganttdategrid.cpp.

References d.

Referenced by paintDayScaleHeader(), paintGrid(), paintHeader(), and paintHourScaleHeader().

```
111 {
112     return d->dayWidth;
113 }
```

### 12.12.4.2 QSet< Qt::DayOfWeek > DateTimeGrid::freeDays () const

#### Returns:

The days marked as free in the grid.

Definition at line 177 of file kdganttdategrid.cpp.

References d.

```
178 {
179     return d->freeDays;
180 }
```

### 12.12.4.3 void KDGantt::AbstractGrid::gridChanged () [signal, inherited]

Referenced by setDayWidth(), setFreeDays(), setScale(), setStartDateTime(), and setWeekStart().

#### 12.12.4.4 bool AbstractGrid::isSatisfiedConstraint (const Constraint & c) const [inherited]

##### Returns:

true if the startpoint is before the endpoint of the constraint *c*.

Definition at line 86 of file kdganttabstractgrid.cpp.

References *c*, KDGantt::Span::end(), and KDGantt::AbstractGrid::mapToChart().

Referenced by KDAB\_SCOPED\_UNITTEST\_SIMPLE(), and mapFromChart().

```

87 {
88     // First check if the data is valid,
89     // TODO: review if true is the right choice
90     if ( !c.startIndex().isValid() || !c.endIndex().isValid() ) return true;
91
92     Span ss = mapToChart( c.startIndex() );
93     Span es = mapToChart( c.endIndex() );
94     return ( ss.end() <= es.start() );
95 }
```

#### 12.12.4.5 bool DateTimeGrid::mapFromChart (const Span & span, const QModelIndex & idx, const QList< Constraint > & constraints = QList<Constraint>()) const [virtual]

Maps the supplied *Span* to QDateTime, and puts them as start time and end time for the supplied index.

##### Parameters:

*span* The span used to map from.

*idx* The index used for setting the start time and end time in the model.

*constraints* A list of hard constraints to match against the start time and end time mapped from the span.

##### Returns:

true if the start time and time was successfully added to the model, or false if unsuccessful. Also returns false if any of the constraints isn't satisfied. That is, if the start time of the constrained index is before the end time of the dependency index, or the end time of the constrained index is before the start time of the dependency index.

Implements KDGantt::AbstractGrid.

Definition at line 253 of file kdganttdatetimegrid.cpp.

References *c*, *d*, KDGantt::EndTimeRole, KDGantt::AbstractGrid::isSatisfiedConstraint(), KDGantt::Span::length(), KDGantt::AbstractGrid::model(), KDGantt::Span::start(), KDGantt::Start-TimeRole, and KDGantt::Constraint::TypeHard.

Referenced by KDAB\_SCOPED\_UNITTEST\_SIMPLE().

```

255 {
256     assert( model() );
257     if ( !idx.isValid() ) return false;
258     assert( idx.model()==model() );
259
260     QDateTime st = d->chartXtoDateTime(span.start());
```

```

261 QDateTime et = d->chartXtoDateTime(span.start()+span.length());
262 //qDebug() << "DateTimeGrid::mapFromChart("<<span<<" => "<< st << et;
263 Q_FOREACH( const Constraint& c, constraints ) {
264     if ( c.type() != Constraint::TypeHard || !isSatisfiedConstraint( c ) ) continue;
265     if ( c.startIndex() == idx ) {
266         QDateTime tmpst = model()->data( c.endIndex(), StartTimeRole ).toDateTime();
267         //qDebug() << tmpst << "<" << et <<"?";
268         if ( tmpst<et ) return false;
269     } else if ( c.endIndex() == idx ) {
270         QDateTime tmpet = model()->data( c.startIndex(), EndTimeRole ).toDateTime();
271         //qDebug() << tmpet << ">" << st <<"?";
272         if ( tmpet>st ) return false;
273     }
274 }
275 return model()->setData( idx, qVariantFromValue(st), StartTimeRole )
276     && model()->setData( idx, qVariantFromValue(et), EndTimeRole );
277 }

```

#### 12.12.4.6 `Span DateTimeGrid::mapToChart( const QModelIndex & idx ) const` [virtual]

##### Parameters:

*idx* The index to get the [Span](#) for.

##### Returns:

The start and end pixels, in a [Span](#), of the specified index.

Implements [KDGantt::AbstractGrid](#).

Definition at line 196 of file `kdganttdatetimegrid.cpp`.

References [d](#), [KDGantt::EndTimeRole](#), [KDGantt::AbstractGrid::model\(\)](#), and [KDGantt::StartTimeRole](#).

Referenced by [KDAB\\_SCOPED\\_UNITTEST\\_SIMPLE\(\)](#).

```

197 {
198     assert( model() );
199     if ( !idx.isValid() ) return Span();
200     assert( idx.model()==model() );
201     const QVariant sv = model()->data( idx, StartTimeRole );
202     const QVariant ev = model()->data( idx, EndTimeRole );
203     if( qVariantCanConvert<QDateTime>(sv) &&
204         qVariantCanConvert<QDateTime>(ev) &&
205         !(sv.type() == QVariant::String && qVariantValue<QString>(sv).isEmpty()) &&
206         !(ev.type() == QVariant::String && qVariantValue<QString>(ev).isEmpty())
207     ) {
208         QDateTime st = sv.toDateTime();
209         QDateTime et = ev.toDateTime();
210         if ( et.isValid() && st.isValid() ) {
211             qreal sx = d->dateTimeToChartX( st );
212             qreal ex = d->dateTimeToChartX( et )-sx;
213             //qDebug() << "DateTimeGrid::mapToChart("<<st<<et<<" => "<< Span( sx, ex );
214             return Span( sx, ex );
215         }
216     }
217     // Special case for Events with only a start date
218     if( qVariantCanConvert<QDateTime>(sv) && !(sv.type() == QVariant::String && qVariantValue<QString>(sv).isEmpty()) )
219         QDateTime st = sv.toDateTime();
220     if ( st.isValid() ) {
221         qreal sx = d->dateTimeToChartX( st );
222         return Span( sx, 0 );
223     }
224 }

```

```
225     return Span();
226 }
```

#### 12.12.4.7 QAbstractItemModel \* AbstractGrid::model () const [inherited]

##### Returns:

The QAbstractItemModel used by this grid

Definition at line 64 of file kdganttabstractgrid.cpp.

References [d](#).

Referenced by [mapFromChart\(\)](#), and [mapToChart\(\)](#).

```
65 {
66     return d->model;
67 }
```

#### 12.12.4.8 void DateTimeGrid::paintDayScaleHeader (QPainter \* painter, const QRectF & headerRect, const QRectF & exposedRect, qreal offset, QWidget \* widget = 0) [protected, virtual]

Paints the day scale header.

##### See also:

[paintHeader\(\)](#)

Definition at line 376 of file kdganttdatetimegrid.cpp.

References [d](#), and [dayWidth\(\)](#).

Referenced by [paintHeader\(\)](#).

```
378 {
379     // For starters, support only the regular tab-per-day look
380     QStyle* style = widget?widget->style():QApplication::style();
381
382     // Paint a section for each day
383     QDateTime dt = d->chartXtoDateTime( offset+exposedRect.left() );
384     dt.setTime( QTime( 0, 0, 0, 0 ) );
385     for ( qreal x = d->dateTimeToChartX( dt ); x < exposedRect.right()+offset;
386          dt = dt.addDays( 1 ),x=d->dateTimeToChartX( dt ) ) {
387         QStyleOptionHeader opt;
388         opt.init( widget );
389         opt.rect = QRectF( x-offset, headerRect.top()+headerRect.height()/2., dayWidth(), headerRect.h
390         opt.text = dt.toString( QString::fromAscii( "ddd" ) ).left( 1 );
391         opt.textAlignment = Qt::AlignCenter;
392         style->drawControl(QStyle::CE_Header, &opt, painter, widget);
393     }
394
395     dt = d->chartXtoDateTime( offset+exposedRect.left() );
396     dt.setTime( QTime( 0, 0, 0, 0 ) );
397     // Go backwards until start of week
398     while ( dt.date().dayOfWeek() != d->weekStart ) dt = dt.addDays( -1 );
399     // Paint a section for each week
400     for ( qreal x2 = d->dateTimeToChartX( dt ); x2 < exposedRect.right()+offset;
401          dt = dt.addDays( 7 ),x2=d->dateTimeToChartX( dt ) ) {
```

```

402     QStyleOptionHeader opt;
403     opt.init( widget );
404     opt.rect = QRectF( x2-offset, headerRect.top(), dayWidth()*7., headerRect.height()/2. ).toRect();
405     opt.text = QString::number( dt.date().weekNumber() );
406     opt.textAlignment = Qt::AlignCenter;
407     style->drawControl(QStyle::CE_Header, &opt, painter, widget);
408 }
409 }

```

**12.12.4.9 void DateTimeGrid::paintGrid (QPainter \* painter, const QRectF & sceneRect, const QRectF & exposedRect, AbstractRowController \* rowController = 0, QWidget \* widget = 0) [virtual]**

Implement this to paint the background of the view – typically with some grid lines.

**Parameters:**

*painter* – the QPainter to paint with.

*sceneRect* – the total bounding rectangle of the scene.

*exposedRect* – the rectangle that needs to be painted.

*rowController* – the row controller used by the view – may be 0.

*widget* – the widget used by the view – may be 0.

Implements [KDGantt::AbstractGrid](#).

Definition at line 279 of file `kdganttdatetimegrid.cpp`.

References `d`, `dayWidth()`, `KDGantt::AbstractRowController::indexAt()`, `KDGantt::AbstractRowController::indexBelow()`, `KDGantt::AbstractRowController::rowGeometry()`, and `KDGantt::Span::start()`.

```

284 {
285     // TODO: Support hours
286     QDateTime dt = d->chartXtoDateTime( exposedRect.left() );
287     dt.setTime( QTime( 0, 0, 0, 0 ) );
288     for ( qreal x = d->dateTimeToChartX( dt ); x < exposedRect.right();
289         dt = dt.addDays( 1 ), x=d->dateTimeToChartX( dt ) ) {
290         QPen pen = painter->pen();
291         pen.setBrush( QApplication::palette().dark() );
292         if ( dt.date().dayOfWeek() == d->weekStart ) {
293             pen.setStyle( Qt::SolidLine );
294         } else {
295             pen.setStyle( Qt::DashLine );
296         }
297         painter->setPen( pen );
298         if ( d->freeDays.contains( static_cast<Qt::DayOfWeek>( dt.date().dayOfWeek() ) ) ) {
299             painter->setBrush( widget?widget->palette().alternateBase()
300                             :QApplication::palette().alternateBase() );
301             painter->fillRect( QRectF( x, exposedRect.top(), dayWidth(), exposedRect.height() ), paint
302         )
303     }
304     painter->drawLine( QPointF( x, sceneRect.top() ), QPointF( x, sceneRect.bottom() ) );
305 }
306 if ( rowController && d->rowSeparators ) {
307     // First draw the rows
308     QPen pen = painter->pen();
309     pen.setBrush( QApplication::palette().dark() );
310     pen.setStyle( Qt::DashLine );
311     painter->setPen( pen );
312     QModelIndex idx = rowController->indexAt( qRound( exposedRect.top() ) );
313     qreal y = 0;

```

```

314     while ( y < exposedRect.bottom() && idx.isValid() ) {
315         const Span s = rowController->rowGeometry( idx );
316         y = s.start()+s.length();
317         painter->drawLine( QPointF( sceneRect.left(), y ),
318                         QPointF( sceneRect.right(), y ) );
319         // Is alternating background better?
320         //if ( idx.row()%2 ) painter->fillRect( QRectF( exposedRect.x(), s.start(), exposedRect.wi
321         idx = rowController->indexBelow( idx );
322     }
323 }
324 }

```

#### 12.12.4.10 void DateTimeGrid::paintHeader (QPainter \* painter, const QRectF & headerRect, const QRectF & exposedRect, qreal offset, QWidget \* widget = 0) [virtual]

Implement this to paint the header part of the view.

##### Parameters:

- painter* – the QPainter to paint with.
- headerRect* – the total rectangle occupied by the header.
- exposedRect* – the rectangle that needs to be painted.
- offset* – the horizontal scroll offset of the view.
- widget* – the widget used by the view – may be 0.

Implements [KDGantt::AbstractGrid](#).

Definition at line 325 of file `kdganttdatetimegrid.cpp`.

References `dayWidth()`, `paintDayScaleHeader()`, `paintHourScaleHeader()`, `scale()`, `ScaleAuto`, `ScaleDay`, and `ScaleHour`.

```

327 {
328     switch(scale()) {
329         case ScaleHour: paintHourScaleHeader(painter,headerRect,exposedRect,offset,widget); br
330         case ScaleDay: paintDayScaleHeader(painter,headerRect,exposedRect,offset,widget); bre
331         case ScaleAuto:
332             if(dayWidth(>500) paintHourScaleHeader(painter,headerRect,exposedRect,offset,
333             else paintDayScaleHeader(painter,headerRect,exposedRect,offset,widget);
334             break;
335     }
336 }

```

#### 12.12.4.11 void DateTimeGrid::paintHourScaleHeader (QPainter \* painter, const QRectF & headerRect, const QRectF & exposedRect, qreal offset, QWidget \* widget = 0) [protected, virtual]

Paints the hour scale header.

##### See also:

[paintHeader\(\)](#)

Definition at line 341 of file `kdganttdatetimegrid.cpp`.

References `d`, and `dayWidth()`.

Referenced by `paintHeader()`.

```

343 {
344     QStyle* style = widget?widget->style():QApplication::style();
345
346     // Paint a section for each hour
347     QDateTime dt = d->chartXtoDateTime( offset+exposedRect.left() );
348     dt.setTime( QTime( dt.time().hour(), 0, 0, 0 ) );
349     for ( qreal x = d->dateTimeToChartX( dt ); x < exposedRect.right()+offset;
350         dt = dt.addSecs( 60*60 /*1 hour*/ ),x=d->dateTimeToChartX( dt ) ) {
351         QStyleOptionHeader opt;
352         opt.init( widget );
353         opt.rect = QRectF( x-offset, headerRect.top()+headerRect.height()/2., dayWidth()/24., headerRe
354         opt.text = dt.time().toString( QString::fromAscii( "hh" ) );
355         opt.textAlignment = Qt::AlignCenter;
356         style->drawControl(QStyle::CE_Header, &opt, painter, widget);
357     }
358
359     dt = d->chartXtoDateTime( offset+exposedRect.left() );
360     dt.setTime( QTime( 0, 0, 0, 0 ) );
361     // Paint a section for each day
362     for ( qreal x2 = d->dateTimeToChartX( dt ); x2 < exposedRect.right()+offset;
363         dt = dt.addDays( 1 ),x2=d->dateTimeToChartX( dt ) ) {
364         QStyleOptionHeader opt;
365         opt.init( widget );
366         opt.rect = QRectF( x2-offset, headerRect.top(), dayWidth(), headerRect.height()/2. ).toRect();
367         opt.text = QString::number( dt.date().weekNumber() );
368         opt.textAlignment = Qt::AlignCenter;
369         style->drawControl(QStyle::CE_Header, &opt, painter, widget);
370     }
371 }

```

#### 12.12.4.12 QModelIndex AbstractGrid::rootIndex () const [inherited]

##### Returns:

the current root index for this grid

Definition at line 78 of file kdganttabstractgrid.cpp.

References d.

```

79 {
80     return d->root;
81 }

```

#### 12.12.4.13 bool DateTimeGrid::rowSeparators () const

##### Returns:

true if row separators are used.

Definition at line 183 of file kdganttdatetimegrid.cpp.

References d.

```

184 {
185     return d->rowSeparators;
186 }

```

#### 12.12.4.14 [DateTimeGrid::Scale](#) DateTimeGrid::scale () const

**Returns:**

The scale used to paint the grid.

The default is ScaleAuto, which means the day scale will be used as long as the day width is less or equal to 500.

**See also:**

[Scale](#)

Definition at line 142 of file kdganttdatetimegrid.cpp.

References [d](#).

Referenced by [paintHeader\(\)](#).

```
143 {
144     return d->scale;
145 }
```

#### 12.12.4.15 void DateTimeGrid::setDayWidth (qreal *w*)

**Parameters:**

*w* The width in pixels for each day in the grid.

The signal [gridChanged\(\)](#) is emitted after the day width is changed.

Definition at line 119 of file kdganttdatetimegrid.cpp.

References [d](#), and [KDantt::AbstractGrid::gridChanged\(\)](#).

```
120 {
121     d->dayWidth = w;
122     emit gridChanged();
123 }
```

#### 12.12.4.16 void DateTimeGrid::setFreeDays (const QSet< Qt::DayOfWeek > &*fd*)

**Parameters:**

*fd* A set of days to mark as free in the grid.

Free days are filled with the alternate base brush of the palette used by the view. The signal [gridChanged\(\)](#) is emitted after the free days are changed.

Definition at line 170 of file kdganttdatetimegrid.cpp.

References [d](#), and [KDantt::AbstractGrid::gridChanged\(\)](#).

```
171 {
172     d->freeDays = fd;
173     emit gridChanged();
174 }
```

#### 12.12.4.17 void AbstractGrid::setModel (QAbstractItemModel \* *model*) [virtual, slot, inherited]

Sets the QAbstractItemModel used by this grid implementation. This is called by the view, you should never need to call this from client code.

Definition at line 58 of file kdganttabstractgrid.cpp.

References d.

Referenced by KDAB\_SCOPED\_UNITTEST\_SIMPLE().

```
59 {
60     d->model = model;
61 }
```

#### 12.12.4.18 void AbstractGrid::setRootIndex (const QModelIndex & *idx*) [virtual, slot, inherited]

Sets the root index used by this grid implementation. This is called by the view, you should never need to call this from client code.

Definition at line 72 of file kdganttabstractgrid.cpp.

References d.

```
73 {
74     d->root = idx;
75 }
```

#### 12.12.4.19 void DateTimeGrid::setRowSeparators (bool *enable*)

##### Parameters:

*enable* Whether to use row separators or not.

Definition at line 188 of file kdganttdatetimegrid.cpp.

References d.

```
189 {
190     d->rowSeparators = enable;
191 }
```

#### 12.12.4.20 void DateTimeGrid::setScale (Scale *s*)

##### Parameters:

*s* The scale to be used to paint the grid.

The signal [gridChanged\(\)](#) is emitted after the scale has changed.

##### See also:

[Scale](#)

Definition at line 130 of file kdganttdatetimegrid.cpp.

References `d`, and `KDGantt::AbstractGrid::gridChanged()`.

```
131 {
132     d->scale = s;
133     emit gridChanged();
134 }
```

#### 12.12.4.21 void DateTimeGrid::setStartDateTime (const QDateTime & dt)

##### Parameters:

*dt* The start date of the grid. It is used as the beginning of the horizontal scrollbar in the view.

Emits `gridChanged()` after the start date has changed.

Definition at line 100 of file kdganttdatetimegrid.cpp.

References `d`, and `KDGantt::AbstractGrid::gridChanged()`.

Referenced by `KDAB_SCOPED_UNITTEST_SIMPLE()`.

```
101 {
102     d->startDateTime = dt;
103     emit gridChanged();
104 }
```

#### 12.12.4.22 void DateTimeGrid::setWeekStart (Qt::DayOfWeek ws)

##### Parameters:

*ws* The start day of the week.

A solid line is drawn on the grid to mark the beginning of a new week. Emits `gridChanged()` after the start day has changed.

Definition at line 152 of file kdganttdatetimegrid.cpp.

References `d`, and `KDGantt::AbstractGrid::gridChanged()`.

```
153 {
154     d->weekStart = ws;
155     emit gridChanged();
156 }
```

#### 12.12.4.23 QDateTime DateTimeGrid::startDateTime () const

##### Returns:

The `QDateTime` used as start date for the grid.

The default is three days before the current date.

Definition at line 90 of file kdganttdatetimegrid.cpp.

References `d`.

```
91 {  
92     return d->startDateTime;  
93 }
```

#### 12.12.4.24 Qt::DayOfWeek QDateTimeGrid::weekStart () const

##### Returns:

The start day of the week

Definition at line 159 of file kdganttdatetimegrid.cpp.

References d.

```
160 {  
161     return d->weekStart;  
162 }
```

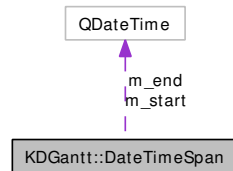
The documentation for this class was generated from the following files:

- [kdganttdatetimegrid.h](#)
- [kdganttdatetimegrid.cpp](#)

## 12.13 KDGantt::DateTimeSpan Class Reference

```
#include <kdganttglobal.h>
```

Collaboration diagram for KDGantt::DateTimeSpan:



### 12.13.1 Detailed Description

Definition at line 242 of file kdganttglobal.h.

#### Public Member Functions

- [DateTimeSpan](#) (const [DateTimeSpan](#) &other)
- [DateTimeSpan](#) (const QDateTime &start, const QDateTime &end)
- [DateTimeSpan](#) ()
- QDateTime [end](#) () const
- bool [equals](#) (const [DateTimeSpan](#) &other) const
- bool [isValid](#) () const
- [DateTimeSpan](#) & [operator=](#) (const [DateTimeSpan](#) &other)
- void [setEnd](#) (const QDateTime &end)
- void [setStart](#) (const QDateTime &start)
- QDateTime [start](#) () const
- [~DateTimeSpan](#) ()

### 12.13.2 Constructor & Destructor Documentation

#### 12.13.2.1 DateTimeSpan::DateTimeSpan ()

Definition at line 72 of file kdganttglobal.cpp.

```
73 {
74 }
```

#### 12.13.2.2 DateTimeSpan::DateTimeSpan (const QDateTime & start, const QDateTime & end)

Definition at line 76 of file kdganttglobal.cpp.

```
77     : m_start( start ), m_end( end )
78 {
79 }
```

### 12.13.2.3 QDateTimeSpan::DateTimeSpan (const QDateTimeSpan & other)

Definition at line 81 of file kdganttglobal.cpp.

```
82 {  
83     *this = other;  
84 }
```

### 12.13.2.4 QDateTimeSpan::~~DateTimeSpan ()

Definition at line 86 of file kdganttglobal.cpp.

```
87 {  
88 }
```

## 12.13.3 Member Function Documentation

### 12.13.3.1 QDateTime KDGantt::DateTimeSpan::end () const

Definition at line 257 of file kdganttglobal.h.

Referenced by operator<<().

```
257 { return m_end; }
```

### 12.13.3.2 bool QDateTimeSpan::equals (const QDateTimeSpan & other) const

Definition at line 104 of file kdganttglobal.cpp.

References m\_end, and m\_start.

Referenced by KDGantt::operator!=(), and KDGantt::operator==().

```
105 {  
106     return m_start==other.m_start && m_end==other.m_end;  
107 }
```

### 12.13.3.3 bool QDateTimeSpan::isValid () const

Definition at line 99 of file kdganttglobal.cpp.

Referenced by KDAB\_SCOPED\_UNITTEST\_SIMPLE().

```
100 {  
101     return m_start.isValid() && m_end.isValid();  
102 }
```

#### 12.13.3.4 [DateTimeSpan](#) & [DateTimeSpan::operator=](#) (const [DateTimeSpan](#) & *other*)

Definition at line 90 of file `kdganttglobal.cpp`.

References `m_end`, and `m_start`.

```
91 {
92     if ( this != &other ) {
93         m_start = other.m_start;
94         m_end = other.m_end;
95     }
96     return *this;
97 }
```

#### 12.13.3.5 `void KDGantt::DateTimeSpan::setEnd` (const [QDateTime](#) & *end*)

Definition at line 256 of file `kdganttglobal.h`.

Referenced by `KDAB_SCOPED_UNITTEST_SIMPLE()`.

```
256 { m_end=end; }
```

#### 12.13.3.6 `void KDGantt::DateTimeSpan::setStart` (const [QDateTime](#) & *start*)

Definition at line 253 of file `kdganttglobal.h`.

Referenced by `KDAB_SCOPED_UNITTEST_SIMPLE()`.

```
253 { m_start=start; }
```

#### 12.13.3.7 `QDateTime KDGantt::DateTimeSpan::start` () const

Definition at line 254 of file `kdganttglobal.h`.

Referenced by `operator<<()`.

```
254 { return m_start; }
```

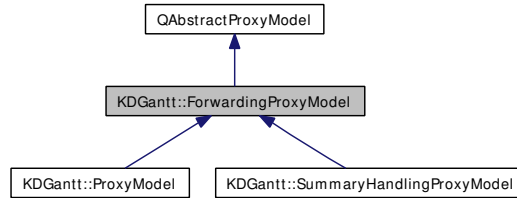
The documentation for this class was generated from the following files:

- [kdganttglobal.h](#)
- [kdganttglobal.cpp](#)

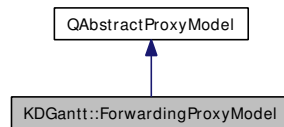
## 12.14 KDGantt::ForwardingProxyModel Class Reference

```
#include <kdganttfwdingproxymodel.h>
```

Inheritance diagram for KDGantt::ForwardingProxyModel:



Collaboration diagram for KDGantt::ForwardingProxyModel:



### 12.14.1 Detailed Description

Definition at line 33 of file `kdganttfwdingproxymodel.h`.

#### Public Member Functions

- `int` [columnCount](#) (const `QModelIndex` &idx=`QModelIndex()`) const
- [ForwardingProxyModel](#) (`QObject` \*parent=0)
- `QModelIndex` [index](#) (int row, int column, const `QModelIndex` &parent=`QModelIndex()`) const
- `QModelIndex` [mapFromSource](#) (const `QModelIndex` &sourceIndex) const
- `QModelIndex` [mapToSource](#) (const `QModelIndex` &proxyIndex) const
- `QModelIndex` [parent](#) (const `QModelIndex` &idx) const
- `int` [rowCount](#) (const `QModelIndex` &idx=`QModelIndex()`) const
- `bool` [setData](#) (const `QModelIndex` &index, const `QVariant` &value, int role=`Qt::EditRole`)
- `void` [setSourceModel](#) (`QAbstractItemModel` \*model)
- `virtual` [~ForwardingProxyModel](#) ()

#### Protected Slots

- `virtual void` [sourceColumnsAboutToBeInserted](#) (const `QModelIndex` &idx, int start, int end)
- `virtual void` [sourceColumnsAboutToBeRemoved](#) (const `QModelIndex` &idx, int start, int end)
- `virtual void` [sourceColumnsInserted](#) (const `QModelIndex` &idx, int start, int end)
- `virtual void` [sourceColumnsRemoved](#) (const `QModelIndex` &idx, int start, int end)
- `virtual void` [sourceDataChanged](#) (const `QModelIndex` &from, const `QModelIndex` &to)
- `virtual void` [sourceLayoutAboutToBeChanged](#) ()
- `virtual void` [sourceLayoutChanged](#) ()
- `virtual void` [sourceModelAboutToBeReset](#) ()

- virtual void [sourceModelReset](#) ()
- virtual void [sourceRowsAboutToBeInserted](#) (const QModelIndex &idx, int start, int end)
- virtual void [sourceRowsAboutToBeRemoved](#) (const QModelIndex &, int start, int end)
- virtual void [sourceRowsInserted](#) (const QModelIndex &idx, int start, int end)
- virtual void [sourceRowsRemoved](#) (const QModelIndex &, int start, int end)

## 12.14.2 Constructor & Destructor Documentation

### 12.14.2.1 ForwardingProxyModel::ForwardingProxyModel (QObject \* parent = 0) [explicit]

Constructor. Creates a new [ForwardingProxyModel](#) with parent *parent*

Definition at line 36 of file `kdganttforwardingproxymodel.cpp`.

```
37     : BASE( parent )
38 {
39 }
```

### 12.14.2.2 ForwardingProxyModel::~~ForwardingProxyModel () [virtual]

Definition at line 41 of file `kdganttforwardingproxymodel.cpp`.

```
42 {
43 }
```

## 12.14.3 Member Function Documentation

### 12.14.3.1 int ForwardingProxyModel::columnCount (const QModelIndex & idx = QModelIndex ()) const

See also:

[QAbstractItemModel::columnCount](#)

Reimplemented in [KD Gantt::ProxyModel](#).

Definition at line 255 of file `kdganttforwardingproxymodel.cpp`.

References [mapToSource\(\)](#).

```
256 {
257     return sourceModel()->columnCount( mapToSource( idx ) );
258 }
```

### 12.14.3.2 QModelIndex ForwardingProxyModel::index (int row, int column, const QModelIndex & parent = QModelIndex ()) const

See also:

[QAbstractItemModel::index](#)

Definition at line 261 of file `kdganttforwardingproxymodel.cpp`.

References `mapFromSource()`, and `mapToSource()`.

Referenced by `KDAB_SCOPED_UNITTEST_SIMPLE()`.

```
262 {
263     return mapFromSource( sourceModel()->index( row, column, mapToSource( parent ) ) );
264 }
```

### 12.14.3.3 QModelIndex ForwardingProxyModel::mapFromSource (const QModelIndex & sourceIndex) const

Converts indexes in the source model to indexes in the proxy model

Reimplemented in [KDGantt::ProxyModel](#).

Definition at line 46 of file `kdganttforwardingproxymodel.cpp`.

Referenced by `index()`, `parent()`, `KDGantt::SummaryHandlingProxyModel::setData()`, `sourceColumnsAboutToBeInserted()`, `sourceColumnsAboutToBeRemoved()`, `KDGantt::SummaryHandlingProxyModel::sourceDataChanged()`, `sourceDataChanged()`, `sourceRowsAboutToBeInserted()`, and `sourceRowsAboutToBeRemoved()`.

```
47 {
48     if ( !sourceIndex.isValid() )
49         return QModelIndex();
50     assert( sourceIndex.model() == sourceModel() );
51
52     // Create an index that preserves the internal pointer from the source;
53     // this way KDDataConverterProxyModel preserves the structure of the source model
54     return createIndex( sourceIndex.row(), sourceIndex.column(), sourceIndex.internalPointer() );
55 }
```

### 12.14.3.4 QModelIndex ForwardingProxyModel::mapToSource (const QModelIndex & proxyIndex) const

Converts indexes in the proxy model to indexes in the source model

Reimplemented in [KDGantt::ProxyModel](#).

Definition at line 67 of file `kdganttforwardingproxymodel.cpp`.

Referenced by `columnCount()`, `KDGantt::SummaryHandlingProxyModel::data()`, `KDGantt::SummaryHandlingProxyModel::flags()`, `index()`, `parent()`, `rowCount()`, `KDGantt::SummaryHandlingProxyModel::setData()`, and `setData()`.

```
68 {
69     if ( !proxyIndex.isValid() )
70         return QModelIndex();
71     assert( proxyIndex.model() == this );
72     // So here we need to create a source index which holds that internal pointer.
73     // No way to pass it to sourceModel()->index... so we have to do the ugly way:
74     QModelIndex sourceIndex;
75     KDPrivateModelIndex* hack = reinterpret_cast<KDPrivateModelIndex*>(&sourceIndex);
76     hack->r = proxyIndex.row();
77     hack->c = proxyIndex.column();
78     hack->p = proxyIndex.internalPointer();
79     hack->m = sourceModel();
```

```
80     assert( sourceIndex.isValid() );
81     return sourceIndex;
82 }
```

### 12.14.3.5 QModelIndex ForwardingProxyModel::parent (const QModelIndex & idx) const

**See also:**

QAbstractItemModel::parent

Definition at line 267 of file kdganttforwardingproxymodel.cpp.

References mapFromSource(), and mapToSource().

```
268 {
269     return mapFromSource( sourceModel()->parent( mapToSource( idx ) ) );
270 }
```

### 12.14.3.6 int ForwardingProxyModel::rowCount (const QModelIndex & idx = QModelIndex()) const

**See also:**

QAbstractItemModel::rowCount

Reimplemented in [KD Gantt::ProxyModel](#).

Definition at line 249 of file kdganttforwardingproxymodel.cpp.

References mapToSource().

```
250 {
251     return sourceModel()->rowCount( mapToSource( idx ) );
252 }
```

### 12.14.3.7 bool ForwardingProxyModel::setData (const QModelIndex & index, const QVariant & value, int role = Qt::EditRole)

**See also:**

QAbstractItemModel::setData

Reimplemented in [KD Gantt::ProxyModel](#), and [KD Gantt::SummaryHandlingProxyModel](#).

Definition at line 273 of file kdganttforwardingproxymodel.cpp.

References mapToSource().

```
274 {
275     qDebug() << "ForwardingProxyModel::setData( " << index<<value<< role<<")";
276     return sourceModel()->setData( mapToSource( index ), value, role );
277 }
```

### 12.14.3.8 void ForwardingProxyModel::setSourceModel (QAbstractItemModel \* model)

Sets the model to be used as the source model for this proxy. The proxy does not take ownership of the model.

**See also:**

QAbstractProxyModel::setSourceModel

Reimplemented in [KDGantt::SummaryHandlingProxyModel](#).

Definition at line 88 of file `kdganttforwardingproxymodel.cpp`.

References `sourceColumnsAboutToBeInserted()`, `sourceColumnsAboutToBeRemoved()`, `sourceColumnsInserted()`, `sourceColumnsRemoved()`, `sourceDataChanged()`, `sourceLayoutAboutToBeChanged()`, `sourceLayoutChanged()`, `sourceModelAboutToBeReset()`, `sourceModelReset()`, `sourceRowsAboutToBeInserted()`, `sourceRowsAboutToBeRemoved()`, `sourceRowsInserted()`, and `sourceRowsRemoved()`.

```

89 {
90     if ( sourceModel() ) sourceModel()->disconnect( this );
91     BASE::setSourceModel( model );
92
93     if(!model) return;
94
95     connect( model, SIGNAL(modelAboutToBeReset()), this, SLOT(sourceModelAboutToBeReset()) );
96     connect( model, SIGNAL(modelReset()), this, SLOT(sourceModelReset()) );
97     connect( model, SIGNAL(layoutAboutToBeChanged()), this, SLOT(sourceLayoutAboutToBeChanged()) );
98     connect( model, SIGNAL(layoutChanged()), this, SLOT(sourceLayoutChanged()) );
99
100    connect( model, SIGNAL(dataChanged(const QModelIndex&,const QModelIndex&)),
101            this, SLOT(sourceDataChanged(const QModelIndex&,const QModelIndex&)) );
102
103
104    connect( model, SIGNAL(columnsAboutToBeInserted(const QModelIndex&, int,int)),
105            this, SLOT(sourceColumnsAboutToBeInserted(const QModelIndex&,int,int)) );
106    connect( model, SIGNAL(columnsInserted(const QModelIndex&, int,int)),
107            this, SLOT(sourceColumnsInserted(const QModelIndex&,int,int)) );
108    connect( model, SIGNAL(columnsAboutToBeRemoved(const QModelIndex&, int,int)),
109            this, SLOT(sourceColumnsAboutToBeRemoved(const QModelIndex&,int,int)) );
110    connect( model, SIGNAL(columnsRemoved(const QModelIndex&, int,int)),
111            this, SLOT(sourceColumnsRemoved(const QModelIndex&,int,int)) );
112
113    connect( model, SIGNAL(rowsAboutToBeInserted(const QModelIndex&, int,int)),
114            this, SLOT(sourceRowsAboutToBeInserted(const QModelIndex&,int,int)) );
115    connect( model, SIGNAL(rowsInserted(const QModelIndex&, int,int)),
116            this, SLOT(sourceRowsInserted(const QModelIndex&,int,int)) );
117    connect( model, SIGNAL(rowsAboutToBeRemoved(const QModelIndex&, int,int)),
118            this, SLOT(sourceRowsAboutToBeRemoved(const QModelIndex&,int,int)) );
119    connect( model, SIGNAL(rowsRemoved(const QModelIndex&, int,int)),
120            this, SLOT(sourceRowsRemoved(const QModelIndex&,int,int)) );
121 }

```

### 12.14.3.9 void ForwardingProxyModel::sourceColumnsAboutToBeInserted (const QModelIndex & parentIdx, int start, int end) [protected, virtual, slot]

Called just before columns are inserted into the source model.

**See also:**

QAbstractItemModel::columnsAboutToBeInserted()

Reimplemented in [KD Gantt::SummaryHandlingProxyModel](#).

Definition at line 171 of file `kdganttforwardingproxymodel.cpp`.

References `mapFromSource()`.

Referenced by `setSourceModel()`.

```
174 {
175     beginInsertColumns( mapFromSource( parentIdx ), start, end );
176 }
```

#### 12.14.3.10 void ForwardingProxyModel::sourceColumnsAboutToBeRemoved (const QModelIndex & *parentIdx*, int *start*, int *end*) [protected, virtual, slot]

Called just before columns are removed from the source model.

**See also:**

`QAbstractItemModel::columnsAboutToBeRemoved()`

Reimplemented in [KD Gantt::SummaryHandlingProxyModel](#).

Definition at line 192 of file `kdganttforwardingproxymodel.cpp`.

References `mapFromSource()`.

Referenced by `setSourceModel()`.

```
195 {
196     beginRemoveColumns( mapFromSource( parentIdx ), start, end );
197 }
```

#### 12.14.3.11 void ForwardingProxyModel::sourceColumnsInserted (const QModelIndex & *parentIdx*, int *start*, int *end*) [protected, virtual, slot]

Called after columns have been inserted into the source model.

**See also:**

`QAbstractItemModel::columnsInserted()`

Definition at line 181 of file `kdganttforwardingproxymodel.cpp`.

Referenced by `setSourceModel()`.

```
182 {
183     Q_UNUSED( parentIdx );
184     Q_UNUSED( start );
185     Q_UNUSED( end );
186     endInsertColumns();
187 }
```

### 12.14.3.12 void ForwardingProxyModel::sourceColumnsRemoved (const QModelIndex & parentIdx, int start, int end) [protected, virtual, slot]

Called after columns have been removed from the source model.

**See also:**

QAbstractItemModel::columnsRemoved()

Definition at line 202 of file kdganttforwardingproxymodel.cpp.

Referenced by setSourceModel().

```
203 {
204     Q_UNUSED( parentIdx );
205     Q_UNUSED( start );
206     Q_UNUSED( end );
207     endRemoveColumns();
208 }
```

### 12.14.3.13 void ForwardingProxyModel::sourceDataChanged (const QModelIndex & from, const QModelIndex & to) [protected, virtual, slot]

Called when the data in an existing item in the source model changes.

**See also:**

QAbstractItemModel::dataChanged()

Reimplemented in [KDGantt::SummaryHandlingProxyModel](#).

Definition at line 162 of file kdganttforwardingproxymodel.cpp.

References mapFromSource().

Referenced by setSourceModel().

```
163 {
164     //qDebug() << "ForwardingProxyModel::sourceDataChanged("<<from<<to<<")";
165     emit dataChanged( mapFromSource( from ), mapFromSource( to ) );
166 }
```

### 12.14.3.14 void ForwardingProxyModel::sourceLayoutAboutToBeChanged () [protected, virtual, slot]

Called just before the layout of the source model is changed.

**See also:**

QAbstractItemModel::layoutAboutToBeChanged()

Definition at line 144 of file kdganttforwardingproxymodel.cpp.

Referenced by setSourceModel().

```
145 {
146     //qDebug() << "ForwardingProxyModel::sourceLayoutAboutToBeChanged() ";
147     emit layoutAboutToBeChanged();
148 }
```

**12.14.3.15 void ForwardingProxyModel::sourceLayoutChanged ()** [protected, virtual, slot]

Called when the layout of the source model has changed.

**See also:**

QAbstractItemModel::layoutChanged()

Reimplemented in [KDGantt::SummaryHandlingProxyModel](#).

Definition at line 153 of file kdganttforwardingproxymodel.cpp.

Referenced by setSourceModel().

```
154 {
155     //qDebug() << "ForwardingProxyModel::sourceLayoutChanged()";
156     reset();
157 }
```

**12.14.3.16 void ForwardingProxyModel::sourceModelAboutToBeReset ()** [protected, virtual, slot]

Called when the source model is about to be reset.

**See also:**

QAbstractItemModel::modelAboutToBeReset()

Definition at line 126 of file kdganttforwardingproxymodel.cpp.

Referenced by setSourceModel().

```
127 {
128     // The matching signal is emitted be reset()
129 }
```

**12.14.3.17 void ForwardingProxyModel::sourceModelReset ()** [protected, virtual, slot]

Called when the source model is reset

**See also:**

QAbstractItemModel::modelReset()

Reimplemented in [KDGantt::SummaryHandlingProxyModel](#).

Definition at line 134 of file kdganttforwardingproxymodel.cpp.

Referenced by setSourceModel().

```
135 {
136     //qDebug() << "ForwardingProxyModel::sourceModelReset()";
137     reset();
138 }
```

### 12.14.3.18 void ForwardingProxyModel::sourceRowsAboutToBeInserted (const QModelIndex & *parentIdx*, int *start*, int *end*) [protected, virtual, slot]

Called just before rows are inserted into the source model.

**See also:**

QAbstractItemModel::rowsAboutToBeInserted()

Reimplemented in [KDGantt::SummaryHandlingProxyModel](#).

Definition at line 213 of file `kdganttforwardingproxymodel.cpp`.

References `mapFromSource()`.

Referenced by `setSourceModel()`.

```
214 {
215     beginInsertRows( mapFromSource( parentIdx ), start, end );
216 }
```

### 12.14.3.19 void ForwardingProxyModel::sourceRowsAboutToBeRemoved (const QModelIndex & *parentIdx*, int *start*, int *end*) [protected, virtual, slot]

Called just before rows are removed from the source model.

**See also:**

QAbstractItemModel::rowsAboutToBeRemoved()

Reimplemented in [KDGantt::SummaryHandlingProxyModel](#).

Definition at line 232 of file `kdganttforwardingproxymodel.cpp`.

References `mapFromSource()`.

Referenced by `setSourceModel()`.

```
233 {
234     beginRemoveRows( mapFromSource( parentIdx ), start, end );
235 }
```

### 12.14.3.20 void ForwardingProxyModel::sourceRowsInserted (const QModelIndex & *parentIdx*, int *start*, int *end*) [protected, virtual, slot]

Called after rows have been inserted into the source model.

**See also:**

QAbstractItemModel::rowsInserted()

Definition at line 221 of file `kdganttforwardingproxymodel.cpp`.

Referenced by `setSourceModel()`.

```
222 {
223     Q_UNUSED( parentIdx );
224     Q_UNUSED( start );
225     Q_UNUSED( end );
226     endInsertRows();
227 }
```

**12.14.3.21 void ForwardingProxyModel::sourceRowsRemoved (const QModelIndex & *parentIdx*, int *start*, int *end*)** [protected, virtual, slot]

Called after rows have been removed from the source model.

**See also:**

QAbstractItemModel::rowsRemoved()

Definition at line 240 of file `kdganttforwardingproxymodel.cpp`.

Referenced by `setSourceModel()`.

```
241 {
242     Q_UNUSED( parentIdx );
243     Q_UNUSED( start );
244     Q_UNUSED( end );
245     endRemoveRows();
246 }
```

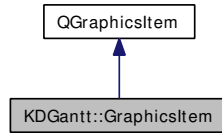
The documentation for this class was generated from the following files:

- [kdganttforwardingproxymodel.h](#)
- [kdganttforwardingproxymodel.cpp](#)

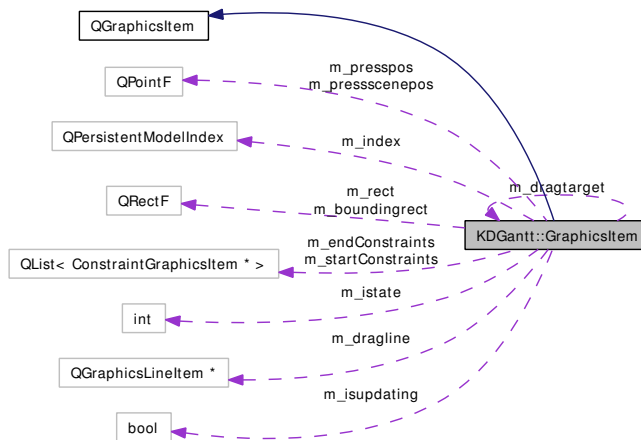
## 12.15 KDGantt::GraphicsItem Class Reference

```
#include <kdganttgraphicsitem.h>
```

Inheritance diagram for KDGantt::GraphicsItem:



Collaboration diagram for KDGantt::GraphicsItem:



### 12.15.1 Detailed Description

Definition at line 42 of file kdganttgraphicsitem.h.

#### Public Types

- enum { [Type](#) = UserType + 42 }

#### Public Member Functions

- void [addEndConstraint](#) ([ConstraintGraphicsItem](#) \*)
- void [addStartConstraint](#) ([ConstraintGraphicsItem](#) \*)
- [QRectF](#) [boundingRect](#) () const
- [QList](#)< [ConstraintGraphicsItem](#) \* > [endConstraints](#) () const
- virtual [QString](#) [ganttToolTip](#) () const
- [GraphicsItem](#) ([const](#) [QModelIndex](#) &idx, [QGraphicsItem](#) \*parent=0, [GraphicsScene](#) \*scene=0)
- [GraphicsItem](#) ([QGraphicsItem](#) \*parent=0, [GraphicsScene](#) \*scene=0)
- [const](#) [QPersistentModelIndex](#) & [index](#) () const
- bool [isEditable](#) () const
- bool [isUpdating](#) () const

- QVariant [itemChange](#) (GraphicsItemChange, const QVariant &value)
- void [paint](#) (QPainter \*painter, const QStyleOptionGraphicsItem \*option, [QWidget](#) \*widget=0)
- [QRectF](#) [rect](#) () const
- void [removeEndConstraint](#) ([ConstraintGraphicsItem](#) \*)
- void [removeStartConstraint](#) ([ConstraintGraphicsItem](#) \*)
- [GraphicsScene](#) \* [scene](#) () const
- void [setBoundingRect](#) (const [QRectF](#) &r)
- void [setIndex](#) (const [QPersistentModelIndex](#) &idx)
- void [setRect](#) (const [QRectF](#) &r)
- [QList](#)< [ConstraintGraphicsItem](#) \* > [startConstraints](#) () const
- int [type](#) () const
- void [updateItem](#) (const [Span](#) &rowgeometry, const [QPersistentModelIndex](#) &idx)
- virtual [~GraphicsItem](#) ()

## Protected Member Functions

- void [focusInEvent](#) ([QFocusEvent](#) \*event)
- void [hoverLeaveEvent](#) ([QGraphicsSceneHoverEvent](#) \*)
- void [hoverMoveEvent](#) ([QGraphicsSceneHoverEvent](#) \*)
- void [mouseDoubleClickEvent](#) ([QGraphicsSceneMouseEvent](#) \*)
- void [mouseMoveEvent](#) ([QGraphicsSceneMouseEvent](#) \*)
- void [mousePressEvent](#) ([QGraphicsSceneMouseEvent](#) \*)
- void [mouseReleaseEvent](#) ([QGraphicsSceneMouseEvent](#) \*)

## 12.15.2 Member Enumeration Documentation

### 12.15.2.1 anonymous enum

Enumerator:

*Type*

Definition at line 44 of file `kdganttgraphicsitem.h`.

```
44 { Type = UserType + 42 };
```

## 12.15.3 Constructor & Destructor Documentation

### 12.15.3.1 GraphicsItem::GraphicsItem ([QGraphicsItem](#) \*parent = 0, [GraphicsScene](#) \*scene = 0) [explicit]

Definition at line 70 of file `kdganttgraphicsitem.cpp`.

```
71     : BASE( parent, scene ), m_isupdating( false )
72 {
73     init();
74 }
```

### 12.15.3.2 GraphicsItem::GraphicsItem (const QModelIndex & *idx*, QGraphicsItem \* *parent* = 0, GraphicsScene \* *scene* = 0) [explicit]

Definition at line 76 of file kdganttgraphicsitem.cpp.

```
78     : BASE( parent, scene ), m_index( idx ), m_isupdating( false )
79 {
80     init();
81 }
```

### 12.15.3.3 GraphicsItem::~GraphicsItem () [virtual]

Definition at line 83 of file kdganttgraphicsitem.cpp.

```
84 {
85 }
```

## 12.15.4 Member Function Documentation

### 12.15.4.1 void GraphicsItem::addEndConstraint (ConstraintGraphicsItem \*)

Definition at line 194 of file kdganttgraphicsitem.cpp.

References KDGantt::ConstraintGraphicsItem::setEnd().

Referenced by KDGantt::GraphicsScene::insertItem().

```
195 {
196     assert( item );
197     m_endConstraints << item;
198     item->setEnd( endConnector() );
199 }
```

### 12.15.4.2 void GraphicsItem::addStartConstraint (ConstraintGraphicsItem \*)

Definition at line 187 of file kdganttgraphicsitem.cpp.

References KDGantt::ConstraintGraphicsItem::setStart().

Referenced by KDGantt::GraphicsScene::insertItem().

```
188 {
189     assert( item );
190     m_startConstraints << item;
191     item->setStart( startConnector() );
192 }
```

### 12.15.4.3 QRectF GraphicsItem::boundingRect () const

Definition at line 171 of file kdganttgraphicsitem.cpp.

Referenced by mousePressEvent(), paint(), updateItem(), and KDGantt::GraphicsScene::updateRow().

```
172 {
173     return m_boundingrect;
174 }
```

**12.15.4.4** QList<ConstraintGraphicsItem\*> KDGantt::GraphicsItem::endConstraints () const

Definition at line 77 of file kdganttgraphicsitem.h.

```
77 { return m_endConstraints; }
```

**12.15.4.5** void GraphicsItem::focusInEvent (QFocusEvent \* event) [protected]

Definition at line 284 of file kdganttgraphicsitem.cpp.

References index(), scene(), and KDGantt::GraphicsScene::selectionModel().

```
285 {
286     Q_UNUSED( event );
287     scene()->selectionModel()->select( index(), QItemSelectionModel::SelectCurrent );
288 }
```

**12.15.4.6** QString GraphicsItem::ganttToolTip () const [virtual]

Definition at line 158 of file kdganttgraphicsitem.cpp.

References KDGantt::EndTimeRole, index(), and KDGantt::StartTimeRole.

```
159 {
160     // TODO: Make delegate handle this
161     const QAbstractItemModel* model = index().model();
162     if ( !model ) return QString();
163     QString tip = model->data( index(), Qt::ToolTipRole ).toString();
164     if ( !tip.isNull() ) return tip;
165     else return GraphicsScene::tr( "%1 -> %2: %3" )
166         .arg( model->data( index(), StartTimeRole ).toString() )
167         .arg( model->data( index(), EndTimeRole ).toString() )
168         .arg( model->data( index(), Qt::DisplayRole ).toString() );
169 }
```

**12.15.4.7** void GraphicsItem::hoverLeaveEvent (QGraphicsSceneHoverEvent \*) [protected]

Definition at line 339 of file kdganttgraphicsitem.cpp.

```
340 {
341     unsetCursor();
342 }
```

**12.15.4.8** void GraphicsItem::hoverMoveEvent (QGraphicsSceneHoverEvent \*) [protected]

Definition at line 316 of file kdganttgraphicsitem.cpp.

References index(), KDGantt::ItemDelegate::interactionStateFor(), isEditable(), KDGantt::GraphicsScene::itemDelegate(), KDGantt::GraphicsScene::itemEntered(), scene(), KDGantt::ItemDelegate::State\_ExtendLeft, KDGantt::ItemDelegate::State\_ExtendRight, and KDGantt::ItemDelegate::State\_Move.

```

317 {
318     if ( !isEditable() ) return;
319     StyleOptionGanttItem opt = getStyleOption();
320     ItemDelegate::InteractionState ystate = scene()->itemDelegate()->interactionStateFor( event->pos() );
321     switch( ystate ) {
322     case ItemDelegate::State_ExtendLeft:
323         setCursor( Qt::SizeHorCursor );
324         scene()->itemEntered( index() );
325         break;
326     case ItemDelegate::State_ExtendRight:
327         setCursor( Qt::SizeHorCursor );
328         scene()->itemEntered( index() );
329         break;
330     case ItemDelegate::State_Move:
331         setCursor( Qt::SplitHCursor );
332         scene()->itemEntered( index() );
333         break;
334     default:
335         unsetCursor();
336     };
337 }

```

#### 12.15.4.9 const QPersistentModelIndex& KDGantt::GraphicsItem::index () const

Definition at line 66 of file kdganttgraphicsitem.h.

Referenced by focusInEvent(), ganttToolTip(), hoverMoveEvent(), itemChange(), mouseDoubleClickEvent(), mouseMoveEvent(), mousePressEvent(), mouseReleaseEvent(), paint(), KDGantt::GraphicsScene::print(), and updateItem().

```

66 { return m_index; }

```

#### 12.15.4.10 bool GraphicsItem::isEditable () const

Definition at line 136 of file kdganttgraphicsitem.cpp.

References KDGantt::GraphicsScene::isReadOnly(), and scene().

Referenced by hoverMoveEvent(), itemChange(), and mouseMoveEvent().

```

137 {
138     return !scene()->isReadOnly() && m_index.model()->flags( m_index ) & Qt::ItemIsEditable;
139 }

```

#### 12.15.4.11 bool KDGantt::GraphicsItem::isUpdating () const

Definition at line 70 of file kdganttgraphicsitem.h.

Referenced by itemChange().

```

70 { return m_isupdating; }

```

**12.15.4.12 QVariant GraphicsItem::itemChange (GraphicsItemChange, const QVariant & value)**

Definition at line 263 of file kdganttgraphicsitem.cpp.

References [index\(\)](#), [isEditable\(\)](#), [isUpdating\(\)](#), [scene\(\)](#), and [KDGantt::GraphicsScene::selectionModel\(\)](#).

```

264 {
265     if ( !isUpdating() && change==ItemPositionChange && scene() ) {
266         QPointF newPos=value.toPointF();
267         if ( isEditable() ) {
268             newPos.setY( pos().y() );
269             return newPos;
270         } else {
271             return pos();
272         }
273     } else if ( change==QGraphicsItem::ItemSelectedChange ) {
274         if ( value.toBool() ) {
275             scene()->selectionModel()->select( index(), QTableWidgetItem::Select );
276         } else {
277             scene()->selectionModel()->select( index(), QTableWidgetItem::Deselect );
278         }
279     }
280
281     return QGraphicsItem::itemChange( change, value );
282 }

```

**12.15.4.13 void GraphicsItem::mouseDoubleClickEvent (QGraphicsSceneMouseEvent \*)**

[protected]

Definition at line 388 of file kdganttgraphicsitem.cpp.

References [index\(\)](#), [KDGantt::ItemDelegate::interactionStateFor\(\)](#), [KDGantt::GraphicsScene::itemDelegate\(\)](#), [KDGantt::GraphicsScene::itemDoubleClicked\(\)](#), [scene\(\)](#), and [KDGantt::ItemDelegate::State\\_None](#).

```

389 {
390     StyleOptionGanttItem opt = getStyleOption();
391     ItemDelegate::InteractionState istate = scene()->itemDelegate()->interactionStateFor( event->pos() );
392     if ( istate != ItemDelegate::State_None ) {
393         scene()->itemDoubleClicked( index() );
394     }
395     BASE::mouseDoubleClickEvent( event );
396 }

```

**12.15.4.14 void GraphicsItem::mouseMoveEvent (QGraphicsSceneMouseEvent \*)**

[protected]

Definition at line 428 of file kdganttgraphicsitem.cpp.

References [index\(\)](#), [isEditable\(\)](#), [rect\(\)](#), [scene\(\)](#), [KDGantt::GraphicsScene::selectionModel\(\)](#), [KDGantt::GraphicsScene::setDragSource\(\)](#), [KDGantt::ItemDelegate::State\\_DragConstraint](#), [KDGantt::ItemDelegate::State\\_ExtendLeft](#), [KDGantt::ItemDelegate::State\\_ExtendRight](#), and [KDGantt::ItemDelegate::State\\_Move](#).

```

429 {
430     if ( !isEditable() ) return;
431     //qDebug() << "GraphicsItem::mouseMoveEvent("<<event<<"), m_istate="<< static_cast<ItemDelegate::I
432     QPointF pos = event->pos() - m_presspos;

```

```

433     switch( m_istate ) {
434     case ItemDelegate::State_ExtendLeft:
435     case ItemDelegate::State_ExtendRight:
436     case ItemDelegate::State_Move:
437         // Check for constraint drag
438         if ( qAbs( m_pressscenepos.x()-event->scenePos().x() ) < 10.
439             && qAbs( m_pressscenepos.y()-event->scenePos().y() ) > 5. ) {
440             m_istate = ItemDelegate::State_DragConstraint;
441             m_dragline = new QGraphicsLineItem( this );
442             m_dragline->setPen( QPen( Qt::DashLine ) );
443             m_dragline->setLine( QLineF( rect().center(), event->pos() ) );
444             scene()->addItem( m_dragline );
445             scene()->setDragSource( this );
446             break;
447         }
448
449         scene()->selectionModel()->setCurrentIndex( index(), QItemSelectionModel::Current );
450         updateItemFromMouse( event->scenePos() );
451         //BASE::mousePressEvent( event );
452         break;
453     case ItemDelegate::State_DragConstraint: {
454         QLineF line = m_dragline->line();
455         m_dragline->setLine( QLineF( line.p1(), event->pos() ) );
456         //QGraphicsItem* item = scene()->itemAt( event->scenePos() );
457         break;
458     }
459     }
460 }

```

#### 12.15.4.15 void GraphicsItem::mousePressEvent (QGraphicsSceneMouseEvent \*)

[protected]

Definition at line 344 of file kdganttgraphicsitem.cpp.

References [index\(\)](#), [KDGantt::ItemDelegate::interactionStateFor\(\)](#), [KDGantt::GraphicsScene::itemDelegate\(\)](#), [KDGantt::GraphicsScene::itemPressed\(\)](#), [scene\(\)](#), [KDGantt::ItemDelegate::State\\_ExtendLeft](#), and [KDGantt::ItemDelegate::State\\_ExtendRight](#).

```

345 {
346     StyleOptionGanttItem opt = getStyleOption();
347     m_istate = scene()->itemDelegate()->interactionStateFor( event->pos(), opt, index() );
348     m_presspos = event->pos();
349     m_pressscenepos = event->scenePos();
350     scene()->itemPressed( index() );
351
352     switch( m_istate ) {
353     case ItemDelegate::State_ExtendLeft:
354     case ItemDelegate::State_ExtendRight:
355     default: /* None and Move */
356         BASE::mousePressEvent( event );
357         break;
358     }
359 }

```

#### 12.15.4.16 void GraphicsItem::mouseReleaseEvent (QGraphicsSceneMouseEvent \*)

[protected]

Definition at line 361 of file kdganttgraphicsitem.cpp.

References [KDGantt::GraphicsView::addConstraint\(\)](#), [boundingRect\(\)](#), [index\(\)](#), [KDGantt::GraphicsScene::itemClicked\(\)](#), [rect\(\)](#), [scene\(\)](#), and [KDGantt::GraphicsScene::setDragSource\(\)](#).

```

362 {
363     if ( !m_presspos.isNull() ) {
364         scene()->itemClicked( index() );
365     }
366     delete m_dragline; m_dragline = 0;
367     if ( scene()->dragSource() ) {
368         // Create a new constraint
369         GraphicsItem* other = qgraphicsitem_cast<GraphicsItem*>( scene()->itemAt( event->scenePos() ) )
370         if ( other && scene()->dragSource() != other &&
371             other->mapToScene( other->rect() ).boundingRect().contains( event->scenePos() ) ) {
372             QGraphicsView* view = qobject_cast<QGraphicsView*>( event->widget()->parentWidget() );
373             if ( view ) {
374                 view->addConstraint( scene()->summaryHandlingModel()->mapToSource( scene()->dragSource() ),
375                                   scene()->summaryHandlingModel()->mapToSource( other->index() ),
376                                   other );
377             }
378             scene()->setDragSource( 0 );
379         } else {
380             updateItemFromMouse( event->scenePos() );
381             updateModel();
382         }
383     }
384     m_presspos = QPointF();
385     BASE::mouseReleaseEvent( event );
386 }

```

#### 12.15.4.17 void GraphicsItem::paint (QPainter \* painter, const QStyleOptionGraphicsItem \* option, QWidget \* widget = 0)

Definition at line 141 of file kdganttgraphicsitem.cpp.

References [boundingRect\(\)](#), [index\(\)](#), [KDGantt::GraphicsScene::itemDelegate\(\)](#), [KDGantt::ItemDelegate::paintGanttItem\(\)](#), and [scene\(\)](#).

```

143 {
144     Q_UNUSED( widget );
145     if ( boundingRect().isValid() && scene() ) {
146         StyleOptionGanttItem opt = getStyleOption();
147         *static_cast<QStyleOption*>(&opt) = *static_cast<const QStyleOption*>( option );
148         scene()->itemDelegate()->paintGanttItem( painter, opt, index() );
149     }
150 }

```

#### 12.15.4.18 QRectF KDGantt::GraphicsItem::rect () const

Definition at line 60 of file kdganttgraphicsitem.h.

Referenced by [mouseMoveEvent\(\)](#), [mouseReleaseEvent\(\)](#), [updateItem\(\)](#), and [KDGantt::GraphicsScene::updateItems\(\)](#).

```

60 { return m_rect; }

```

#### 12.15.4.19 void GraphicsItem::removeEndConstraint (ConstraintGraphicsItem \*)

Definition at line 207 of file kdganttgraphicsitem.cpp.

Referenced by [KDGantt::GraphicsScene::removeItem\(\)](#).

```
208 {
209     assert( item );
210     m_endConstraints.removeAll( item );
211 }
```

#### 12.15.4.20 void GraphicsItem::removeStartConstraint (ConstraintGraphicsItem \*)

Definition at line 201 of file kdganttgraphicsitem.cpp.

Referenced by KDGantt::GraphicsScene::removeItem().

```
202 {
203     assert( item );
204     m_startConstraints.removeAll( item );
205 }
```

#### 12.15.4.21 GraphicsScene \* GraphicsItem::scene () const

Definition at line 116 of file kdganttgraphicsitem.cpp.

Referenced by focusInEvent(), hoverMoveEvent(), isEditable(), itemChange(), mouseDoubleClickEvent(), mouseMoveEvent(), mousePressEvent(), mouseReleaseEvent(), paint(), and updateItem().

```
117 {
118     return qobject_cast<GraphicsScene*>( QGraphicsItem::scene() );
119 }
```

#### 12.15.4.22 void GraphicsItem::setBoundingRect (const QRectF & r)

Definition at line 129 of file kdganttgraphicsitem.cpp.

Referenced by updateItem().

```
130 {
131     prepareGeometryChange();
132     m_boundingrect = r;
133     update();
134 }
```

#### 12.15.4.23 void GraphicsItem::setIndex (const QPersistentModelIndex & idx)

Definition at line 152 of file kdganttgraphicsitem.cpp.

Referenced by updateItem(), and KDGantt::GraphicsScene::updateRow().

```
153 {
154     m_index=idx;
155     update();
156 }
```

**12.15.4.24 void GraphicsItem::setRect (const QRectF & r)**

Definition at line 121 of file kdganttgraphicsitem.cpp.

Referenced by updateItem().

```
122 {
123     prepareGeometryChange();
124     m_rect = r;
125     updateConstraintItems();
126     update();
127 }
```

**12.15.4.25 QList<ConstraintGraphicsItem\*> KDGantt::GraphicsItem::startConstraints () const**

Definition at line 76 of file kdganttgraphicsitem.h.

```
76 { return m_startConstraints; }
```

**12.15.4.26 int GraphicsItem::type () const**

Definition at line 96 of file kdganttgraphicsitem.cpp.

References Type.

```
97 {
98     return Type;
99 }
```

**12.15.4.27 void GraphicsItem::updateItem (const Span & rowgeometry, const QPersistentModelIndex & idx)**

Definition at line 227 of file kdganttgraphicsitem.cpp.

References boundingRect(), KDGantt::GraphicsScene::grid(), index(), KDGantt::ItemDelegate::itemBoundingSpan(), KDGantt::GraphicsScene::itemDelegate(), KDGantt::ItemTypeRole, KDGantt::Span::length(), KDGantt::AbstractGrid::mapToChart(), KDGantt::AbstractRowController::maximumItemHeight(), r, rect(), KDGantt::GraphicsScene::rowController(), scene(), setBoundingRect(), setIndex(), setRect(), KDGantt::Span::start(), and KDGantt::TypeMulti.

Referenced by KDGantt::GraphicsScene::updateItems(), and KDGantt::GraphicsScene::updateRow().

```
228 {
229     Updater updater( &m_isupdating );
230     //qDebug() << "GraphicsItem::updateItem("<<rowGeometry<<idx<<")";
231     if ( !idx.isValid() || idx.data( ItemTypeRole )==TypeMulti ) {
232         setRect( QRectF() );
233         hide();
234         return;
235     }
236
237     Span s = scene()->grid()->mapToChart( idx );
238     setPos( QPointF( s.start(), rowGeometry.start() ) );
239     setRect( QRectF( 0., 0., s.length(), rowGeometry.length() ) );
240     setIndex( idx );
```

```
241     Span bs = scene()->itemDelegate()->itemBoundingSpan( getStyleOption(), index() );
242     setBoundingRect( QRectF( bs.start(), 0., bs.length(), rowGeometry.length() ) );
243     int maxh = scene()->rowController()->maximumItemHeight();
244     if ( maxh < rowGeometry.length() ) {
245         QRectF r = rect();
246         Qt::Alignment align = getStyleOption().displayAlignment;
247         if ( align & Qt::AlignTop ) {
248             // Do nothing
249         } else if ( align & Qt::AlignBottom ) {
250             r.setY( rowGeometry.length()-maxh );
251         } else {
252             // Center
253             r.setY( ( rowGeometry.length()-maxh ) / 2. );
254         }
255         r.setHeight( maxh );
256         setRect( r );
257     }
258
259     scene()->setSceneRect( scene()->sceneRect().united( mapToScene( boundingRect() ).boundingRect() ) );
260     updateConstraintItems();
261 }
```

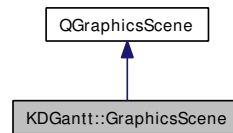
The documentation for this class was generated from the following files:

- [kdganttgraphicsitem.h](#)
- [kdganttgraphicsitem.cpp](#)

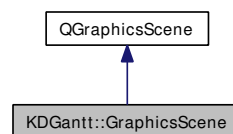
## 12.16 KDGantt::GraphicsScene Class Reference

```
#include <kdganttgraphicsscene.h>
```

Inheritance diagram for KDGantt::GraphicsScene:



Collaboration diagram for KDGantt::GraphicsScene:



### 12.16.1 Detailed Description

Definition at line 47 of file `kdganttgraphicsscene.h`.

#### Public Slots

- void [setConstraintModel](#) ([ConstraintModel](#) \*)
- void [setModel](#) ([QAbstractItemModel](#) \*)
- void [setReadOnly](#) (bool)
- void [setRootIndex](#) (const [QModelIndex](#) &idx)
- void [setSelectionModel](#) ([QItemSelectionModel](#) \*selectionmodel)
- void [setSummaryHandlingModel](#) ([QAbstractProxyModel](#) \*)

#### Signals

- void [clicked](#) (const [QModelIndex](#) &index)
- void [doubleClicked](#) (const [QModelIndex](#) &index)
- void [entered](#) (const [QModelIndex](#) &index)
- void [gridChanged](#) ()
- void [pressed](#) (const [QModelIndex](#) &index)

#### Public Member Functions

- void [clearConstraintItems](#) ()
- void [clearItems](#) ()
- [ConstraintModel](#) \* [constraintModel](#) () const
- [GraphicsItem](#) \* [createItem](#) ([ItemType](#) type) const
- void [deleteSubtree](#) (const [QModelIndex](#) &)

- [GraphicsItem](#) \* [dragSource](#) () const
- [ConstraintGraphicsItem](#) \* [findConstraintItem](#) (const [Constraint](#) &) const
- [QList](#)< [ConstraintGraphicsItem](#) \* > [findConstraintItems](#) (const [QModelIndex](#) &idx) const
- [GraphicsItem](#) \* [findItem](#) (const [QPersistentModelIndex](#) &) const
- [GraphicsItem](#) \* [findItem](#) (const [QModelIndex](#) &) const
- [GraphicsScene](#) ([QObject](#) \*parent=0)
- [AbstractGrid](#) \* [grid](#) () const
- void [insertItem](#) (const [QPersistentModelIndex](#) &, [GraphicsItem](#) \*)
- bool [isReadOnly](#) () const
- void [itemClicked](#) (const [QModelIndex](#) &)
- [ItemDelegate](#) \* [itemDelegate](#) () const
- void [itemDoubleClicked](#) (const [QModelIndex](#) &)
- void [itemEntered](#) (const [QModelIndex](#) &)
- void [itemPressed](#) (const [QModelIndex](#) &)
- [QAbstractItemModel](#) \* [model](#) () const
- void [print](#) ([QPainter](#) \*painter, const [QRectF](#) &target=[QRectF](#)(), bool drawRowLabels=true)
- void [removeItem](#) (const [QModelIndex](#) &)
- [QModelIndex](#) [rootIndex](#) () const
- [AbstractRowController](#) \* [rowController](#) () const
- [QItemSelectionModel](#) \* [selectionModel](#) () const
- void [setDragSource](#) ([GraphicsItem](#) \*item)
- void [setGrid](#) ([AbstractGrid](#) \*grid)
- void [setItemDelegate](#) ([ItemDelegate](#) \*)
- void [setRowController](#) ([AbstractRowController](#) \*rc)
- [QAbstractProxyModel](#) \* [summaryHandlingModel](#) () const
- void [updateItems](#) ()
- void [updateRow](#) (const [QModelIndex](#) &idx)
- virtual [~GraphicsScene](#) ()

## Static Public Member Functions

- static [QModelIndex](#) [dataIndex](#) (const [QModelIndex](#) &idx)
- static [QModelIndex](#) [mainIndex](#) (const [QModelIndex](#) &idx)

## Protected Member Functions

- void [drawBackground](#) ([QPainter](#) \*painter, const [QRectF](#) &rect)
- void [helpEvent](#) ([QGraphicsSceneHelpEvent](#) \*helpEvent)

## 12.16.2 Constructor & Destructor Documentation

### 12.16.2.1 [GraphicsScene::GraphicsScene](#) ([QObject](#) \*parent = 0) [explicit]

Definition at line 139 of file [kdganttgraphicsscene.cpp](#).

```

140     : QGraphicsScene( parent ), _d( new Private( this ) )
141 {
142     init();
143 }
```

### 12.16.2.2 GraphicsScene::~GraphicsScene () [virtual]

Definition at line 145 of file kdganttgraphicsscene.cpp.

References clearConstraintItems(), and clearItems().

```
146 {
147     clearConstraintItems();
148     clearItems();
149 }
```

## 12.16.3 Member Function Documentation

### 12.16.3.1 void GraphicsScene::clearConstraintItems ()

Definition at line 488 of file kdganttgraphicsscene.cpp.

Referenced by ~GraphicsScene().

```
489 {
490     // TODO
491     // d->constraintItems.clearConstraintItems();
492 }
```

### 12.16.3.2 void GraphicsScene::clearItems ()

Definition at line 457 of file kdganttgraphicsscene.cpp.

References d.

Referenced by ~GraphicsScene().

```
458 {
459     qDeleteAll( items() );
460     d->items.clear();
461 }
```

### 12.16.3.3 void KDGantt::GraphicsScene::clicked (const QModelIndex & *index*) [signal]

Referenced by itemClicked().

### 12.16.3.4 ConstraintModel \* GraphicsScene::constraintModel () const

Definition at line 209 of file kdganttgraphicsscene.cpp.

References d.

```
210 {
211     return d->constraintModel;
212 }
```

### 12.16.3.5 GraphicsItem \* GraphicsScene::createItem (ItemType type) const

Creates a new item of type type. TODO: If the user should be allowed to override this in any way, it needs to be in View!

Definition at line 313 of file kdganttgraphicsscene.cpp.

References KDGantt::TypeEvent, KDGantt::TypeSummary, and KDGantt::TypeTask.

Referenced by updateRow().

```

314 {
315 #if 0
316     switch( type ) {
317         case TypeEvent:    return 0;
318         case TypeTask:    return new TaskItem;
319         case TypeSummary: return new SummaryItem;
320         default:          return 0;
321     }
322 #endif
323     //qDebug() << "GraphicsScene::createItem("<<type<<")";
324     Q_UNUSED( type );
325     return new GraphicsItem;
326 }

```

### 12.16.3.6 QModelIndex GraphicsScene::dataIndex (const QModelIndex & idx) [static]

Returns the index pointing to the last column in the same row as idx. This can be thought of as in "inverse" of [mainIndex\(\)](#)

Definition at line 295 of file kdganttgraphicsscene.cpp.

References model().

Referenced by deleteSubtree().

```

296 {
297 #if 0
298     if ( idx.isValid() ) {
299         const QAbstractItemModel* model = idx.model();
300         return model->index( idx.row(), model->columnCount( idx.parent() )-1,idx.parent() );
301     } else {
302         return QModelIndex();
303     }
304 #else
305     return idx;
306 #endif
307 }

```

### 12.16.3.7 void GraphicsScene::deleteSubtree (const QModelIndex &)

Definition at line 473 of file kdganttgraphicsscene.cpp.

References dataIndex(), removeItem(), and summaryHandlingModel().

```

474 {
475     QModelIndex idx = dataIndex( _idx );
476     removeItem( idx );
477     for ( int i = 0; i < summaryHandlingModel()->rowCount( _idx ); ++i ) {
478         deleteSubtree( summaryHandlingModel()->index( i, summaryHandlingModel()->columnCount( _idx )-1,
479     }
480 }

```

### 12.16.3.8 void KDGantt::GraphicsScene::doubleClicked (const QModelIndex & *index*) [signal]

Referenced by itemDoubleClicked().

### 12.16.3.9 GraphicsItem \* GraphicsScene::dragSource () const

Definition at line 555 of file kdganttgraphicsscene.cpp.

References d.

```
556 {
557     return d->dragSource;
558 }
```

### 12.16.3.10 void GraphicsScene::drawBackground (QPainter \* *painter*, const QRectF & *rect*) [protected]

Definition at line 525 of file kdganttgraphicsscene.cpp.

References d.

```
526 {
527     d->grid->paintGrid( painter, sceneRect(), rect, d->rowController );
528 }
```

### 12.16.3.11 void KDGantt::GraphicsScene::entered (const QModelIndex & *index*) [signal]

Referenced by itemEntered().

### 12.16.3.12 ConstraintGraphicsItem \* GraphicsScene::findConstraintItem (const Constraint & *c*) const

Definition at line 483 of file kdganttgraphicsscene.cpp.

References c, and d.

```
484 {
485     return d->findConstraintItem( c );
486 }
```

### 12.16.3.13 QList<ConstraintGraphicsItem\*> KDGantt::GraphicsScene::findConstraintItems (const QModelIndex & *idx*) const

### 12.16.3.14 GraphicsItem \* GraphicsScene::findItem (const QPersistentModelIndex & *idx*) const

Definition at line 449 of file kdganttgraphicsscene.cpp.

References d, and summaryHandlingModel().

```

450 {
451     if ( !idx.isValid() ) return 0;
452     assert( idx.model() == summaryHandlingModel() );
453     QHash<QPersistentModelIndex,GraphicsItem*>::const_iterator it = d->items.find( idx );
454     return ( it != d->items.end() )?*it:0;
455 }

```

### 12.16.3.15 **GraphicsItem \* GraphicsScene::findItem (const QModelIndex &) const**

Definition at line 441 of file kdganttgraphicsscene.cpp.

References `d`, and `summaryHandlingModel()`.

Referenced by `updateRow()`.

```

442 {
443     if ( !idx.isValid() ) return 0;
444     assert( idx.model() == summaryHandlingModel() );
445     QHash<QPersistentModelIndex,GraphicsItem*>::const_iterator it = d->items.find( idx );
446     return ( it != d->items.end() )?*it:0;
447 }

```

### 12.16.3.16 **AbstractGrid \* GraphicsScene::grid () const**

Definition at line 258 of file kdganttgraphicsscene.cpp.

References `d`.

Referenced by `setGrid()`, and `KDGantt::GraphicsItem::updateItem()`.

```

259 {
260     return d->grid;
261 }

```

### 12.16.3.17 **void KDGantt::GraphicsScene::gridChanged () [signal]**

Referenced by `setGrid()`.

### 12.16.3.18 **void GraphicsScene::helpEvent (QGraphicsSceneHelpEvent \* helpEvent)** [protected]

Definition at line 511 of file kdganttgraphicsscene.cpp.

```

512 {
513     #ifndef QT_NO_TOOLTIP
514         QGraphicsItem *item = itemAt( helpEvent->scenePos() );
515         if ( GraphicsItem* gitem = qgraphicsitem_cast<GraphicsItem*>( item ) ) {
516             QToolTip::showText( helpEvent->screenPos(), gitem->ganttToolTip() );
517         } else if ( ConstraintGraphicsItem* citem = qgraphicsitem_cast<ConstraintGraphicsItem*>( item ) )
518             QToolTip::showText( helpEvent->screenPos(), citem->ganttToolTip() );
519         } else {
520             QGraphicsScene::helpEvent( helpEvent );
521         }
522     #endif /* QT_NO_TOOLTIP */
523 }

```

**12.16.3.19 void GraphicsScene::insertItem (const QPersistentModelIndex &, GraphicsItem \*)**

Definition at line 376 of file kdganttgraphicsscene.cpp.

References KDGantt::GraphicsItem::addEndConstraint(), KDGantt::GraphicsItem::addStartConstraint(), c, d, and summaryHandlingModel().

Referenced by updateRow().

```

377 {
378     if ( !d->constraintModel.isNull() ) {
379         // Create items for constraints
380         const QModelIndex sidx = summaryHandlingModel()->mapToSource( idx );
381         const QList<Constraint> clst = d->constraintModel->constraintsForIndex( sidx );
382         Q_FOREACH( Constraint c, clst ) {
383             QModelIndex other_idx;
384             if ( c.startIndex() == sidx ) {
385                 other_idx = c.endIndex();
386                 GraphicsItem* other_item = d->items.value(summaryHandlingModel()->mapFromSource( other_idx ));
387                 if ( !other_item ) continue;
388                 ConstraintGraphicsItem* citem = new ConstraintGraphicsItem( c );
389                 item->addStartConstraint( citem );
390                 other_item->addEndConstraint( citem );
391                 addItem( citem );
392             } else if ( c.endIndex() == sidx ) {
393                 other_idx = c.startIndex();
394                 GraphicsItem* other_item = d->items.value(summaryHandlingModel()->mapFromSource( other_idx ));
395                 if ( !other_item ) continue;
396                 ConstraintGraphicsItem* citem = new ConstraintGraphicsItem( c );
397                 other_item->addStartConstraint( citem );
398                 item->addEndConstraint( citem );
399                 addItem( citem );
400             } else {
401                 assert( 0 ); // Impossible
402             }
403         }
404     }
405     d->items.insert( idx, item );
406     addItem( item );
407 }

```

**12.16.3.20 bool GraphicsScene::isReadOnly () const**

Definition at line 268 of file kdganttgraphicsscene.cpp.

References d.

Referenced by KDGantt::GraphicsItem::isEditable().

```

269 {
270     return d->readOnly;
271 }

```

**12.16.3.21 void GraphicsScene::itemClicked (const QModelIndex &)**

Definition at line 540 of file kdganttgraphicsscene.cpp.

References clicked().

Referenced by KDGantt::GraphicsItem::mousePressEvent().

```
541 {  
542     emit clicked( idx );  
543 }
```

### 12.16.3.22 `ItemDelegate * GraphicsScene::itemDelegate () const`

Definition at line 169 of file `kdganttgraphicsscene.cpp`.

References `d`.

Referenced by `KDGantt::ConstraintGraphicsItem::boundingRect()`, `KDGantt::GraphicsItem::hoverMoveEvent()`, `KDGantt::GraphicsItem::mouseDoubleClickEvent()`, `KDGantt::GraphicsItem::mousePressEvent()`, `KDGantt::GraphicsItem::paint()`, `KDGantt::ConstraintGraphicsItem::paint()`, and `KDGantt::GraphicsItem::updateItem()`.

```
170 {  
171     return d->itemDelegate;  
172 }
```

### 12.16.3.23 `void GraphicsScene::itemDoubleClicked (const QModelIndex &)`

Definition at line 545 of file `kdganttgraphicsscene.cpp`.

References `doubleClicked()`.

Referenced by `KDGantt::GraphicsItem::mouseDoubleClickEvent()`.

```
546 {  
547     emit doubleClicked( idx );  
548 }
```

### 12.16.3.24 `void GraphicsScene::itemEntered (const QModelIndex &)`

Definition at line 530 of file `kdganttgraphicsscene.cpp`.

References `entered()`.

Referenced by `KDGantt::GraphicsItem::hoverMoveEvent()`.

```
531 {  
532     emit entered( idx );  
533 }
```

### 12.16.3.25 `void GraphicsScene::itemPressed (const QModelIndex &)`

Definition at line 535 of file `kdganttgraphicsscene.cpp`.

References `pressed()`.

Referenced by `KDGantt::GraphicsItem::mousePressEvent()`.

```
536 {  
537     emit pressed( idx );  
538 }
```

**12.16.3.26 QModelIndex GraphicsScene::mainIndex (const QModelIndex & idx)** [static]

Definition at line 278 of file kdganttgraphicsscene.cpp.

```

279 {
280 #if 0
281     if ( idx.isValid() ) {
282         return idx.model()->index( idx.row(), 0,idx.parent() );
283     } else {
284         return QModelIndex();
285     }
286 #else
287     return idx;
288 #endif
289 }
```

**12.16.3.27 QAbstractItemModel \* GraphicsScene::model () const**

Definition at line 174 of file kdganttgraphicsscene.cpp.

References d.

Referenced by dataIndex(), setGrid(), setSummaryHandlingModel(), and updateRow().

```

175 {
176     assert(!d->summaryHandlingModel.isNull());
177     return d->summaryHandlingModel->sourceModel();
178 }
```

**12.16.3.28 void KDGantt::GraphicsScene::pressed (const QModelIndex & index)** [signal]

Referenced by itemPressed().

**12.16.3.29 void GraphicsScene::print (QPainter \* painter, const QRectF & target = QRectF (), bool drawRowLabels = true)**

Definition at line 560 of file kdganttgraphicsscene.cpp.

References d, KDGantt::GraphicsItem::index(), KDGantt::ItemTypeRole, KDGantt::Span::length(), rowController(), KDGantt::AbstractRowController::rowGeometry(), KDGantt::Span::start(), summaryHandlingModel(), and KDGantt::TypeMulti.

```

561 {
562     QRectF targetRect(target);
563
564     assert(rowController());
565
566     QVector<QGraphicsTextItem*> labelItems;
567     if(drawRowLabels) {
568         labelItems.reserve(d->items.size());
569         qreal textWidth = 0.;
570         qreal rowHeight = 0.;
571         {Q_FOREACH( GraphicsItem* item, d->items ) {
572             QModelIndex sidx = summaryHandlingModel()->mapToSource( item->index() );
573             if( sidx.parent().isValid() && sidx.parent().data( ItemTypeRole ).toInt() == TypeMulti ) {
574                 continue;
575             }

```

```

576     const Span rg=rowController()->rowGeometry( sidx );
577     const QString txt = item->index().data( Qt::DisplayRole ).toString();
578     QGraphicsTextItem* ti = new QGraphicsTextItem(txt,0,this);
579     ti->setPos( 0, rg.start() );
580     if( ti->document()->size().width() > textWidth ) textWidth = ti->document()->size().width();
581     if( rg.length() > rowHeight ) rowHeight = rg.length();
582     labelItems << ti;
583 }
584 {Q_FOREACH( QGraphicsTextItem* item, labelItems ) {
585     item->setPos( -textWidth-rowHeight, item->pos().y() );
586     item->show();
587 }}
588 }
589 QRectF oldSceneRect( sceneRect() );
590 setSceneRect( itemsBoundingRect() );
591 if(targetRect.isNull()) targetRect = sceneRect();
592 render( painter, target );
593 qDeleteAll(labelItems);
594
595 setSceneRect( oldSceneRect );
596 }

```

### 12.16.3.30 void GraphicsScene::removeItem (const QModelIndex &)

Definition at line 409 of file kdganttgraphicsscene.cpp.

References [KDGantt::ConstraintGraphicsItem::constraint\(\)](#), [d](#), [KDGantt::Constraint::endIndex\(\)](#), [KDGantt::GraphicsItem::removeEndConstraint\(\)](#), [KDGantt::GraphicsItem::removeStartConstraint\(\)](#), and [summaryHandlingModel\(\)](#).

Referenced by [deleteSubtree\(\)](#), and [updateRow\(\)](#).

```

410 {
411     //qDebug() << "GraphicsScene::removeItem("<<idx<<")";
412     QHash<QPersistentModelIndex,GraphicsItem*>::iterator it = d->items.find( idx );
413     if ( it != d->items.end() ) {
414         {
415             // Remove any constraintitems starting here
416             const QList<ConstraintGraphicsItem*> clst = ( *it )->startConstraints();
417             qDebug() << clst;
418             Q_FOREACH( ConstraintGraphicsItem* citeM, clst ) {
419                 GraphicsItem* end_item = d->items.value( summaryHandlingModel()->mapFromSource( citeM-
420                 if ( end_item ) end_item->removeEndConstraint( citeM );
421                 d->constraintModel->removeConstraint( citeM->constraint() );
422                 delete citeM;
423             }
424         }
425         {// Remove any constraintitems ending here
426             const QList<ConstraintGraphicsItem*> clst = ( *it )->endConstraints();
427             qDebug() << clst;
428             Q_FOREACH( ConstraintGraphicsItem* citeM, clst ) {
429                 GraphicsItem* end_item = d->items.value( summaryHandlingModel()->mapFromSource( citeM-
430                 if ( end_item ) end_item->removeStartConstraint( citeM );
431                 d->constraintModel->removeConstraint( citeM->constraint() );
432                 delete citeM;
433             }
434         }
435         // Get rid of the item
436         delete *it;
437         d->items.erase( it );
438     }
439 }

```

### 12.16.3.31 QModelIndex GraphicsScene::rootIndex () const

Definition at line 204 of file kdganttgraphicsscene.cpp.

References d.

```
205 {
206     return d->grid->rootIndex();
207 }
```

### 12.16.3.32 AbstractRowController \* GraphicsScene::rowController () const

Definition at line 242 of file kdganttgraphicsscene.cpp.

References d.

Referenced by print(), KD Gantt::GraphicsItem::updateItem(), and updateRow().

```
243 {
244     return d->rowController;
245 }
```

### 12.16.3.33 QItemSelectionModel \* GraphicsScene::selectionModel () const

Definition at line 232 of file kdganttgraphicsscene.cpp.

References d.

Referenced by KD Gantt::GraphicsItem::focusInEvent(), KD Gantt::GraphicsItem::itemChange(), and KD Gantt::GraphicsItem::mouseMoveEvent().

```
233 {
234     return d->selectionModel;
235 }
```

### 12.16.3.34 void GraphicsScene::setConstraintModel (ConstraintModel \*) [slot]

Definition at line 214 of file kdganttgraphicsscene.cpp.

References d.

```
215 {
216     if ( !d->constraintModel.isNull() ) {
217         disconnect( d->constraintModel );
218     }
219     d->constraintModel = cm;
220
221     connect( cm, SIGNAL( constraintAdded( const Constraint& ) ), this, SLOT( slotConstraintAdded( const Constraint& ) ) );
222     connect( cm, SIGNAL( constraintRemoved( const Constraint& ) ), this, SLOT( slotConstraintRemoved( const Constraint& ) ) );
223     d->resetConstraintItems();
224 }
```

**12.16.3.35 void GraphicsScene::setDragSource (GraphicsItem \* item)**

Definition at line 550 of file kdganttgraphicsscene.cpp.

References `d`.

Referenced by `KDGantt::GraphicsItem::mouseMoveEvent()`, and `KDGantt::GraphicsItem::mouseReleaseEvent()`.

```
551 {
552     d->dragSource = item;
553 }
```

**12.16.3.36 void GraphicsScene::setGrid (AbstractGrid \* grid)**

Definition at line 247 of file kdganttgraphicsscene.cpp.

References `d`, `grid()`, `gridChanged()`, and `model()`.

```
248 {
249     QAbstractItemModel* model = d->grid->model();
250     if ( grid == 0 ) grid = &d->default_grid;
251     if ( d->grid ) disconnect( d->grid );
252     d->grid = grid;
253     connect( d->grid, SIGNAL( gridChanged() ), this, SLOT( slotGridChanged() ) );
254     d->grid->setModel( model );
255     slotGridChanged();
256 }
```

**12.16.3.37 void GraphicsScene::setItemDelegate (ItemDelegate \*)**

Definition at line 162 of file kdganttgraphicsscene.cpp.

References `d`.

```
163 {
164     if ( !d->itemDelegate.isNull() && d->itemDelegate->parent()==this ) delete d->itemDelegate;
165     d->itemDelegate = delegate;
166     update();
167 }
```

**12.16.3.38 void GraphicsScene::setModel (QAbstractItemModel \*) [slot]**

Definition at line 180 of file kdganttgraphicsscene.cpp.

References `d`, and `setSelectionModel()`.

```
181 {
182     assert(!d->summaryHandlingModel.isNull());
183     d->summaryHandlingModel->setSourceModel(model);
184     d->grid->setModel( d->summaryHandlingModel );
185     setSelectionModel( new QItemSelectionModel( model, this ) );
186 }
```

**12.16.3.39 void GraphicsScene::setReadOnly (bool) [slot]**

Definition at line 263 of file kdganttgraphicsscene.cpp.

References [d](#).

```
264 {
265     d->readOnly = ro;
266 }
```

**12.16.3.40 void GraphicsScene::setRootIndex (const QModelIndex & idx) [slot]**

Definition at line 199 of file kdganttgraphicsscene.cpp.

References [d](#).

```
200 {
201     d->grid->setRootIndex( idx );
202 }
```

**12.16.3.41 void GraphicsScene::setRowController (AbstractRowController \* rc)**

Definition at line 237 of file kdganttgraphicsscene.cpp.

References [d](#).

```
238 {
239     d->rowController = rc;
240 }
```

**12.16.3.42 void GraphicsScene::setSelectionModel (QItemSelectionModel \* selectionmodel) [slot]**

Definition at line 226 of file kdganttgraphicsscene.cpp.

References [d](#).

Referenced by [setModel\(\)](#).

```
227 {
228     d->selectionModel = smodel;
229     // TODO: update selection from model and connect signals
230 }
```

**12.16.3.43 void GraphicsScene::setSummaryHandlingModel (QAbstractProxyModel \*) [slot]**

Definition at line 193 of file kdganttgraphicsscene.cpp.

References [d](#), and [model\(\)](#).

```
194 {
195     proxyModel->setSourceModel( model() );
196     d->summaryHandlingModel = proxyModel;
197 }
```

**12.16.3.44 QAbstractProxyModel \* GraphicsScene::summaryHandlingModel () const**

Definition at line 188 of file kdganttgraphicsscene.cpp.

References `d`.

Referenced by `deleteSubtree()`, `findItem()`, `insertItem()`, `print()`, `KDGantt::ConstraintGraphicsItem::proxy-Constraint()`, `removeItem()`, and `updateRow()`.

```
189 {
190     return d->summaryHandlingModel;
191 }
```

**12.16.3.45 void GraphicsScene::updateItems ()**

Definition at line 463 of file kdganttgraphicsscene.cpp.

References `d`, `KDGantt::GraphicsItem::rect()`, and `KDGantt::GraphicsItem::updateItem()`.

```
464 {
465     for ( QHash<QPersistentModelIndex,GraphicsItem*>::iterator it = d->items.begin();
466          it != d->items.end(); ++it ) {
467         GraphicsItem* const item = it.value();
468         const QPersistentModelIndex& idx = it.key();
469         item->updateItem( Span( item->pos().y(), item->rect().height() ), idx );
470     }
471 }
```

**12.16.3.46 void GraphicsScene::updateRow (const QModelIndex & idx)**

Definition at line 328 of file kdganttgraphicsscene.cpp.

References `KDGantt::GraphicsItem::boundingRect()`, `createItem()`, `findItem()`, `insertItem()`, `KDGantt::ItemTypeRole`, `model()`, `removeItem()`, `rowController()`, `KDGantt::AbstractRowController::rowGeometry()`, `KDGantt::GraphicsItem::setIndex()`, `summaryHandlingModel()`, `KDGantt::TypeMulti`, `KDGantt::TypeNone`, and `KDGantt::GraphicsItem::updateItem()`.

```
329 {
330     //qDebug() << "GraphicsScene::updateRow("<<rowidx<<")";
331     if ( !rowidx.isValid() ) return;
332     const QAbstractItemModel* model = rowidx.model(); // why const?
333     assert( model );
334     assert( rowController() );
335     assert( model == summaryHandlingModel() );
336
337     const QModelIndex sidx = summaryHandlingModel()->mapToSource( rowidx );
338     const Span rg=rowController()->rowGeometry( sidx );
339     for ( int col = 0; col < summaryHandlingModel()->columnCount( rowidx.parent() ); ++col ) {
340         const QModelIndex idx = summaryHandlingModel()->index( rowidx.row(), col, rowidx.parent() );
341         const int itemtype = summaryHandlingModel()->data( idx, ItemTypeRole ).toInt();
342         if ( itemtype == TypeNone ) {
343             removeItem( idx );
344             continue;
345         }
346         if ( itemtype == TypeMulti ) {
347             int cr=0;
348             QModelIndex child;
349             while ( ( child = idx.child( cr, 0 ) ).isValid() ) {
350                 GraphicsItem* item = findItem( child );
```

```
351         if (!item) {
352             item = createItem( static_cast<ItemType>( itemtype ) );
353             item->setIndex( child );
354             insertItem( child, item);
355         }
356         item->updateItem( rg, child );
357         setSceneRect( sceneRect().united( item->boundingRect() ) );
358         ++cr;
359     }
360 }
361 {
362     if ( summaryHandlingModel()->data( rowidx.parent(), ItemTypeRole ).toInt() == TypeMulti )
363
364     GraphicsItem* item = findItem( idx );
365     if (!item) {
366         item = createItem( static_cast<ItemType>( itemtype ) );
367         item->setIndex( idx );
368         insertItem(idx, item);
369     }
370     item->updateItem( rg, idx );
371     setSceneRect( sceneRect().united( item->boundingRect() ) );
372 }
373 }
374 }
```

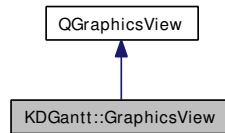
The documentation for this class was generated from the following files:

- [kdganttgraphicsscene.h](#)
- [kdganttgraphicsscene.cpp](#)

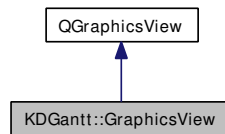
## 12.17 KDGantt::GraphicsView Class Reference

```
#include <KDGanttGraphicsView>
```

Inheritance diagram for KDGantt::GraphicsView:



Collaboration diagram for KDGantt::GraphicsView:



### 12.17.1 Detailed Description

The [GraphicsView](#) class provides a model/view implementation of a gantt chart.

Definition at line 44 of file `kdganttgraphicsview.h`.

#### Public Slots

- void [setConstraintModel](#) ([ConstraintModel](#) \*)
- void [setGrid](#) ([AbstractGrid](#) \*)
- void [setItemDelegate](#) ([ItemDelegate](#) \*delegate)
- void [setModel](#) ([QAbstractItemModel](#) \*)
- void [setReadOnly](#) (bool)
- void [setRootIndex](#) (const [QModelIndex](#) &)
- void [setRowController](#) ([AbstractRowController](#) \*)
- void [setSelectionModel](#) ([QItemSelectionModel](#) \*)
- void [setSummaryHandlingModel](#) ([QAbstractProxyModel](#) \*model)

#### Signals

- void [activated](#) (const [QModelIndex](#) &index)
- void [clicked](#) (const [QModelIndex](#) &index)
- void [doubleClicked](#) (const [QModelIndex](#) &index)
- void [entered](#) (const [QModelIndex](#) &index)
- void [pressed](#) (const [QModelIndex](#) &index)

## Public Member Functions

- virtual void [addConstraint](#) (const QModelIndex &from, const QModelIndex &to, Qt::KeyboardModifiers modifiers)
- void [clearItems](#) ()
- [ConstraintModel](#) \* [constraintModel](#) () const
- void [deleteSubtree](#) (const QModelIndex &)
- [GraphicsView](#) ([QWidget](#) \*parent=0)
- [AbstractGrid](#) \* [grid](#) () const
- QModelIndex [indexAt](#) (const QPoint &pos) const
- bool [isReadOnly](#) () const
- [ItemDelegate](#) \* [itemDelegate](#) () const
- [QAbstractItemModel](#) \* [model](#) () const
- void [print](#) ([QPainter](#) \*painter, const QRectF &target=QRectF(), bool drawRowLabels=true)
- [Q\\_PRIVATE\\_SLOT](#) (d, void slotItemDoubleClicked(const QModelIndex &idx))
- [Q\\_PRIVATE\\_SLOT](#) (d, void slotItemClicked(const QModelIndex &idx))
- [Q\\_PRIVATE\\_SLOT](#) (d, void slotRowsRemoved(const QModelIndex &parent, int start, int end))
- [Q\\_PRIVATE\\_SLOT](#) (d, void slotRowsAboutToBeRemoved(const QModelIndex &parent, int start, int end))
- [Q\\_PRIVATE\\_SLOT](#) (d, void slotRowsInserted(const QModelIndex &parent, int start, int end))
- [Q\\_PRIVATE\\_SLOT](#) (d, void slotModelReset())
- [Q\\_PRIVATE\\_SLOT](#) (d, void slotLayoutChanged())
- [Q\\_PRIVATE\\_SLOT](#) (d, void slotDataChanged(const QModelIndex &topLeft, const QModelIndex &bottomRight))
- [Q\\_PRIVATE\\_SLOT](#) (d, void slotColumnsRemoved(const QModelIndex &parent, int start, int end))
- [Q\\_PRIVATE\\_SLOT](#) (d, void slotColumnsInserted(const QModelIndex &parent, int start, int end))
- [Q\\_PRIVATE\\_SLOT](#) (d, void slotHorizontalScrollValueChanged(int))
- [Q\\_PRIVATE\\_SLOT](#) (d, void slotGridChanged())
- QModelIndex [rootIndex](#) () const
- [AbstractRowController](#) \* [rowController](#) () const
- [QItemSelectionModel](#) \* [selectionModel](#) () const
- [QAbstractProxyModel](#) \* [summaryHandlingModel](#) () const
- void [updateRow](#) (const QModelIndex &)
- void [updateScene](#) ()
- void [updateSceneRect](#) ()
- virtual [~GraphicsView](#) ()

## Protected Member Functions

- void [resizeEvent](#) ([QResizeEvent](#) \*)

## Properties

- bool [readOnly](#) []

## 12.17.2 Constructor & Destructor Documentation

### 12.17.2.1 GraphicsView::GraphicsView (QWidget \*parent = 0) [explicit]

Constructor. Creates a new [KDGantt::GraphicsView](#) with parent *parent*.

Definition at line 294 of file `kdganttgraphicsview.cpp`.

References `clicked()`, `doubleClicked()`, `entered()`, `pressed()`, and `setSummaryHandlingModel()`.

```

295     : BASE( parent ), _d( new Private( this ) )
296 {
297
298 #if defined KDAB_EVAL
299     EvalDialog::checkEvalLicense( "KD Gantt" );
300 #endif
301     connect( horizontalScrollBar(), SIGNAL( valueChanged( int ) ),
302             this, SLOT( slotHorizontalScrollValueChanged( int ) ) );
303     connect( &_d->scene, SIGNAL( gridChanged() ),
304             this, SLOT( slotGridChanged() ) );
305     connect( &_d->scene, SIGNAL( entered( const QModelIndex& ) ),
306             this, SIGNAL( entered( const QModelIndex& ) ) );
307     connect( &_d->scene, SIGNAL( pressed( const QModelIndex& ) ),
308             this, SIGNAL( pressed( const QModelIndex& ) ) );
309     connect( &_d->scene, SIGNAL( clicked( const QModelIndex& ) ),
310             this, SLOT( slotItemClicked( const QModelIndex& ) ) );
311     connect( &_d->scene, SIGNAL( doubleClicked( const QModelIndex& ) ),
312             this, SLOT( slotItemDoubleClicked( const QModelIndex& ) ) );
313     setScene( &_d->scene );
314
315     // HACK!
316     setSummaryHandlingModel( _d->scene.summaryHandlingModel() );
317 }
```

### 12.17.2.2 GraphicsView::~GraphicsView () [virtual]

Destroys this view.

Definition at line 320 of file `kdganttgraphicsview.cpp`.

```

321 {
322     delete _d;
323 }
```

## 12.17.3 Member Function Documentation

### 12.17.3.1 void KDGantt::GraphicsView::activated (const QModelIndex &index) [signal]

### 12.17.3.2 void GraphicsView::addConstraint (const QModelIndex &from, const QModelIndex &to, Qt::KeyboardModifiers modifiers) [virtual]

Adds a constraint from *from* to *to*. *modifiers* are the keyboard modifiers pressed by the user when the action is invoked.

Override this to control how constraints are added. The default implementation adds a soft constraint unless the Shift key is pressed, in that case it adds a hard constraint. If a constraint is already present, it is removed and nothing is added.

Definition at line 502 of file `kdganttgraphicsview.cpp`.

References KDGantt::ConstraintModel::addConstraint(), `c`, `constraintModel()`, KDGantt::ConstraintModel::hasConstraint(), `isReadOnly()`, KDGantt::ConstraintModel::removeConstraint(), KDGantt::Constraint::TypeHard, and KDGantt::Constraint::TypeSoft.

Referenced by KDGantt::GraphicsItem::mousePressEvent().

```

505 {
506     if ( isReadOnly() ) return;
507     ConstraintModel* cmodel = constraintModel();
508     assert( cmodel );
509     Constraint c( from, to, ( modifiers&Qt::ShiftModifier )?Constraint::TypeHard:Constraint::TypeSoft
510     if ( cmodel->hasConstraint( c ) ) cmodel->removeConstraint( c );
511     else cmodel->addConstraint( c );
512 }
```

### 12.17.3.3 void GraphicsView::clearItems ()

Definition at line 546 of file `kdganttgraphicsview.cpp`.

References `d`.

Referenced by `updateScene()`.

```

547 {
548     d->scene.clearItems();
549 }
```

### 12.17.3.4 void KDGantt::GraphicsView::clicked (const QModelIndex & *index*) [signal]

Referenced by `GraphicsView()`.

### 12.17.3.5 ConstraintModel \* GraphicsView::constraintModel () const

#### Returns:

the [KDGantt::ConstraintModel](#) displayed by this view.

Definition at line 391 of file `kdganttgraphicsview.cpp`.

References `d`.

Referenced by `addConstraint()`.

```

392 {
393     return d->scene.constraintModel();
394 }
```

### 12.17.3.6 void GraphicsView::deleteSubtree (const QModelIndex &)

Definition at line 597 of file `kdganttgraphicsview.cpp`.

References `d`.

```

598 {
599     d->scene.deleteSubtree( d->scene.summaryHandlingModel()->mapFromSource( idx ) );
600 }
```

### 12.17.3.7 void KDGantt::GraphicsView::doubleClicked (const QModelIndex & *index*) [signal]

Referenced by GraphicsView().

### 12.17.3.8 void KDGantt::GraphicsView::entered (const QModelIndex & *index*) [signal]

Referenced by GraphicsView().

### 12.17.3.9 AbstractGrid \* GraphicsView::grid () const

#### Returns:

the [AbstractGrid](#) used by this view.

Definition at line 474 of file kdganttgraphicsview.cpp.

References [d](#).

Referenced by [setGrid\(\)](#).

```
475 {
476     return d->scene.grid();
477 }
```

### 12.17.3.10 QModelIndex GraphicsView::indexAt (const QPoint & *pos*) const

#### Returns:

The QModelIndex for the item located at position *pos* in the view or an invalid index if no item was present at that position.

This is useful for for example contextmenus.

Definition at line 535 of file kdganttgraphicsview.cpp.

References [d](#).

```
536 {
537     QGraphicsItem* item = itemAt( pos );
538     if ( QGraphicsItem* gitem = qgraphicsitem_cast<GraphicsItem*>( item ) ) {
539         return d->scene.summaryHandlingModel()->mapToSource( gitem->index() );
540     } else {
541         return QModelIndex();
542     }
543 }
```

### 12.17.3.11 bool GraphicsView::isReadOnly () const

#### Returns:

true iff the view is in read-only mode

Definition at line 489 of file kdganttgraphicsview.cpp.

References d.

Referenced by addConstraint().

```
490 {  
491     return d->scene.isReadOnly();  
492 }
```

### 12.17.3.12 [ItemDelegate](#) \* GraphicsView::itemDelegate () const

#### Returns:

the [ItemDelegate](#) used by this view to render items

Definition at line 436 of file kdganttgraphicsview.cpp.

References d.

```
437 {  
438     return d->scene.itemDelegate();  
439 }
```

### 12.17.3.13 [QAbstractItemModel](#) \* GraphicsView::model () const

#### Returns:

the current model displayed by this view

Definition at line 348 of file kdganttgraphicsview.cpp.

References d.

Referenced by updateScene().

```
349 {  
350     return d->scene.model();  
351 }
```

### 12.17.3.14 void KDGantt::GraphicsView::pressed (const QModelIndex & *index*) [signal]

Referenced by GraphicsView().

### 12.17.3.15 void GraphicsView::print (QPainter \* *painter*, const QRectF & *target* = QRectF(), bool *drawRowLabels* = true)

Render the GanttView inside the rectangle *target* using the painter *painter*. This is useful for printing. If *target* is a null rect, the dimensions of painter's paint device will be used. If *drawRowLabels* is true, an additional per-row label is drawn, mimicing the look of a listview.

Definition at line 608 of file kdganttgraphicsview.cpp.

References d.

```

609 {
610     d->scene.print (painter, target, drawRowLabels);
611 }

```

**12.17.3.16** **KDGantt::GraphicsView::Q\_PRIVATE\_SLOT** (d, void *slotItemDoubleClicked*(const QModelIndex &idx))

**12.17.3.17** **KDGantt::GraphicsView::Q\_PRIVATE\_SLOT** (d, void *slotItemClicked*(const QModelIndex &idx))

**12.17.3.18** **KDGantt::GraphicsView::Q\_PRIVATE\_SLOT** (d, void *slotRowsRemoved*(const QModelIndex &parent, int start, int end))

**12.17.3.19** **KDGantt::GraphicsView::Q\_PRIVATE\_SLOT** (d, void *slotRowsAboutToBeRemoved*(const QModelIndex &parent, int start, int end))

**12.17.3.20** **KDGantt::GraphicsView::Q\_PRIVATE\_SLOT** (d, void *slotRowsInserted*(const QModelIndex &parent, int start, int end))

**12.17.3.21** **KDGantt::GraphicsView::Q\_PRIVATE\_SLOT** (d, void *slotModelReset*())

**12.17.3.22** **KDGantt::GraphicsView::Q\_PRIVATE\_SLOT** (d, void *slotLayoutChanged*())

**12.17.3.23** **KDGantt::GraphicsView::Q\_PRIVATE\_SLOT** (d, void *slotDataChanged*(const QModelIndex &topLeft, const QModelIndex &bottomRight))

**12.17.3.24** **KDGantt::GraphicsView::Q\_PRIVATE\_SLOT** (d, void *slotColumnsRemoved*(const QModelIndex &parent, int start, int end))

**12.17.3.25** **KDGantt::GraphicsView::Q\_PRIVATE\_SLOT** (d, void *slotColumnsInserted*(const QModelIndex &parent, int start, int end))

**12.17.3.26** **KDGantt::GraphicsView::Q\_PRIVATE\_SLOT** (d, void *slotHorizontalScrollValueChanged*(int))

**12.17.3.27** **KDGantt::GraphicsView::Q\_PRIVATE\_SLOT** (d, void *slotGridChanged*())

**12.17.3.28** **void GraphicsView::resizeEvent** (QResizeEvent \*) [protected]

Definition at line 514 of file `kdganttgraphicsview.cpp`.

References `d`, and `r`.

```

515 {
516     d->updateHeaderGeometry();
517     QRectF r = scene()->itemsBoundingRect();
518     // To scroll more to the left than the actual item start, bug #4516
519     r.setLeft( qMin( 0.0, r.left() ) );
520
521     QSizeF size = viewport()->size();
522     size.setHeight( size.height() + verticalScrollBar()->maximum() );
523     r.setSize( size.expandedTo( viewport()->size() ) );
524
525     scene()->setSceneRect( r );

```

```
526     BASE::resizeEvent( ev );
527 }
```

### 12.17.3.29 QModelIndex GraphicsView::rootIndex () const

**Returns:**

the rootindex for this view.

Definition at line 406 of file kdganttgraphicsview.cpp.

References [d](#).

Referenced by [updateScene\(\)](#).

```
407 {
408     return d->scene.rootIndex();
409 }
```

### 12.17.3.30 AbstractRowController \* GraphicsView::rowController () const

**Returns:**

the [AbstractRowController](#) for this view.

**See also:**

[setRowController](#)

Definition at line 456 of file kdganttgraphicsview.cpp.

References [d](#).

Referenced by [updateScene\(\)](#).

```
457 {
458     return d->rowcontroller;
459 }
```

### 12.17.3.31 QItemSelectionModel \* GraphicsView::selectionModel () const

**Returns:**

the [QItemSelectionModel](#) used by this view

Definition at line 421 of file kdganttgraphicsview.cpp.

References [d](#).

```
422 {
423     return d->scene.selectionModel();
424 }
```

**12.17.3.32 void GraphicsView::setConstraintModel (ConstraintModel \* *cmodel*)** [slot]

Sets the constraintmodel displayed by this view.

**See also:**

[KDGantt::ConstraintModel](#).

Definition at line 384 of file kdganttgraphicsview.cpp.

References [d](#).

```
385 {
386     d->scene.setConstraintModel( cmodel );
387 }
```

**12.17.3.33 void GraphicsView::setGrid (AbstractGrid \* *grid*)** [slot]

Sets the [AbstractGrid](#) for this view. The grid is an object that controls how QModelIndexes are mapped to and from the view and how the background and header is rendered.

**See also:**

[AbstractGrid](#) and [DateTimeGrid](#).

Definition at line 466 of file kdganttgraphicsview.cpp.

References [d](#), and [grid\(\)](#).

```
467 {
468     d->scene.setGrid( grid );
469     d->slotGridChanged();
470 }
```

**12.17.3.34 void GraphicsView::setItemDelegate (ItemDelegate \* *delegate*)** [slot]

Sets the [KDGantt::ItemDelegate](#) used for rendering items on this view.

**See also:**

[ItemDelegate](#) and [QAbstractItemDelegate](#).

Definition at line 429 of file kdganttgraphicsview.cpp.

References [d](#).

```
430 {
431     d->scene.setItemDelegate( delegate );
432 }
```

**12.17.3.35 void GraphicsView::setModel (QAbstractItemModel \* *model*)** [slot]

Sets the model to be displayed in this view to *model*. The view does not take ownership of the model.

To make a model work well with [GraphicsView](#) it must have a certain layout. Whether the model is flat or has a treestructure is not important, as long as an [AbstractRowController](#) is provided that can navigate the model.

[GraphicsView](#) operates per row in the model. The data is always taken from the `_last_` item in the row. The ItemRoles used are `Qt::DisplayRole` and the roles defined in [KDGantt::ItemDataRole](#).

Definition at line 340 of file `kdganttgraphicsview.cpp`.

References [d](#), and [updateScene\(\)](#).

```
341 {
342     d->scene.setModel( model );
343     updateScene();
344 }
```

**12.17.3.36 void GraphicsView::setReadOnly (bool *ro*)** [slot]

Sets the view to read-only mode if *ro* is true. The default is read/write if the model permits it.

Definition at line 482 of file `kdganttgraphicsview.cpp`.

References [d](#).

```
483 {
484     d->scene.setReadOnly( ro );
485 }
```

**12.17.3.37 void GraphicsView::setRootIndex (const QModelIndex & *idx*)** [slot]

Sets the root index of the model displayed by this view. Similar to `QAbstractItemView::setRootIndex`, default is `QModelIndex()`.

Definition at line 399 of file `kdganttgraphicsview.cpp`.

References [d](#).

```
400 {
401     d->scene.setRootIndex( idx );
402 }
```

**12.17.3.38 void GraphicsView::setRowController (AbstractRowController \* *rowcontroller*)**  
[slot]

Sets the [AbstractRowController](#) used by this view. The [AbstractRowController](#) deals with the height and position of each row and with which parts of the model are displayed.

**See also:**

[AbstractRowController](#)

Definition at line 446 of file kdganttgraphicsview.cpp.

References [d](#), and [updateScene\(\)](#).

```
447 {
448     d->rowcontroller = rowcontroller;
449     d->scene.setRowController( rowcontroller );
450     updateScene();
451 }
```

### 12.17.3.39 void GraphicsView::setSelectionModel (QItemSelectionModel \* model) [slot]

Sets the QItemSelectionModel used by this view to manage selections. Similar to QAbstractItemView::setSelectionModel

Definition at line 414 of file kdganttgraphicsview.cpp.

References [d](#).

```
415 {
416     d->scene.setSelectionModel( model );
417 }
```

### 12.17.3.40 void GraphicsView::setSummaryHandlingModel (QAbstractProxyModel \* model) [slot]

Definition at line 353 of file kdganttgraphicsview.cpp.

References [d](#), and [updateScene\(\)](#).

Referenced by [GraphicsView\(\)](#).

```
354 {
355     disconnect( d->scene.summaryHandlingModel() );
356     d->scene.setSummaryHandlingModel( proxyModel );
357
358     /* Connections. We have to rely on the treeview
359      * to receive the signals before we do(!)
360      */
361     connect( proxyModel, SIGNAL( columnsInserted( const QModelIndex&, int, int ) ),
362             this, SLOT( slotColumnsInserted( const QModelIndex&, int, int ) ) );
363     connect( proxyModel, SIGNAL( columnsRemoved( const QModelIndex&, int, int ) ),
364             this, SLOT( slotColumnsRemoved( const QModelIndex&, int, int ) ) );
365     connect( proxyModel, SIGNAL( dataChanged( const QModelIndex&, const QModelIndex& ) ),
366             this, SLOT( slotDataChanged( const QModelIndex&, const QModelIndex& ) ) );
367     connect( proxyModel, SIGNAL( layoutChanged() ),
368             this, SLOT( slotLayoutChanged() ) );
369     connect( proxyModel, SIGNAL( modelReset() ),
370             this, SLOT( slotModelReset() ) );
371     connect( proxyModel, SIGNAL( rowsInserted( const QModelIndex&, int, int ) ),
372             this, SLOT( slotRowsInserted( const QModelIndex&, int, int ) ) );
373     connect( proxyModel, SIGNAL( rowsAboutToBeRemoved( const QModelIndex&, int, int ) ),
374             this, SLOT( slotRowsAboutToBeRemoved( const QModelIndex&, int, int ) ) );
375     connect( proxyModel, SIGNAL( rowsRemoved( const QModelIndex&, int, int ) ),
376             this, SLOT( slotRowsRemoved( const QModelIndex&, int, int ) ) );
377
378     updateScene();
379 }
```

**12.17.3.41** `QAbstractProxyModel*` `KDGantt::GraphicsView::summaryHandlingModel () const`**12.17.3.42** `void GraphicsView::updateRow (const QModelIndex &)`

Definition at line 552 of file `kdganttgraphicsview.cpp`.

References `d`.

Referenced by `updateScene()`.

```
553 {
554     d->scene.updateRow( d->scene.summaryHandlingModel()->mapFromSource( idx ) );
555 }
```

**12.17.3.43** `void GraphicsView::updateScene ()`

Definition at line 576 of file `kdganttgraphicsview.cpp`.

References `clearItems()`, `model()`, `rootIndex()`, `rowController()`, `updateRow()`, and `updateSceneRect()`.

Referenced by `setModel()`, `setRowController()`, and `setSummaryHandlingModel()`.

```
577 {
578     clearItems();
579     if( !model() ) return;
580     if( !rowController() ) return;
581     QModelIndex idx = model()->index( 0, 0, rootIndex() );
582     do {
583         updateRow( idx );
584     } while ( ( idx = rowController()->indexBelow( idx ) ) != QModelIndex() && rowController()->isRowValid( idx ) );
585     //constraintModel()->cleanup();
586     //qDebug() << constraintModel();
587     updateSceneRect();
588 }
```

**12.17.3.44** `void GraphicsView::updateSceneRect ()`

Definition at line 560 of file `kdganttgraphicsview.cpp`.

References `d`, and `r`.

Referenced by `updateScene()`.

```
561 {
562     /* What to do with this? We need to shrink the view to
563      * make collapsing items work
564      */
565     QRectF r = d->scene.itemsBoundingRect();
566     // To scroll more to the left than the actual item start, bug #4516
567     r.setLeft( qMin( 0.0, r.left() ) );
568     r.setSize( r.size().expandedTo( viewport()->size() ) );
569
570     d->scene.setSceneRect( r );
571 }
```

## 12.17.4 Property Documentation

### 12.17.4.1 `bool KDGantt::GraphicsView::readOnly` [read, write]

Definition at line 48 of file `kdganttgraphicsview.h`.

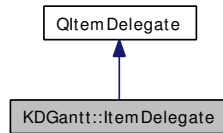
The documentation for this class was generated from the following files:

- [kdganttgraphicsview.h](#)
- [kdganttgraphicsview.cpp](#)

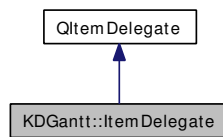
## 12.18 KDGantt::ItemDelegate Class Reference

```
#include <KDGanttItemDelegate>
```

Inheritance diagram for KDGantt::ItemDelegate:



Collaboration diagram for KDGantt::ItemDelegate:



### 12.18.1 Detailed Description

Class used to render gantt items in a [KDGantt::GraphicsView](#).

Definition at line 39 of file `kdganttitemdelegate.h`.

#### Public Types

- enum [InteractionState](#) {
  - [State\\_None](#) = 0,
  - [State\\_Move](#),
  - [State\\_ExtendLeft](#),
  - [State\\_ExtendRight](#),
  - [State\\_DragConstraint](#) }

#### Public Member Functions

- virtual [QRectF](#) [constraintBoundingRect](#) (const [QPointF](#) &start, const [QPointF](#) &end) const
- [QBrush](#) [defaultBrush](#) ([ItemType](#) type) const
- [QPen](#) [defaultPen](#) ([ItemType](#) type) const
- virtual [InteractionState](#) [interactionStateFor](#) (const [QPointF](#) &pos, const [StyleOptionGanttItem](#) &opt, const [QModelIndex](#) &idx) const
- virtual [Span](#) [itemBoundingSpan](#) (const [StyleOptionGanttItem](#) &opt, const [QModelIndex](#) &idx) const
- [ItemDelegate](#) ([QObject](#) \*parent=0)
- virtual void [paintConstraintItem](#) ([QPainter](#) \*p, const [QStyleOptionGraphicsItem](#) &opt, const [QPointF](#) &start, const [QPointF](#) &end, const [QPen](#) &pen)
- virtual void [paintGanttItem](#) ([QPainter](#) \*p, const [StyleOptionGanttItem](#) &opt, const [QModelIndex](#) &idx)

- void [setDefaultBrush](#) (ItemType type, const QBrush &brush)
- void [setDefaultPen](#) (ItemType type, const QPen &pen)
- virtual [~ItemDelegate](#) ()

## 12.18.2 Member Enumeration Documentation

### 12.18.2.1 enum [KDGantt::ItemDelegate::InteractionState](#)

This enum is used for communication between the view and the delegate about user interaction with gantt items.

See also:

[KDGantt::ItemDelegate::interactionStateFor](#)

Enumerator:

*State\_None*

*State\_Move*

*State\_ExtendLeft*

*State\_ExtendRight*

*State\_DragConstraint*

Definition at line 43 of file `kdganttitemdelegate.h`.

```

43             { State_None = 0,
44               State_Move,
45               State_ExtendLeft,
46               State_ExtendRight,
47               State_DragConstraint
48           };

```

## 12.18.3 Constructor & Destructor Documentation

### 12.18.3.1 [ItemDelegate::ItemDelegate](#) (QObject \*parent = 0) [explicit]

Constructor. Creates an [ItemDelegate](#) with parent *parent*

Definition at line 102 of file `kdganttitemdelegate.cpp`.

```

103     : QItemDelegate( parent ), _d( new Private )
104 {
105 }

```

### 12.18.3.2 [ItemDelegate::~ItemDelegate](#) () [virtual]

Destructor

Definition at line 108 of file `kdganttitemdelegate.cpp`.

```

109 {
110     delete _d;
111 }

```

## 12.18.4 Member Function Documentation

### 12.18.4.1 QRectF ItemDelegate::constraintBoundingRect (const QPointF & *start*, const QPointF & *end*) const [virtual]

#### Returns:

The bounding rectangle for the graphics used to represent a constraint between points *start* and *end* (typically an arrow)

Definition at line 350 of file `kdganttitemdelegate.cpp`.

Referenced by `KDGantt::ConstraintGraphicsItem::boundingRect()`.

```

351 {
352     QPolygonF poly;
353     if ( start.x() > end.x()-TURN ) {
354         if ( start.y() < end.y() ) {
355             poly << QPointF( start.x()+TURN, start.y()-TURN/2. )
356                 << QPointF( end.x()-TURN, end.y()+TURN/2. );
357         } else {
358             poly << QPointF( start.x()+TURN, start.y()+TURN/2. )
359                 << QPointF( end.x()-TURN, end.y()-TURN/2. );
360         }
361     } else {
362         if ( start.y() < end.y() ) {
363             poly << QPointF( start.x(), start.y()-TURN/2. )
364                 << QPointF( end.x(), end.y()+TURN/2. );
365         } else {
366             poly << QPointF( start.x(), start.y()+TURN/2. )
367                 << QPointF( end.x(), end.y()-TURN/2. );
368         }
369     }
370     return poly.boundingRect().adjusted( -PW, -PW, PW, PW );
371 }
```

### 12.18.4.2 QBrush ItemDelegate::defaultBrush (Item Type *type*) const

#### Returns:

The default brush for item type *type*

#### Todo

Move this to [GraphicsView](#) to make delegate stateless.

Definition at line 130 of file `kdganttitemdelegate.cpp`.

References `d`.

Referenced by `paintGanttItem()`.

```

131 {
132     return d->defaultbrush[type];
133 }
```

### 12.18.4.3 QPen ItemDelegate::defaultPen (Item Type *type*) const

#### Returns:

The default pen for item type *type*

**Todo**

Move this to [GraphicsView](#) to make delegate stateless.

Definition at line 150 of file `kdganttitemdelegate.cpp`.

References `d`.

Referenced by `paintGanttItem()`.

```
151 {
152     return d->defaultpen[type];
153 }
```

#### 12.18.4.4 **ItemDelegate::InteractionState** `ItemDelegate::interactionStateFor (const QPointF & pos, const StyleOptionGanttItem & opt, const QModelIndex & idx) const` [virtual]

**Returns:**

The interaction state for position *pos* on item *idx* when rendered with options *opt*. This is used to tell the view about how the item should react to mouse click/drag.

Override to implement new items or interactions.

Definition at line 198 of file `kdganttitemdelegate.cpp`.

References `KDGantt::StyleOptionGanttItem::itemRect`, `KDGantt::ItemTypeRole`, `State_ExtendLeft`, `State_ExtendRight`, `State_Move`, `State_None`, `KDGantt::TypeEvent`, `KDGantt::TypeNone`, and `KDGantt::TypeSummary`.

Referenced by `KDGantt::GraphicsItem::hoverMoveEvent()`, `KDGantt::GraphicsItem::mouseDoubleClickEvent()`, and `KDGantt::GraphicsItem::mousePressEvent()`.

```
201 {
202     if ( !idx.isValid() ) return State_None;
203     if ( !( idx.model()->flags( idx ) & Qt::ItemIsEditable ) ) return State_None;
204
205     int typ = static_cast<ItemType>( idx.model()->data( idx, ItemTypeRole ).toInt() );
206     if ( typ == TypeNone || typ == TypeSummary ) return State_None;
207     if ( typ == TypeEvent ) return State_Move;
208     if ( !opt.itemRect.contains(pos) ) return State_None;
209
210     qreal delta = 5.;
211     if ( opt.itemRect.width() < 15 ) delta = 1.;
212     if( pos.x() >= opt.itemRect.left() && pos.x() < opt.itemRect.left()+delta ) {
213         return State_ExtendLeft;
214     } else if( pos.x() <= opt.itemRect.right() && pos.x() > opt.itemRect.right()-delta ) {
215         return State_ExtendRight;
216     } else {
217         return State_Move;
218     }
219 }
```

#### 12.18.4.5 **Span** `ItemDelegate::itemBoundingSpan (const StyleOptionGanttItem & opt, const QModelIndex & idx) const` [virtual]

**Returns:**

The bounding [Span](#) for the item identified by *idx* when rendered with options *opt*. This is often the same as the span given by the [AbstractGrid](#) for *idx*, but it might be larger in case there are additional texts or decorations on the item.

Override this to implement new itemtypes or to change the look of the existing ones.

Definition at line 163 of file kdganttitemdelegate.cpp.

References KDGantt::StyleOptionGanttItem::Center, KDGantt::StyleOptionGanttItem::displayPosition, KDGantt::StyleOptionGanttItem::itemRect, KDGantt::ItemTypeRole, KDGantt::StyleOptionGanttItem::Left, KDGantt::StyleOptionGanttItem::Right, and KDGantt::TypeEvent.

Referenced by KDGantt::GraphicsItem::updateItem().

```

165 {
166     //qDebug() << "itemBoundingSpan("<<opt<<idx<<"");
167     if ( !idx.isValid() ) return Span();
168
169     QString txt = idx.model()->data( idx, Qt::DisplayRole ).toString();
170     int typ = idx.model()->data( idx, ItemTypeRole ).toInt();
171     QRectF itemRect = opt.itemRect;
172
173
174     if ( typ == TypeEvent ) itemRect = QRectF( itemRect.left()-itemRect.height()/2.,
175                                               itemRect.top(),
176                                               itemRect.height(),
177                                               itemRect.height() );
178
179     int tw = opt.fontMetrics.width( txt );
180     tw += static_cast<int>( itemRect.height()/2. );
181     switch ( opt.displayPosition ) {
182     case StyleOptionGanttItem::Left:
183         return Span( itemRect.left()-tw, itemRect.width()+tw );
184     case StyleOptionGanttItem::Right:
185         return Span( itemRect.left(), itemRect.width()+tw );
186     case StyleOptionGanttItem::Center:
187         return Span( itemRect.left(), itemRect.width() );
188     }
189     return Span();
190 }

```

#### 12.18.4.6 void ItemDelegate::paintConstraintItem (QPainter \* painter, const QStyleOptionGraphicsItem & opt, const QPointF & start, const QPointF & end, const QPen & pen) [virtual]

Paints the constraint item between points *start* and *end* using *painter* and *opt*.

#### Todo

Review *opt*'s type

Definition at line 379 of file kdganttitemdelegate.cpp.

Referenced by KDGantt::ConstraintGraphicsItem::paint().

```

381 {
382     Q_UNUSED( opt );
383     qreal midx = end.x() - TURN;
384     qreal midy = ( end.y()-start.y() )/2. + start.y();
385
386     painter->setPen( pen );
387     painter->setBrush( pen.color() );
388
389     if ( start.x() > end.x()-TURN ) {
390         QPolygonF poly;
391         poly << start

```

```

392         << QPointF( start.x()+TURN, start.y() )
393         << QPointF( start.x()+TURN, midy )
394         << QPointF( end.x()-TURN, midy )
395         << QPointF( end.x()-TURN, end.y() )
396         << end;
397     painter->drawPolyline( poly );
398     QPolygonF arrow;
399     arrow << end
400         << QPointF( end.x()-TURN/2., end.y()-TURN/2. )
401         << QPointF( end.x()-TURN/2., end.y()+TURN/2. );
402     painter->drawPolygon( arrow );
403 } else {
404     QPolygonF poly;
405     poly << start
406         << QPointF( midx, start.y() )
407         << QPointF( midx, end.y() )
408         << end;
409     painter->drawPolyline( poly );
410     QPolygonF arrow;
411     arrow << end
412         << QPointF( end.x()-TURN/2., end.y()-TURN/2. )
413         << QPointF( end.x()-TURN/2., end.y()+TURN/2. );
414     painter->drawPolygon( arrow );
415 }
416 }

```

#### 12.18.4.7 void ItemDelegate::paintGanttItem (QPainter \*painter, const StyleOptionGanttItem &opt, const QModelIndex &idx) [virtual]

Paints the gantt item *idx* using *painter* and *opt*

Definition at line 229 of file kdganttitemdelegate.cpp.

References KDGantt::StyleOptionGanttItem::boundingRect, KDGantt::StyleOptionGanttItem::Center, defaultBrush(), defaultPen(), KDGantt::StyleOptionGanttItem::displayPosition, KDGantt::StyleOptionGanttItem::itemRect, KDGantt::ItemTypeRole, KDGantt::StyleOptionGanttItem::Left, r, KDGantt::StyleOptionGanttItem::Right, KDGantt::TaskCompletionRole, KDGantt::StyleOptionGanttItem::text, KDGantt::TypeEvent, KDGantt::TypeSummary, and KDGantt::TypeTask.

Referenced by KDGantt::GraphicsItem::paint().

```

232 {
233     if ( !idx.isValid() ) return;
234     const ItemType typ = static_cast<ItemType>( idx.model()->data( idx, ItemTypeRole ).toInt() );
235     const QString& txt = opt.text;
236     QRectF itemRect = opt.itemRect;
237     QRectF boundingRect = opt.boundingRect;
238     boundingRect.setY( itemRect.y() );
239     boundingRect.setHeight( itemRect.height() );
240     //qDebug() << "itemRect="<<itemRect<<", boundingRect="<<boundingRect;
241
242     painter->save();
243
244     QPen pen = defaultPen( typ );
245     if ( opt.state & QStyle::State_Selected ) pen.setWidth( 2*pen.width() );
246     painter->setPen( pen );
247     painter->setBrush( defaultBrush( typ ) );
248
249     qreal pw = painter->pen().width()/2.;
250     switch( typ ) {
251     case TypeTask:
252         if ( itemRect.isValid() ) {
253             // TODO

```

```

254         qreal pw = painter->pen().width()/2.;
255         pw-=1;
256         QRectF r = itemRect;
257         r.translate( 0., r.height()/6. );
258         r.setHeight( 2.*r.height()/3. );
259         painter->setBrushOrigin( itemRect.topLeft() );
260         painter->save();
261         painter->translate( 0.5, 0.5 );
262         painter->drawRect( r );
263         bool ok;
264         qreal completion = idx.model()->data( idx, KDGantt::TaskCompletionRole ).toDouble( &ok );
265         if ( ok ) {
266             qreal h = r.height();
267             QRectF cr( r.x(), r.y()+h/4.,
268                     r.width()*completion/100., h/2.+1 /*??*/ );
269             painter->fillRect( cr, painter->pen().brush() );
270         }
271         painter->restore();
272         Qt::Alignment ta;
273         switch( opt.displayPosition ) {
274             case StyleOptionGanttItem::Left: ta = Qt::AlignLeft; break;
275             case StyleOptionGanttItem::Right: ta = Qt::AlignRight; break;
276             case StyleOptionGanttItem::Center: ta = Qt::AlignCenter; break;
277         }
278         painter->drawText( boundingRect, ta, txt );
279     }
280     break;
281 case TypeSummary:
282     if ( opt.itemRect.isValid() ) {
283         // TODO
284         pw-=1;
285         const QRectF r = QRectF( opt.itemRect ).adjusted( -pw, -pw, pw, pw );
286         QPainterPath path;
287         const qreal delta = r.height()/2.;
288         path.moveTo( r.topLeft() );
289         path.lineTo( r.topRight() );
290         path.lineTo( QPointF( r.right(), r.top() + 2.*delta ) );
291         //path.lineTo( QPointF( r.right()-3./2.*delta, r.top() + delta ) );
292         path.quadTo( QPointF( r.right()-0.5*delta, r.top() + delta ), QPointF( r.right()-2.*delta,
293         //path.lineTo( QPointF( r.left()+3./2.*delta, r.top() + delta ) );
294         path.lineTo( QPointF( r.left() + 2.*delta, r.top() + delta ) );
295         path.quadTo( QPointF( r.left()+0.5*delta, r.top() + delta ), QPointF( r.left(), r.top() + 2
296         path.closeSubpath();
297         painter->setBrushOrigin( itemRect.topLeft() );
298         painter->save();
299         painter->translate( 0.5, 0.5 );
300         painter->drawPath( path );
301         painter->restore();
302         Qt::Alignment ta;
303         switch( opt.displayPosition ) {
304             case StyleOptionGanttItem::Left: ta = Qt::AlignLeft; break;
305             case StyleOptionGanttItem::Right: ta = Qt::AlignRight; break;
306             case StyleOptionGanttItem::Center: ta = Qt::AlignCenter; break;
307         }
308         painter->drawText( boundingRect, ta | Qt::AlignVCenter, txt );
309     }
310     break;
311 case TypeEvent: /* TODO */
312     //qDebug() << opt.boundingRect << opt.itemRect;
313     if ( opt.boundingRect.isValid() ) {
314         const qreal pw = painter->pen().width() / 2. - 1;
315         const QRectF r = QRectF( opt.rect ).adjusted( -pw, -pw, pw, pw );
316         QPainterPath path;
317         const qreal delta = static_cast< int >( r.height() / 2 );
318         path.moveTo( delta, 0. );
319         path.lineTo( 2.*delta, delta );
320         path.lineTo( delta, 2.*delta );

```

```

321         path.lineTo( 0., delta );
322         path.closeSubpath();
323         painter->save();
324         painter->translate( r.topLeft() );
325         painter->translate( 0.5, 0.5 );
326         painter->drawPath( path );
327         painter->restore();
328         Qt::Alignment ta;
329         switch( opt.displayPosition ) {
330             case StyleOptionGanttItem::Left: ta = Qt::AlignLeft; break;
331             case StyleOptionGanttItem::Right: ta = Qt::AlignRight; break;
332             case StyleOptionGanttItem::Center: ta = Qt::AlignCenter; break;
333         }
334         painter->drawText( boundingRect, ta | Qt::AlignVCenter, txt );
335     }
336     break;
337     default:
338         break;
339 }
340 painter->restore();
341 }

```

#### 12.18.4.8 void ItemDelegate::setDefaultBrush (ItemType type, const QBrush & brush)

Sets the default brush used for items of type *type* to *brush*. The default brush is used in the case when the model does not provide an explicit brush.

#### Todo

Move this to [GraphicsView](#) to make delegate stateless.

Definition at line 121 of file `kdganttitemdelegate.cpp`.

References d.

```

122 {
123     d->defaultbrush[type] = brush;
124 }

```

#### 12.18.4.9 void ItemDelegate::setDefaultPen (ItemType type, const QPen & pen)

Sets the default pen used for items of type *type* to *pen*. The default pen is used in the case when the model does not provide an explicit pen.

#### Todo

Move this to [GraphicsView](#) to make delegate stateless.

Definition at line 141 of file `kdganttitemdelegate.cpp`.

References d.

```

142 {
143     d->defaultpen[type]=pen;
144 }

```

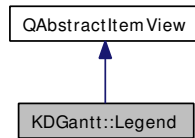
The documentation for this class was generated from the following files:

- [kdganttitemdelegate.h](#)
- [kdganttitemdelegate.cpp](#)

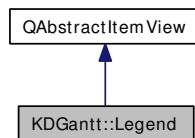
## 12.19 KDGantt::Legend Class Reference

```
#include <KDGanttLegend>
```

Inheritance diagram for KDGantt::Legend:



Collaboration diagram for KDGantt::Legend:



### 12.19.1 Detailed Description

[Legend](#) showing an image and a description for Gantt items.

This is an item view class showing a small Gantt item and its text defined by LegendRole.

Definition at line 35 of file `kdganttlegend.h`.

#### Public Member Functions

- QModelIndex [indexAt](#) (const QPoint &point) const
- [Legend](#) (QWidget \*parent=0)
- QSize [minimumSizeHint](#) () const
- void [scrollTo](#) (const QModelIndex &, ScrollHint=EnsureVisible)
- void [setModel](#) (QAbstractItemModel \*model)
- QSize [sizeHint](#) () const
- QRect [visualRect](#) (const QModelIndex &index) const
- virtual [~Legend](#) ()

#### Protected Slots

- virtual void [modelDataChanged](#) ()

#### Protected Member Functions

- virtual QRect [drawItem](#) (QPainter \*painter, const QModelIndex &index, const QPoint &pos=QPoint()) const
- virtual [StyleOptionGanttItem](#) [getStyleOption](#) (const QModelIndex &index) const
- int [horizontalOffset](#) () const

- bool `isIndexHidden` (const QModelIndex &) const
- virtual QSize `measureItem` (const QModelIndex &index, bool recursive=true) const
- QModelIndex `moveCursor` (CursorAction, Qt::KeyboardModifiers)
- void `paintEvent` (QPaintEvent \*event)
- void `setSelection` (const QRect &, QItemSelectionModel::SelectionFlags)
- int `verticalOffset` () const
- QRegion `visualRegionForSelection` (const QItemSelection &) const

## 12.19.2 Constructor & Destructor Documentation

### 12.19.2.1 Legend::Legend (QWidget \*parent = 0) [explicit]

Constructor. Creates a [Legend](#) with parent *parent*. The [QObject](#) parent is not used for anything internally.

Definition at line 47 of file `kdganttlegend.cpp`.

```

48     : QAbstractItemView( parent ),
49       _d( new Private )
50 {
51     setItemDelegate( new ItemDelegate( this ) );
52     setFrameStyle( QFrame::NoFrame );
53 }
```

### 12.19.2.2 Legend::~~Legend () [virtual]

Destructor. Does nothing

Definition at line 56 of file `kdganttlegend.cpp`.

```

57 {
58     delete _d;
59 }
```

## 12.19.3 Member Function Documentation

### 12.19.3.1 QRect Legend::drawItem (QPainter \*painter, const QModelIndex &index, const QPoint &pos = QPoint()) const [protected, virtual]

Draws the legend item at *index* and all of it's children recursively at *pos* onto *painter*. Reimplement this if you want to draw items in an user defined way.

#### Returns:

the rectangle drawn.

Definition at line 142 of file `kdganttlegend.cpp`.

References `KDGantt::StyleOptionGanttItem::boundingRect`, `d`, `getStyleOption()`, `KDGantt::StyleOptionGanttItem::itemRect`, `measureItem()`, `r`, and `KDGantt::StyleOptionGanttItem::text`.

Referenced by `paintEvent()`.

```

143 {
144     int xPos = pos.x();
145     int yPos = pos.y();
146
147     if( index.isValid() && index.model() == &d->proxyModel )
148     {
149         ItemDelegate* const delegate = qobject_cast< ItemDelegate* >( itemDelegate( index ) );
150         assert( delegate != 0 );
151         const QRect r( pos, measureItem( index, false ) );
152         StyleOptionGanttItem opt = getStyleOption( index );
153         opt.rect = r;
154         opt.rect.setWidth( r.height() );
155         opt.itemRect = opt.rect;
156         opt.boundingRect = r;
157         opt.boundingRect.setWidth( r.width() + r.height() );
158         if( !opt.text.isNull() )
159             delegate->paintGanttItem( painter, opt, index );
160
161         xPos = r.right();
162         yPos = r.bottom();
163     }
164
165
166     const int rowCount = d->proxyModel.rowCount( index );
167     for( int row = 0; row < rowCount; ++row )
168     {
169         const QRect r = drawItem( painter, d->proxyModel.index( row, 0, index ), QPoint( pos.x(), yPos ) );
170         xPos = qMax( xPos, r.right() );
171         yPos = qMax( yPos, r.bottom() );
172     }
173
174     return QRect( pos, QPoint( xPos, yPos ) );
175 }

```

### 12.19.3.2 `StyleOptionGanttItem Legend::getStyleOption (const QModelIndex & index) const` [protected, virtual]

Creates a `StyleOptionGanttItem` with all style options filled in except the target rectangles.

Definition at line 127 of file `kdganttlegend.cpp`.

References `d`, `KDGantt::StyleOptionGanttItem::displayPosition`, `KDGantt::LegendRole`, `KDGantt::StyleOptionGanttItem::Right`, and `KDGantt::StyleOptionGanttItem::text`.

Referenced by `drawItem()`.

```

128 {
129     StyleOptionGanttItem opt;
130     opt.displayPosition = StyleOptionGanttItem::Right;
131     opt.displayAlignment = Qt::Alignment( d->proxyModel.data( index, Qt::TextAlignmentRole ).toInt() );
132     opt.text = index.model()->data( index, LegendRole ).toString();
133     opt.font = qVariantValue< QFont >( index.model()->data( index, Qt::FontRole ) );
134     return opt;
135 }

```

### 12.19.3.3 `int KDGantt::Legend::horizontalOffset () const` [protected]

Definition at line 60 of file `kdganttlegend.h`.

```
60 { return 0; }
```

### 12.19.3.4 QModelIndex Legend::indexAt (const QPoint & point) const

Definition at line 63 of file kdganttlegend.cpp.

```
64 {
65     return QModelIndex();
66 }
```

### 12.19.3.5 bool KDGantt::Legend::isIndexHidden (const QModelIndex &) const [protected]

Definition at line 61 of file kdganttlegend.h.

```
61 { return false; }
```

### 12.19.3.6 QSize Legend::measureItem (const QModelIndex & index, bool recursive = true) const [protected, virtual]

Calculates the needed space for the legend item at *index* and, if *recursive* is true, all child items.

Definition at line 180 of file kdganttlegend.cpp.

References [d](#), and [KDGantt::LegendRole](#).

Referenced by [drawItem\(\)](#), [minimumSizeHint\(\)](#), and [sizeHint\(\)](#).

```
181 {
182     if( model() == 0 )
183         return QSize();
184
185     QSize baseSize;
186     if( index.model() != 0 )
187     {
188         QFontMetrics fm( qVariantValue< QFont >( index.model()->data( index, Qt::FontRole ) ) );
189         const QString text = index.model()->data( index, LegendRole ).toString();
190         if( !text.isNull() )
191             baseSize += QSize( fm.width( text ) + fm.height() + 2, fm.height() + 2 );
192     }
193
194     if( !recursive )
195         return baseSize;
196
197     QSize childrenSize;
198
199     const int rowCount = d->proxyModel.rowCount( index );
200     for( int row = 0; row < rowCount; ++row )
201     {
202         const QSize childSize = measureItem( d->proxyModel.index( row, 0, index ) );
203         childrenSize.setWidth( qMax( childrenSize.width(), childSize.width() ) );
204         childrenSize.rheight() += childSize.height();
205     }
206     return baseSize + childrenSize;
207 }
```

### 12.19.3.7 QSize Legend::minimumSizeHint () const

Definition at line 78 of file kdganttlegend.cpp.

References [measureItem\(\)](#).

```
79 {
80     return measureItem( rootIndex() );
81 }
```

### 12.19.3.8 void Legend::modelDataChanged () [protected, virtual, slot]

Triggers repainting of the legend.

Definition at line 106 of file kdganttlegend.cpp.

Referenced by setModel().

```
107 {
108     updateGeometry();
109     viewport()->update();
110 }
```

### 12.19.3.9 QModelIndex KDGantt::Legend::moveCursor (CursorAction, Qt::KeyboardModifiers) [protected]

Definition at line 62 of file kdganttlegend.h.

```
62 { return QModelIndex(); }
```

### 12.19.3.10 void Legend::paintEvent (QPaintEvent \* event) [protected]

Definition at line 112 of file kdganttlegend.cpp.

References drawItem(), and p.

```
113 {
114     Q_UNUSED( event );
115     // no model, no legend...
116     if( model() == 0 )
117         return;
118
119     QPainter p( viewport() );
120     p.fillRect( viewport()->rect(), palette().color( QPalette::Window ) );
121     drawItem( &p, rootIndex() );
122 }
```

### 12.19.3.11 void KDGantt::Legend::scrollTo (const QModelIndex &, ScrollHint = EnsureVisible)

Definition at line 46 of file kdganttlegend.h.

```
46 {}
```

**12.19.3.12 void Legend::setModel (QAbstractItemModel \* model)**

Definition at line 83 of file kdganttlegend.cpp.

References d, and modelDataChanged().

```

84 {
85     if( this->model() != 0 )
86     {
87         disconnect( this->model(), SIGNAL( dataChanged( QModelIndex, QModelIndex ) ), this, SLOT( modelDat
88         disconnect( this->model(), SIGNAL( rowsRemoved( QModelIndex, int, int ) ), this, SLOT( modelDataCh
89         disconnect( this->model(), SIGNAL( columnsRemoved( QModelIndex, int, int ) ), this, SLOT( modelDat
90     }
91
92     QAbstractItemView::setModel( model );
93     d->proxyModel.setSourceModel( model );
94
95     if( this->model() != 0 )
96     {
97         connect( this->model(), SIGNAL( dataChanged( QModelIndex, QModelIndex ) ), this, SLOT( modelDat
98         connect( this->model(), SIGNAL( rowsRemoved( QModelIndex, int, int ) ), this, SLOT( modelDataCh
99         connect( this->model(), SIGNAL( columnsRemoved( QModelIndex, int, int ) ), this, SLOT( modelDat
100    }
101
102 }
```

**12.19.3.13 void KDGantt::Legend::setSelection (const QRect &, QItemSelectionModel::Selection-Flags) [protected]**

Definition at line 63 of file kdganttlegend.h.

```
63 {}
```

**12.19.3.14 QSize Legend::sizeHint () const**

Definition at line 73 of file kdganttlegend.cpp.

References measureItem().

```

74 {
75     return measureItem( rootIndex() );
76 }
```

**12.19.3.15 int KDGantt::Legend::verticalOffset () const [protected]**

Definition at line 64 of file kdganttlegend.h.

```
64 { return 0; }
```

**12.19.3.16 QRect Legend::visualRect (const QModelIndex & index) const**

Definition at line 68 of file kdganttlegend.cpp.

```

69 {
70     return QRect();
71 }
```

**12.19.3.17 QRegion KDGantt::Legend::visualRegionForSelection (const QItemSelection &) const**  
[protected]

Definition at line 65 of file kdganttlegend.h.

```
65 { return QRegion(); }
```

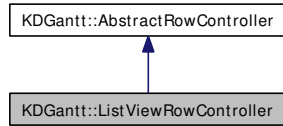
The documentation for this class was generated from the following files:

- [kdganttlegend.h](#)
- [kdganttlegend.cpp](#)

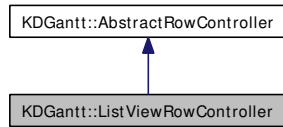
## 12.20 KDGantt::ListViewRowController Class Reference

```
#include <kdganttlistviewrowcontroller.h>
```

Inheritance diagram for KDGantt::ListViewRowController:



Collaboration diagram for KDGantt::ListViewRowController:



### 12.20.1 Detailed Description

Definition at line 34 of file kdganttlistviewrowcontroller.h.

### Public Member Functions

- int [headerHeight](#) () const
- QModelIndex [indexAbove](#) (const QModelIndex &idx) const
- QModelIndex [indexAt](#) (int height) const
- QModelIndex [indexBelow](#) (const QModelIndex &idx) const
- bool [isRowVisible](#) (const QModelIndex &idx) const
- [ListViewRowController](#) (QListView \*lv, [QAbstractProxyModel](#) \*proxy)
- int [maximumItemHeight](#) () const
- [Span rowGeometry](#) (const QModelIndex &idx) const
- [~ListViewRowController](#) ()

### 12.20.2 Constructor & Destructor Documentation

#### 12.20.2.1 [ListViewRowController::ListViewRowController](#) (QListView \* lv, [QAbstractProxyModel](#) \* proxy)

Definition at line 42 of file kdganttlistviewrowcontroller.cpp.

```

43   : _d( new Private(lv,proxy) )
44   {
45   }
```

### 12.20.2.2 ListViewRowController::~~ListViewRowController ()

Definition at line 47 of file kdganttlistviewrowcontroller.cpp.

```
48 {
49     delete _d; _d = 0;
50 }
```

## 12.20.3 Member Function Documentation

### 12.20.3.1 int ListViewRowController::headerHeight () const [virtual]

#### Returns:

The height of the header part of the view.

Implement this to control how much space is reserved at the top of the view for a header

Implements [KDGantt::AbstractRowController](#).

Definition at line 54 of file kdganttlistviewrowcontroller.cpp.

References [d](#).

```
55 {
56     return d->listview->viewport()->y()-d->listview->frameWidth();
57 }
```

### 12.20.3.2 QModelIndex ListViewRowController::indexAbove (const QModelIndex & *idx*) const [virtual]

#### Returns:

The modelindex for the previous row before *idx*.

#### See also:

[QTreeView::indexAbove](#)

Implements [KDGantt::AbstractRowController](#).

Definition at line 85 of file kdganttlistviewrowcontroller.cpp.

References [d](#).

```
86 {
87     const QModelIndex idx = d->proxy->mapToSource( _idx );
88     return d->proxy->mapFromSource( idx.sibling( idx.row()-1, idx.column() ) );
89 }
```

### 12.20.3.3 QModelIndex ListViewRowController::indexAt (int *height*) const [virtual]

Implements [KDGantt::AbstractRowController](#).

Definition at line 80 of file kdganttlistviewrowcontroller.cpp.

References [d](#).

```

81 {
82     return d->proxy->mapFromSource( d->listview->indexAt( QPoint( 1,height ) ) );
83 }

```

#### 12.20.3.4 `QModelIndex ListViewRowController::indexBelow (const QModelIndex & idx) const` [virtual]

##### Returns:

The modelindex for the next row after *idx*.

##### See also:

`QTreeView::indexBelow`

Implements [KDGantt::AbstractRowController](#).

Definition at line 91 of file `kdganttlistviewrowcontroller.cpp`.

References `d`.

```

92 {
93     const QModelIndex idx = d->proxy->mapToSource( _idx );
94     if( !idx.isValid() || idx.column() != 0 ) return QModelIndex();
95     if( idx.model()->rowCount( idx.parent() ) < idx.row()+1 ) return QModelIndex();
96     return d->proxy->mapFromSource( idx.sibling( idx.row()+1, idx.column() ) );
97 }

```

#### 12.20.3.5 `bool ListViewRowController::isRowVisible (const QModelIndex & idx) const` [virtual]

##### Returns:

true if the row containing index *idx* is visible in the view.

Implement this to allow [KDGantt](#) to optimise how items on screen are created. It is not harmful to always return true here, but the [View](#) will not perform optimally.

Implements [KDGantt::AbstractRowController](#).

Definition at line 64 of file `kdganttlistviewrowcontroller.cpp`.

References `d`.

```

65 {
66     const QModelIndex idx = d->proxy->mapToSource( _idx );
67     assert( idx.isValid() ? ( idx.model() == d->listview->model() ) : ( true ) );
68     return d->listview->visualRect( idx ).isValid();
69 }

```

#### 12.20.3.6 `int ListViewRowController::maximumItemHeight () const` [virtual]

Implements [KDGantt::AbstractRowController](#).

Definition at line 59 of file `kdganttlistviewrowcontroller.cpp`.

References `d`.

```
60 {
61     return d->listview->fontMetrics().height();
62 }
```

### 12.20.3.7 [Span](#) ListViewRowController::rowGeometry (const QModelIndex & *idx*) const [virtual]

#### Returns:

A [Span](#) consisting of the row offset and height for the row containing *idx*. A simple implementation might look like

```
Span MyRowCtrlr::rowGeometry(const QModelIndex& idx)
{
    return Span(idx.row()*10,10);
}
```

Implements [KDGantt::AbstractRowController](#).

Definition at line 71 of file `kdganttlistviewrowcontroller.cpp`.

References `d`, and `r`.

```
72 {
73     const QModelIndex idx = d->proxy->mapToSource( _idx );
74     assert( idx.isValid() ? ( idx.model() == d->listview->model() ) : ( true ) );
75     QRect r = d->listview->visualRect( idx ).translated( QPoint( 0,
76         static_cast<Private::HackListView*>(d->listview)->verticalOffset() ) );
77     return Span( r.y(), r.height() );
78 }
```

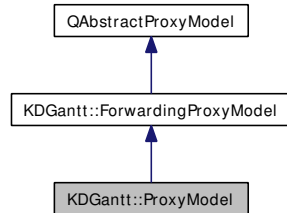
The documentation for this class was generated from the following files:

- [kdganttlistviewrowcontroller.h](#)
- [kdganttlistviewrowcontroller.cpp](#)

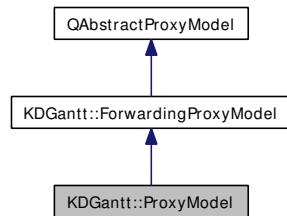
## 12.21 KDGantt::ProxyModel Class Reference

```
#include <kdganttproxymodel.h>
```

Inheritance diagram for KDGantt::ProxyModel:



Collaboration diagram for KDGantt::ProxyModel:



### 12.21.1 Detailed Description

Definition at line 31 of file kdganttproxymodel.h.

#### Public Member Functions

- int [column](#) (int ganttrole) const
- int [columnCount](#) (const QModelIndex &idx) const
- QVariant [data](#) (const QModelIndex &idx, int role=Qt::DisplayRole) const
- QModelIndex [index](#) (int row, int column, const QModelIndex &parent=QModelIndex()) const
- QModelIndex [mapFromSource](#) (const QModelIndex &idx) const
- QModelIndex [mapToSource](#) (const QModelIndex &proxyIdx) const
- QModelIndex [parent](#) (const QModelIndex &idx) const
- [ProxyModel](#) (QObject \*parent=0)
- int [role](#) (int ganttrole) const
- int [rowCount](#) (const QModelIndex &idx) const
- void [setColumn](#) (int ganttrole, int col)
- bool [setData](#) (const QModelIndex &idx, const QVariant &value, int role=Qt::EditRole)
- void [setRole](#) (int ganttrole, int role)
- void [setSourceModel](#) (QAbstractItemModel \*model)
- virtual [~ProxyModel](#) ()

## Protected Slots

- virtual void [sourceColumnsAboutToBeInserted](#) (const QModelIndex &idx, int start, int end)
- virtual void [sourceColumnsAboutToBeRemoved](#) (const QModelIndex &idx, int start, int end)
- virtual void [sourceColumnsInserted](#) (const QModelIndex &idx, int start, int end)
- virtual void [sourceColumnsRemoved](#) (const QModelIndex &idx, int start, int end)
- virtual void [sourceDataChanged](#) (const QModelIndex &from, const QModelIndex &to)
- virtual void [sourceLayoutAboutToBeChanged](#) ()
- virtual void [sourceLayoutChanged](#) ()
- virtual void [sourceModelAboutToBeReset](#) ()
- virtual void [sourceModelReset](#) ()
- virtual void [sourceRowsAboutToBeInserted](#) (const QModelIndex &idx, int start, int end)
- virtual void [sourceRowsAboutToBeRemoved](#) (const QModelIndex &, int start, int end)
- virtual void [sourceRowsInserted](#) (const QModelIndex &idx, int start, int end)
- virtual void [sourceRowsRemoved](#) (const QModelIndex &, int start, int end)

## 12.21.2 Constructor & Destructor Documentation

### 12.21.2.1 ProxyModel::ProxyModel (QObject \*parent = 0) [explicit]

Definition at line 55 of file kdganttproxymodel.cpp.

```
56     : BASE( parent ), _d( new Private( this ) )
57 {
58     init();
59 }
```

### 12.21.2.2 ProxyModel::~~ProxyModel () [virtual]

Definition at line 61 of file kdganttproxymodel.cpp.

```
62 {
63     delete _d; _d = 0;
64 }
```

## 12.21.3 Member Function Documentation

### 12.21.3.1 int ProxyModel::column (int ganttrole) const

Definition at line 110 of file kdganttproxymodel.cpp.

References [d](#).

```
111 {
112     return d->columnMap[ganttrole];
113 }
```

### 12.21.3.2 `int ProxyModel::columnCount (const QModelIndex & idx) const`

See also:

`QAbstractItemModel::columnCount`

Reimplemented from `KDGantt::ForwardingProxyModel`.

Definition at line 146 of file `kdganttproxymodel.cpp`.

References `mapToSource()`.

```
147 {
148     return qMin( sourceModel()->columnCount( mapToSource( proxyIndex ) ), 1 );
149 }
```

### 12.21.3.3 `QVariant ProxyModel::data (const QModelIndex & idx, int role = Qt::DisplayRole) const`

Definition at line 151 of file `kdganttproxymodel.cpp`.

References `d`, and `mapToSource()`.

```
152 {
153     int srole = role;
154     int scol = proxyIdx.column();
155     QHash<int, int>::const_iterator it = d->roleMap.find( role );
156     if ( it != d->roleMap.end() ) srole = *it;
157     it = d->columnMap.find( role );
158     if ( it != d->columnMap.end() ) scol = *it;
159
160     //qDebug() << "mapping role"<<static_cast<ItemDataRole>(role)<<"to"<<static_cast<ItemDataRole>(srole);
161     //qDebug() << "mapping column"<<proxyIdx.column()<<"to"<<scol;
162
163     const QAbstractItemModel* model = sourceModel();
164     return model->data( model->index( proxyIdx.row(), scol, mapToSource( proxyIdx.parent() ) ), srole );
165 }
```

### 12.21.3.4 `QModelIndex ForwardingProxyModel::index (int row, int column, const QModelIndex & parent = QModelIndex()) const` [inherited]

See also:

`QAbstractItemModel::index`

Definition at line 261 of file `kdganttforwardingproxymodel.cpp`.

References `KDGantt::ForwardingProxyModel::mapFromSource()`, and `KDGantt::ForwardingProxyModel::mapToSource()`.

Referenced by `KDAB_SCOPED_UNITTEST_SIMPLE()`.

```
262 {
263     return mapFromSource( sourceModel()->index( row, column, mapToSource( parent ) ) );
264 }
```

**12.21.3.5 QModelIndex ProxyModel::mapFromSource (const QModelIndex & idx) const**

Converts indexes in the source model to indexes in the proxy model

Reimplemented from [KDGantt::ForwardingProxyModel](#).

Definition at line 72 of file `kdganttproxymodel.cpp`.

```

73 {
74     if( sourceIdx.isValid() ) {
75 #if 0
76         if ( calendarMode() ) {
77             const QAbstractItemModel* model = sourceIdx.model();
78             if ( model->hasChildren( sourceIdx ) ) {
79                 return BASE::mapFromSource( model->index( sourceIdx.row(),0,sourceIdx.parent() ) );
80             } else {
81                 // Map children to columns
82                 return BASE::mapFromSource( model->index( sourceIdx.row(),0,sourceIdx.parent() )
83                     .child( 0, sourceIdx.column() ) );
84             }
85         }
86 #endif
87         return BASE::mapFromSource( sourceIdx.model()->index( sourceIdx.row(),0,sourceIdx.parent() ) );
88     }
89     else return QModelIndex();
90 }

```

**12.21.3.6 QModelIndex ProxyModel::mapToSource (const QModelIndex & proxyIdx) const**

Converts indexes in the proxy model to indexes in the source model

Reimplemented from [KDGantt::ForwardingProxyModel](#).

Definition at line 92 of file `kdganttproxymodel.cpp`.

Referenced by `columnCount()`, `data()`, and `setData()`.

```

93 {
94     if( proxyIdx.isValid() ) {
95 #if 0
96         if ( calendarMode() && proxyIdx.column() > 0 ) {
97             return BASE::mapToSource( proxyIdx.model()->index( proxyIdx.column(), 0, proxyIdx ) );
98         }
99 #endif
100         return BASE::mapToSource( proxyIdx );
101     }
102     else return QModelIndex();
103 }

```

**12.21.3.7 QModelIndex ForwardingProxyModel::parent (const QModelIndex & idx) const**  
[inherited]

**See also:**

`QAbstractItemModel::parent`

Definition at line 267 of file `kdganttforwardingproxymodel.cpp`.

References `KDGantt::ForwardingProxyModel::mapFromSource()`, and `KDGantt::ForwardingProxyModel::mapToSource()`.

```
268 {
269     return mapFromSource( sourceModel()->parent( mapToSource( idx ) ) );
270 }
```

### 12.21.3.8 int ProxyModel::role (int *ganttrole*) const

Definition at line 120 of file `kdganttproxymodel.cpp`.

References `d`.

```
121 {
122     return d->roleMap[ganttrole];
123 }
```

### 12.21.3.9 int ProxyModel::rowCount (const QModelIndex & *idx*) const

See also:

`QAbstractItemModel::rowCount`

Reimplemented from [KDGantt::ForwardingProxyModel](#).

Definition at line 140 of file `kdganttproxymodel.cpp`.

```
141 {
142     // TODO
143     return BASE::rowCount( proxyIndex );
144 }
```

### 12.21.3.10 void ProxyModel::setColumn (int *ganttrole*, int *col*)

Definition at line 105 of file `kdganttproxymodel.cpp`.

References `d`.

```
106 {
107     d->columnMap[ganttrole] = col;
108 }
```

### 12.21.3.11 bool ProxyModel::setData (const QModelIndex & *idx*, const QVariant & *value*, int *role* = Qt::EditRole)

See also:

`QAbstractItemModel::setData`

Reimplemented from [KDGantt::ForwardingProxyModel](#).

Definition at line 167 of file `kdganttproxymodel.cpp`.

References `d`, and `mapToSource()`.

```

168 {
169     int srole = role;
170     int scol  = proxyIdx.column();
171     QHash<int, int>::const_iterator it = d->roleMap.find( role );
172     if ( it != d->roleMap.end() ) srole = *it;
173     it = d->columnMap.find( role );
174     if ( it != d->columnMap.end() ) scol = *it;
175
176     QAbstractItemModel* model = sourceModel();
177     return model->setData( model->index( proxyIdx.row(), scol, mapToSource( proxyIdx.parent() ) ), val
178 }

```

### 12.21.3.12 void ProxyModel::setRole (int ganttrole, int role)

Definition at line 115 of file kdganttproxymodel.cpp.

References [d](#).

```

116 {
117     d->roleMap[ganttrole] = role;
118 }

```

### 12.21.3.13 void ForwardingProxyModel::setSourceModel (QAbstractItemModel \* model) [inherited]

Sets the model to be used as the source model for this proxy. The proxy does not take ownership of the model.

**See also:**

[QAbstractProxyModel::setSourceModel](#)

Reimplemented in [KDGantt::SummaryHandlingProxyModel](#).

Definition at line 88 of file kdganttforwardingproxymodel.cpp.

References [KDGantt::ForwardingProxyModel::sourceColumnsAboutToBeInserted\(\)](#), [KDGantt::ForwardingProxyModel::sourceColumnsAboutToBeRemoved\(\)](#), [KDGantt::ForwardingProxyModel::sourceColumnsInserted\(\)](#), [KDGantt::ForwardingProxyModel::sourceColumnsRemoved\(\)](#), [KDGantt::ForwardingProxyModel::sourceDataChanged\(\)](#), [KDGantt::ForwardingProxyModel::sourceLayoutAboutToBeChanged\(\)](#), [KDGantt::ForwardingProxyModel::sourceLayoutChanged\(\)](#), [KDGantt::ForwardingProxyModel::sourceModelAboutToBeReset\(\)](#), [KDGantt::ForwardingProxyModel::sourceModelReset\(\)](#), [KDGantt::ForwardingProxyModel::sourceRowsAboutToBeInserted\(\)](#), [KDGantt::ForwardingProxyModel::sourceRowsAboutToBeRemoved\(\)](#), [KDGantt::ForwardingProxyModel::sourceRowsInserted\(\)](#), and [KDGantt::ForwardingProxyModel::sourceRowsRemoved\(\)](#).

```

89 {
90     if ( sourceModel() ) sourceModel()->disconnect( this );
91     BASE::setSourceModel( model );
92
93     if(!model) return;
94
95     connect( model, SIGNAL(modelAboutToBeReset()), this, SLOT(sourceModelAboutToBeReset()) );
96     connect( model, SIGNAL(modelReset()), this, SLOT(sourceModelReset()) );
97     connect( model, SIGNAL(layoutAboutToBeChanged()), this, SLOT(sourceLayoutAboutToBeChanged()) );
98     connect( model, SIGNAL(layoutChanged()), this, SLOT(sourceLayoutChanged()) );
99
100     connect( model, SIGNAL(dataChanged(const QModelIndex&,const QModelIndex&)),

```

```

101         this, SLOT(sourceDataChanged(const QModelIndex&,const QModelIndex&) );
102
103
104     connect( model, SIGNAL(columnsAboutToBeInserted(const QModelIndex&, int,int)),
105             this, SLOT(sourceColumnsAboutToBeInserted(const QModelIndex&,int,int)) );
106     connect( model, SIGNAL(columnsInserted(const QModelIndex&, int,int)),
107             this, SLOT(sourceColumnsInserted(const QModelIndex&,int,int)) );
108     connect( model, SIGNAL(columnsAboutToBeRemoved(const QModelIndex&, int,int)),
109             this, SLOT(sourceColumnsAboutToBeRemoved(const QModelIndex&,int,int)) );
110     connect( model, SIGNAL(columnsRemoved(const QModelIndex&, int,int)),
111             this, SLOT(sourceColumnsRemoved(const QModelIndex&,int,int)) );
112
113     connect( model, SIGNAL(rowsAboutToBeInserted(const QModelIndex&, int,int)),
114             this, SLOT(sourceRowsAboutToBeInserted(const QModelIndex&,int,int)) );
115     connect( model, SIGNAL(rowsInserted(const QModelIndex&, int,int)),
116             this, SLOT(sourceRowsInserted(const QModelIndex&,int,int)) );
117     connect( model, SIGNAL(rowsAboutToBeRemoved(const QModelIndex&, int,int)),
118             this, SLOT(sourceRowsAboutToBeRemoved(const QModelIndex&,int,int)) );
119     connect( model, SIGNAL(rowsRemoved(const QModelIndex&, int,int)),
120             this, SLOT(sourceRowsRemoved(const QModelIndex&,int,int)) );
121 }

```

#### 12.21.3.14 void ForwardingProxyModel::sourceColumnsAboutToBeInserted (const QModelIndex & parentIdx, int start, int end) [protected, virtual, slot, inherited]

Called just before columns are inserted into the source model.

##### See also:

[QAbstractItemModel::columnsAboutToBeInserted\(\)](#)

Reimplemented in [KDGantt::SummaryHandlingProxyModel](#).

Definition at line 171 of file `kdganttforwardingproxymodel.cpp`.

References [KDGantt::ForwardingProxyModel::mapFromSource\(\)](#).

Referenced by [KDGantt::ForwardingProxyModel::setSourceModel\(\)](#).

```

174 {
175     beginInsertColumns( mapFromSource( parentIdx ), start, end );
176 }

```

#### 12.21.3.15 void ForwardingProxyModel::sourceColumnsAboutToBeRemoved (const QModelIndex & parentIdx, int start, int end) [protected, virtual, slot, inherited]

Called just before columns are removed from the source model.

##### See also:

[QAbstractItemModel::columnsAboutToBeRemoved\(\)](#)

Reimplemented in [KDGantt::SummaryHandlingProxyModel](#).

Definition at line 192 of file `kdganttforwardingproxymodel.cpp`.

References [KDGantt::ForwardingProxyModel::mapFromSource\(\)](#).

Referenced by [KDGantt::ForwardingProxyModel::setSourceModel\(\)](#).

```
195 {
196     beginRemoveColumns( mapFromSource( parentIdx ), start, end );
197 }
```

### 12.21.3.16 void ForwardingProxyModel::sourceColumnsInserted (const QModelIndex & *parentIdx*, int *start*, int *end*) [protected, virtual, slot, inherited]

Called after columns have been inserted into the source model.

#### See also:

QAbstractItemModel::columnsInserted()

Definition at line 181 of file `kdganttforwardingproxymodel.cpp`.

Referenced by `KDGantt::ForwardingProxyModel::setSourceModel()`.

```
182 {
183     Q_UNUSED( parentIdx );
184     Q_UNUSED( start );
185     Q_UNUSED( end );
186     endInsertColumns();
187 }
```

### 12.21.3.17 void ForwardingProxyModel::sourceColumnsRemoved (const QModelIndex & *parentIdx*, int *start*, int *end*) [protected, virtual, slot, inherited]

Called after columns have been removed from the source model.

#### See also:

QAbstractItemModel::columnsRemoved()

Definition at line 202 of file `kdganttforwardingproxymodel.cpp`.

Referenced by `KDGantt::ForwardingProxyModel::setSourceModel()`.

```
203 {
204     Q_UNUSED( parentIdx );
205     Q_UNUSED( start );
206     Q_UNUSED( end );
207     endRemoveColumns();
208 }
```

### 12.21.3.18 void ForwardingProxyModel::sourceDataChanged (const QModelIndex & *from*, const QModelIndex & *to*) [protected, virtual, slot, inherited]

Called when the data in an existing item in the source model changes.

#### See also:

QAbstractItemModel::dataChanged()

Reimplemented in [KDGantt::SummaryHandlingProxyModel](#).

Definition at line 162 of file `kdganttforwardingproxymodel.cpp`.

References `KDGantt::ForwardingProxyModel::mapFromSource()`.

Referenced by `KDGantt::ForwardingProxyModel::setSourceModel()`.

```
163 {
164     //qDebug() << "ForwardingProxyModel::sourceDataChanged("<<from<<to<<")";
165     emit dataChanged( mapFromSource( from ), mapFromSource( to ) );
166 }
```

### 12.21.3.19 void ForwardingProxyModel::sourceLayoutAboutToBeChanged () [protected, virtual, slot, inherited]

Called just before the layout of the source model is changed.

**See also:**

`QAbstractItemModel::layoutAboutToBeChanged()`

Definition at line 144 of file `kdganttforwardingproxymodel.cpp`.

Referenced by `KDGantt::ForwardingProxyModel::setSourceModel()`.

```
145 {
146     //qDebug() << "ForwardingProxyModel::sourceLayoutAboutToBeChanged() ";
147     emit layoutAboutToBeChanged();
148 }
```

### 12.21.3.20 void ForwardingProxyModel::sourceLayoutChanged () [protected, virtual, slot, inherited]

Called when the layout of the source model has changed.

**See also:**

`QAbstractItemModel::layoutChanged()`

Reimplemented in [KDGantt::SummaryHandlingProxyModel](#).

Definition at line 153 of file `kdganttforwardingproxymodel.cpp`.

Referenced by `KDGantt::ForwardingProxyModel::setSourceModel()`.

```
154 {
155     //qDebug() << "ForwardingProxyModel::sourceLayoutChanged() ";
156     reset();
157 }
```

### 12.21.3.21 void ForwardingProxyModel::sourceModelAboutToBeReset () [protected, virtual, slot, inherited]

Called when the source model is about to be reset.

**See also:**

[QAbstractItemModel::modelAboutToBeReset\(\)](#)

Definition at line 126 of file `kdganttforwardingproxymodel.cpp`.

Referenced by `KDGantt::ForwardingProxyModel::setSourceModel()`.

```
127 {
128     // The matching signal is emitted be reset()
129 }
```

### 12.21.3.22 void ForwardingProxyModel::sourceModelReset () [protected, virtual, slot, inherited]

Called when the source model is reset

**See also:**

[QAbstractItemModel::modelReset\(\)](#)

Reimplemented in [KD Gantt::SummaryHandlingProxyModel](#).

Definition at line 134 of file `kdganttforwardingproxymodel.cpp`.

Referenced by `KDGantt::ForwardingProxyModel::setSourceModel()`.

```
135 {
136     //qDebug() << "ForwardingProxyModel::sourceModelReset()";
137     reset();
138 }
```

### 12.21.3.23 void ForwardingProxyModel::sourceRowsAboutToBeInserted (const QModelIndex & parentIdx, int start, int end) [protected, virtual, slot, inherited]

Called just before rows are inserted into the source model.

**See also:**

[QAbstractItemModel::rowsAboutToBeInserted\(\)](#)

Reimplemented in [KD Gantt::SummaryHandlingProxyModel](#).

Definition at line 213 of file `kdganttforwardingproxymodel.cpp`.

References `KDGantt::ForwardingProxyModel::mapFromSource()`.

Referenced by `KDGantt::ForwardingProxyModel::setSourceModel()`.

```
214 {
215     beginInsertRows( mapFromSource( parentIdx ), start, end );
216 }
```

**12.21.3.24 void ForwardingProxyModel::sourceRowsAboutToBeRemoved (const QModelIndex & parentIdx, int start, int end)** [protected, virtual, slot, inherited]

Called just before rows are removed from the source model.

**See also:**

QAbstractItemModel::rowsAboutToBeRemoved()

Reimplemented in [KDGantt::SummaryHandlingProxyModel](#).

Definition at line 232 of file `kdganttforwardingproxymodel.cpp`.

References `KDGantt::ForwardingProxyModel::mapFromSource()`.

Referenced by `KDGantt::ForwardingProxyModel::setSourceModel()`.

```
233 {
234     beginRemoveRows( mapFromSource( parentIdx ), start, end );
235 }
```

**12.21.3.25 void ForwardingProxyModel::sourceRowsInserted (const QModelIndex & parentIdx, int start, int end)** [protected, virtual, slot, inherited]

Called after rows have been inserted into the source model.

**See also:**

QAbstractItemModel::rowsInserted()

Definition at line 221 of file `kdganttforwardingproxymodel.cpp`.

Referenced by `KDGantt::ForwardingProxyModel::setSourceModel()`.

```
222 {
223     Q_UNUSED( parentIdx );
224     Q_UNUSED( start );
225     Q_UNUSED( end );
226     endInsertRows();
227 }
```

**12.21.3.26 void ForwardingProxyModel::sourceRowsRemoved (const QModelIndex & parentIdx, int start, int end)** [protected, virtual, slot, inherited]

Called after rows have been removed from the source model.

**See also:**

QAbstractItemModel::rowsRemoved()

Definition at line 240 of file `kdganttforwardingproxymodel.cpp`.

Referenced by `KDGantt::ForwardingProxyModel::setSourceModel()`.

```
241 {
242     Q_UNUSED( parentIdx );
243     Q_UNUSED( start );
244     Q_UNUSED( end );
245     endRemoveRows();
246 }
```

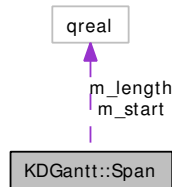
The documentation for this class was generated from the following files:

- [kdganttproxymodel.h](#)
- [kdganttproxymodel.cpp](#)

## 12.22 KDGantt::Span Class Reference

```
#include <KDGanttGlobal>
```

Collaboration diagram for KDGantt::Span:



### 12.22.1 Detailed Description

A class representing a start point and a length.

Definition at line 212 of file kdganttglobal.h.

### Public Member Functions

- qreal [end](#) () const
- bool [equals](#) (const [Span](#) &other) const
- bool [isValid](#) () const
- qreal [length](#) () const
- [Span](#) & [operator=](#) (const [Span](#) &other)
- void [setEnd](#) (qreal end)
- void [setLength](#) (qreal length)
- void [setStart](#) (qreal start)
- [Span](#) (const [Span](#) &other)
- [Span](#) (qreal start, qreal length)
- [Span](#) ()
- qreal [start](#) () const
- [~Span](#) ()

### 12.22.2 Constructor & Destructor Documentation

#### 12.22.2.1 KDGantt::Span::Span ()

Definition at line 216 of file kdganttglobal.h.

```
216 : m_start( -1 ), m_length( 0 ) {}
```

#### 12.22.2.2 KDGantt::Span::Span (qreal start, qreal length)

Definition at line 217 of file kdganttglobal.h.

```
217 : m_start( start ), m_length( length ) {}
```

**12.22.2.3 KDGantt::Span::Span (const [Span](#) & *other*)**

Definition at line 218 of file kdganttglobal.h.

```
218 : m_start(other.m_start), m_length(other.m_length) {}
```

**12.22.2.4 Span::~~Span ()**

Definition at line 68 of file kdganttglobal.cpp.

```
69 {
70 }
```

**12.22.3 Member Function Documentation****12.22.3.1 qreal KDGantt::Span::end () const**

Definition at line 226 of file kdganttglobal.h.

Referenced by KDGantt::AbstractGrid::isSatisfiedConstraint().

```
226 { return m_start+m_length; }
```

**12.22.3.2 bool KDGantt::Span::equals (const [Span](#) & *other*) const**

Definition at line 233 of file kdganttglobal.h.

References `m_length`, and `m_start`.

Referenced by KDGantt::operator!=(()), and KDGantt::operator==(()).

```
233                                     {
234     return m_start == other.m_start && m_length == other.m_length;
235 }
```

**12.22.3.3 bool KDGantt::Span::isValid () const**

Definition at line 231 of file kdganttglobal.h.

Referenced by KDAB\_SCOPED\_UNITTEST\_SIMPLE().

```
231 { return m_start >= 0.;}
```

**12.22.3.4 qreal KDGantt::Span::length () const**

Definition at line 229 of file kdganttglobal.h.

Referenced by KDAB\_SCOPED\_UNITTEST\_SIMPLE(), KDGantt::DateTimeGrid::mapFromChart(), operator<<(), KDGantt::GraphicsScene::print(), and KDGantt::GraphicsItem::updateItem().

```
229 { return m_length; }
```

### 12.22.3.5 [Span](#)& `KDGantt::Span::operator=` (const [Span](#) & *other*)

Definition at line 221 of file `kdganttglobal.h`.

References `m_length`, and `m_start`.

```
221 { m_start=other.m_start; m_length=other.m_length; return *this; }
```

### 12.22.3.6 `void KDGantt::Span::setEnd` (qreal *end*)

Definition at line 225 of file `kdganttglobal.h`.

Referenced by `KDAB_SCOPED_UNITTEST_SIMPLE()`.

```
225 { m_length = m_start-end; }
```

### 12.22.3.7 `void KDGantt::Span::setLength` (qreal *length*)

Definition at line 228 of file `kdganttglobal.h`.

Referenced by `KDAB_SCOPED_UNITTEST_SIMPLE()`.

```
228 { m_length=length; }
```

### 12.22.3.8 `void KDGantt::Span::setStart` (qreal *start*)

Definition at line 223 of file `kdganttglobal.h`.

Referenced by `KDAB_SCOPED_UNITTEST_SIMPLE()`.

```
223 { m_start=start; }
```

### 12.22.3.9 `qreal KDGantt::Span::start` () const

Definition at line 224 of file `kdganttglobal.h`.

Referenced by `KDAB_SCOPED_UNITTEST_SIMPLE()`, `KDGantt::DateTimeGrid::mapFromChart()`, `operator<<()`, `KDGantt::DateTimeGrid::paintGrid()`, `KDGantt::GraphicsScene::print()`, and `KDGantt::GraphicsItem::updateItem()`.

```
224 { return m_start; }
```

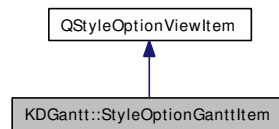
The documentation for this class was generated from the following files:

- [kdganttglobal.h](#)
- [kdganttglobal.cpp](#)

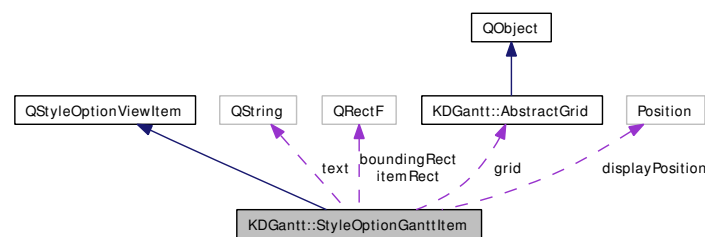
## 12.23 KDGantt::StyleOptionGanttItem Class Reference

```
#include <KDGanttStyleOptionGanttItem>
```

Inheritance diagram for KDGantt::StyleOptionGanttItem:



Collaboration diagram for KDGantt::StyleOptionGanttItem:



### 12.23.1 Detailed Description

QStyleOption subclass for gantt items.

Definition at line 36 of file `kdganttstyleoptionganttitem.h`.

#### Public Types

- enum [Position](#) {
  - [Left](#),
  - [Right](#),
  - [Center](#) }

#### Public Member Functions

- [StyleOptionGanttItem](#) & `operator=` (const [StyleOptionGanttItem](#) &other)
- [StyleOptionGanttItem](#) (const [StyleOptionGanttItem](#) &other)
- [StyleOptionGanttItem](#) ()

#### Public Attributes

- [QRectF](#) `boundingRect`
- [Position](#) `displayPosition`
- [AbstractGrid](#) \* `grid`
- [QRectF](#) `itemRect`
- [QString](#) `text`

## 12.23.2 Member Enumeration Documentation

### 12.23.2.1 enum [KDGantt::StyleOptionGanttItem::Position](#)

This enum is used to describe where the Qt::DisplayRole (the label) should be located relative to the item itself.

**Enumerator:**

*Left*  
*Right*  
*Center*

Definition at line 38 of file `kdganttstyleoptionganttitem.h`.

```
38 { Left, Right, Center };
```

## 12.23.3 Constructor & Destructor Documentation

### 12.23.3.1 [StyleOptionGanttItem::StyleOptionGanttItem \(\)](#)

Constructor. Sets grid to 0.

Definition at line 37 of file `kdganttstyleoptionganttitem.cpp`.

```
38     : BASE(),
39       grid( 0 )
40 {
41     type = QStyleOption::SO_CustomBase+89;
42     version = 1;
43 }
```

### 12.23.3.2 [StyleOptionGanttItem::StyleOptionGanttItem \(const \[StyleOptionGanttItem\]\(#\) & other\)](#)

Copy constructor. Creates a copy of *other*

Definition at line 46 of file `kdganttstyleoptionganttitem.cpp`.

References `operator=()`.

```
47     : BASE(other)
48 {
49     operator=( other );
50 }
```

## 12.23.4 Member Function Documentation

### 12.23.4.1 [StyleOptionGanttItem](#) & [StyleOptionGanttItem::operator= \(const \[StyleOptionGanttItem\]\(#\) & other\)](#)

Assignment operator

Definition at line 53 of file `kdganttstyleoptionganttitem.cpp`.

References `boundingRect`, `displayPosition`, `grid`, `itemRect`, and `text`.

Referenced by `StyleOptionGanttItem()`.

```
54 {
55     BASE::operator=( other );
56     boundingRect = other.boundingRect;
57     itemRect = other.itemRect;
58     displayPosition = other.displayPosition;
59     grid = other.grid;
60     text = other.text;
61     return *this;
62 }
```

## 12.23.5 Member Data Documentation

### 12.23.5.1 [StyleOptionGanttItem::boundingRect](#)

Contains the bounding rectangle for the item

Definition at line 44 of file `kdganttstyleoptionganttitem.h`.

Referenced by `KDGantt::Legend::drawItem()`, `operator<<()`, `operator=()`, and `KDGantt::ItemDelegate::paintGanttItem()`.

### 12.23.5.2 [StyleOptionGanttItem::displayPosition](#)

See also:

[StyleOptionGanttItem::Position](#).

Definition at line 46 of file `kdganttstyleoptionganttitem.h`.

Referenced by `KDGantt::Legend::getStyleOption()`, `KDGantt::ItemDelegate::itemBoundingSpan()`, `operator<<()`, `operator=()`, and `KDGantt::ItemDelegate::paintGanttItem()`.

### 12.23.5.3 [StyleOptionGanttItem::grid](#)

Contains a pointer to the [AbstractGrid](#) used by the view

Definition at line 47 of file `kdganttstyleoptionganttitem.h`.

Referenced by `operator<<()`, and `operator=()`.

### 12.23.5.4 [StyleOptionGanttItem::itemRect](#)

Contains the "active" item rectangle that corresponds to the values from the model.

Definition at line 45 of file `kdganttstyleoptionganttitem.h`.

Referenced by `KDGantt::Legend::drawItem()`, `KDGantt::ItemDelegate::interactionStateFor()`, `KDGantt::ItemDelegate::itemBoundingSpan()`, `operator<<()`, `operator=()`, and `KDGantt::ItemDelegate::paintGanttItem()`.

### 12.23.5.5 [StyleOptionGanttItem::text](#)

Contains a string printed to the item

Definition at line 48 of file `kdganttstyleoptionganttitem.h`.

Referenced by `KDGantt::Legend::drawItem()`, `KDGantt::Legend::getStyleOption()`, `operator<<()`, `operator=()`, and `KDGantt::ItemDelegate::paintGanttItem()`.

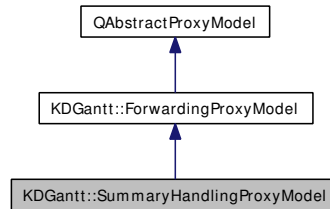
The documentation for this class was generated from the following files:

- [kdganttstyleoptionganttitem.h](#)
- [kdganttstyleoptionganttitem.cpp](#)

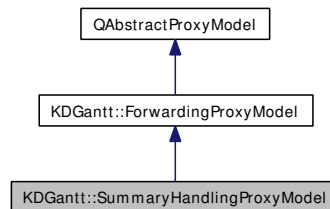
## 12.24 KDGantt::SummaryHandlingProxyModel Class Reference

```
#include <kdganttsummaryhandlingproxymodel.h>
```

Inheritance diagram for KDGantt::SummaryHandlingProxyModel:



Collaboration diagram for KDGantt::SummaryHandlingProxyModel:



### 12.24.1 Detailed Description

Proxy model that supports summary gantt items.

This proxy model provides the functionality of summary items. A summary item is an item with type [KDGantt::TypeSummary](#) and zero or more children in the model that it summarizes. [GraphicsView](#) itself does not dictate any policy for summary items, instead the logic for making the summary items start and end points span it's children is provided by this proxy.

The start and end times of a summary is the min/max of the start/end times of it's children.

**See also:**

[GraphicsView::setModel](#)

Definition at line 31 of file `kdganttsummaryhandlingproxymodel.h`.

### Public Member Functions

- `int` [columnCount](#) (const `QModelIndex &idx=QModelIndex()`) const
- `QVariant` [data](#) (const `QModelIndex &proxyIndex`, int role=`Qt::DisplayRole`) const
- `Qt::ItemFlags` [flags](#) (const `QModelIndex &idx`) const
- `QModelIndex` [index](#) (int row, int column, const `QModelIndex &parent=QModelIndex()`) const
- `QModelIndex` [mapFromSource](#) (const `QModelIndex &sourceIndex`) const
- `QModelIndex` [mapToSource](#) (const `QModelIndex &proxyIndex`) const
- `QModelIndex` [parent](#) (const `QModelIndex &idx`) const
- `int` [rowCount](#) (const `QModelIndex &idx=QModelIndex()`) const

- bool [setData](#) (const QModelIndex &index, const QVariant &value, int role=Qt::EditRole)
- void [setSourceModel](#) (QAbstractItemModel \*model)
- [SummaryHandlingProxyModel](#) (QObject \*parent=0)
- virtual [~SummaryHandlingProxyModel](#) ()

## Protected Slots

- virtual void [sourceColumnsInserted](#) (const QModelIndex &idx, int start, int end)
- virtual void [sourceColumnsRemoved](#) (const QModelIndex &idx, int start, int end)
- virtual void [sourceLayoutAboutToBeChanged](#) ()
- virtual void [sourceModelAboutToBeReset](#) ()
- virtual void [sourceRowsInserted](#) (const QModelIndex &idx, int start, int end)
- virtual void [sourceRowsRemoved](#) (const QModelIndex &, int start, int end)

## Protected Member Functions

- void [sourceColumnsAboutToBeInserted](#) (const QModelIndex &idx, int start, int end)
- void [sourceColumnsAboutToBeRemoved](#) (const QModelIndex &idx, int start, int end)
- void [sourceDataChanged](#) (const QModelIndex &from, const QModelIndex &to)
- void [sourceLayoutChanged](#) ()
- void [sourceModelReset](#) ()
- void [sourceRowsAboutToBeInserted](#) (const QModelIndex &idx, int start, int end)
- void [sourceRowsAboutToBeRemoved](#) (const QModelIndex &, int start, int end)

## 12.24.2 Constructor & Destructor Documentation

### 12.24.2.1 [SummaryHandlingProxyModel::SummaryHandlingProxyModel](#) (QObject \*parent = 0) [explicit]

Constructor. Creates a new [SummaryHandlingProxyModel](#) with parent *parent*

Definition at line 120 of file `kdganttsummaryhandlingproxymodel.cpp`.

```

121     : BASE( parent ), _d( new Private )
122 {
123     init();
124 }
```

### 12.24.2.2 [SummaryHandlingProxyModel::~~SummaryHandlingProxyModel](#) () [virtual]

Definition at line 127 of file `kdganttsummaryhandlingproxymodel.cpp`.

```

128 {
129 }
```

### 12.24.3 Member Function Documentation

#### 12.24.3.1 `int ForwardingProxyModel::columnCount (const QModelIndex & idx = QModelIndex()) const` [inherited]

See also:

`QAbstractItemModel::columnCount`

Reimplemented in [KD Gantt::ProxyModel](#).

Definition at line 255 of file `kdganttforwardingproxymodel.cpp`.

References `KD Gantt::ForwardingProxyModel::mapToSource()`.

```
256 {
257     return sourceModel()->columnCount( mapToSource( idx ) );
258 }
```

#### 12.24.3.2 `QVariant SummaryHandlingProxyModel::data (const QModelIndex & proxyIndex, int role = Qt::DisplayRole) const`

See also:

`QAbstractItemModel::data`

Definition at line 225 of file `kdganttsummaryhandlingproxymodel.cpp`.

References `d`, `KD Gantt::EndTimeRole`, `KD Gantt::ForwardingProxyModel::mapToSource()`, and `KD Gantt::StartTimeRole`.

Referenced by `KDAB_SCOPED_UNITTEST_SIMPLE()`.

```
226 {
227     //qDebug() << "SummaryHandlingProxyModel::data("<<proxyIndex<<role<<")";
228     const QModelIndex sidx = mapToSource( proxyIndex );
229     const QAbstractItemModel* model = sourceModel();
230     if ( d->isSummary(sidx) && ( role==StartTimeRole || role==EndTimeRole ) ) {
231         //qDebug() << "requested summary";
232         QPair<QDateTime,QDateTime> result;
233         if ( d->cacheLookup( sidx, &result ) ) {
234             //qDebug() << "SummaryHandlingProxyModel::data(): Looking up summary for " << proxyIndex <<
235                 switch( role ) {
236                     case StartTimeRole: return result.first;
237                     case EndTimeRole: return result.second;
238                     default: /* fall thru */;
239                 }
240         } else {
241             d->insertInCache( this, sidx );
242             return data( proxyIndex, role ); /* TODO: Optimise */
243         }
244     }
245     return model->data( sidx, role );
246 }
```

#### 12.24.3.3 `Qt::ItemFlags SummaryHandlingProxyModel::flags (const QModelIndex & idx) const`

See also:

`QAbstractItemModel::flags`

Definition at line 213 of file `kdganttsummaryhandlingproxymodel.cpp`.

References `d`, and `KDGantt::ForwardingProxyModel::mapToSource()`.

Referenced by `KDAB_SCOPED_UNITTEST_SIMPLE()`.

```

214 {
215     const QModelIndex sidx = mapToSource( idx );
216     const QAbstractItemModel* model = sourceModel();
217     Qt::ItemFlags f = model->flags( sidx );
218     if ( d->isSummary(sidx) ) {
219         f &= !Qt::ItemIsEditable;
220     }
221     return f;
222 }

```

#### 12.24.3.4 QModelIndex ForwardingProxyModel::index (int row, int column, const QModelIndex & parent = QModelIndex()) const [inherited]

See also:

`QAbstractItemModel::index`

Definition at line 261 of file `kdganttforwardingproxymodel.cpp`.

References `KDGantt::ForwardingProxyModel::mapFromSource()`, and `KDGantt::ForwardingProxyModel::mapToSource()`.

Referenced by `KDAB_SCOPED_UNITTEST_SIMPLE()`.

```

262 {
263     return mapFromSource( sourceModel()->index( row, column, mapToSource( parent ) ) );
264 }

```

#### 12.24.3.5 QModelIndex ForwardingProxyModel::mapFromSource (const QModelIndex & sourceIndex) const [inherited]

Converts indexes in the source model to indexes in the proxy model

Reimplemented in [KDGantt::ProxyModel](#).

Definition at line 46 of file `kdganttforwardingproxymodel.cpp`.

Referenced by `KDGantt::ForwardingProxyModel::index()`, `KDGantt::ForwardingProxyModel::parent()`, `setData()`, `KDGantt::ForwardingProxyModel::sourceColumnsAboutToBeInserted()`, `KDGantt::ForwardingProxyModel::sourceColumnsAboutToBeRemoved()`, `sourceDataChanged()`, `KDGantt::ForwardingProxyModel::sourceDataChanged()`, `KDGantt::ForwardingProxyModel::sourceRowsAboutToBeInserted()`, and `KDGantt::ForwardingProxyModel::sourceRowsAboutToBeRemoved()`.

```

47 {
48     if ( !sourceIndex.isValid() )
49         return QModelIndex();
50     assert( sourceIndex.model() == sourceModel() );
51
52     // Create an index that preserves the internal pointer from the source;
53     // this way KDDataConverterProxyModel preserves the structure of the source model
54     return createIndex( sourceIndex.row(), sourceIndex.column(), sourceIndex.internalPointer() );
55 }

```

### 12.24.3.6 QModelIndex ForwardingProxyModel::mapToSource (const QModelIndex & proxyIndex) const [inherited]

Converts indexes in the proxy model to indexes in the source model

Reimplemented in [KD Gantt::ProxyModel](#).

Definition at line 67 of file `kdganttforwardingproxymodel.cpp`.

Referenced by `KDGantt::ForwardingProxyModel::columnCount()`, `data()`, `flags()`, `KDGantt::ForwardingProxyModel::index()`, `KDGantt::ForwardingProxyModel::parent()`, `KDGantt::ForwardingProxyModel::rowCount()`, `setData()`, and `KDGantt::ForwardingProxyModel::setData()`.

```

68 {
69     if ( !proxyIndex.isValid() )
70         return QModelIndex();
71     assert( proxyIndex.model() == this );
72     // So here we need to create a source index which holds that internal pointer.
73     // No way to pass it to sourceModel()->index... so we have to do the ugly way:
74     QModelIndex sourceIndex;
75     KDPriateModelIndex* hack = reinterpret_cast<KDPriateModelIndex*>(&sourceIndex);
76     hack->r = proxyIndex.row();
77     hack->c = proxyIndex.column();
78     hack->p = proxyIndex.internalPointer();
79     hack->m = sourceModel();
80     assert( sourceIndex.isValid() );
81     return sourceIndex;
82 }

```

### 12.24.3.7 QModelIndex ForwardingProxyModel::parent (const QModelIndex & idx) const [inherited]

See also:

`QAbstractItemModel::parent`

Definition at line 267 of file `kdganttforwardingproxymodel.cpp`.

References `KDGantt::ForwardingProxyModel::mapFromSource()`, and `KDGantt::ForwardingProxyModel::mapToSource()`.

```

268 {
269     return mapFromSource( sourceModel()->parent( mapToSource( idx ) ) );
270 }

```

### 12.24.3.8 int ForwardingProxyModel::rowCount (const QModelIndex & idx = QModelIndex()) const [inherited]

See also:

`QAbstractItemModel::rowCount`

Reimplemented in [KD Gantt::ProxyModel](#).

Definition at line 249 of file `kdganttforwardingproxymodel.cpp`.

References `KDGantt::ForwardingProxyModel::mapToSource()`.

```

250 {
251     return sourceModel()->rowCount( mapToSource( idx ) );
252 }

```

### 12.24.3.9 bool SummaryHandlingProxyModel::setData (const QModelIndex & *index*, const QVariant & *value*, int *role* = Qt::EditRole)

See also:

QAbstractItemModel::setData

Reimplemented from [KDGantt::ForwardingProxyModel](#).

Definition at line 249 of file `kdganttsummaryhandlingproxymodel.cpp`.

References `d`, `KDGantt::EndTimeRole`, `KDGantt::ForwardingProxyModel::mapFromSource()`, `KDGantt::ForwardingProxyModel::mapToSource()`, and `KDGantt::StartTimeRole`.

```

250 {
251     QAbstractItemModel* model = sourceModel();
252     if ( role==StartTimeRole || role==EndTimeRole ) {
253         QModelIndex parentIdx = mapToSource( index );
254         do {
255             if ( d->isSummary(parentIdx) ) {
256                 //qDebug() << "removing " << parentIdx << "from cache";
257                 d->removeFromCache( parentIdx );
258                 QModelIndex proxyParentIdx = mapFromSource( parentIdx );
259                 emit dataChanged( proxyParentIdx, proxyParentIdx );
260             }
261         } while ( ( parentIdx=model->parent( parentIdx ) ) != QModelIndex() );
262     }
263     return BASE::setData( index, value, role );
264 }

```

### 12.24.3.10 void SummaryHandlingProxyModel::setSourceModel (QAbstractItemModel \* *model*)

Sets the model to be used as the source model for this proxy. The proxy does not take ownership of the model.

See also:

QAbstractProxyModel::setSourceModel

Reimplemented from [KDGantt::ForwardingProxyModel](#).

Definition at line 149 of file `kdganttsummaryhandlingproxymodel.cpp`.

References `d`.

Referenced by `KDAB_SCOPED_UNITTEST_SIMPLE()`.

```

150 {
151     BASE::setSourceModel( model );
152     d->clearCache();
153 }

```

**12.24.3.11 void SummaryHandlingProxyModel::sourceColumnsAboutToBeInserted (const QModelIndex & *idx*, int *start*, int *end*)** [protected, virtual]

Called just before columns are inserted into the source model.

**See also:**

QAbstractItemModel::columnsAboutToBeInserted()

Reimplemented from [KDGantt::ForwardingProxyModel](#).

Definition at line 184 of file `kdganttsummaryhandlingproxymodel.cpp`.

References [d](#).

```
187 {
188     BASE::sourceColumnsAboutToBeInserted( parentIdx, start, end );
189     d->clearCache();
190 }
```

**12.24.3.12 void SummaryHandlingProxyModel::sourceColumnsAboutToBeRemoved (const QModelIndex & *idx*, int *start*, int *end*)** [protected, virtual]

Called just before columns are removed from the source model.

**See also:**

QAbstractItemModel::columnsAboutToBeRemoved()

Reimplemented from [KDGantt::ForwardingProxyModel](#).

Definition at line 192 of file `kdganttsummaryhandlingproxymodel.cpp`.

References [d](#).

```
195 {
196     BASE::sourceColumnsAboutToBeRemoved( parentIdx, start, end );
197     d->clearCache();
198 }
```

**12.24.3.13 void ForwardingProxyModel::sourceColumnsInserted (const QModelIndex & *parentIdx*, int *start*, int *end*)** [protected, virtual, slot, inherited]

Called after columns have been inserted into the source model.

**See also:**

QAbstractItemModel::columnsInserted()

Definition at line 181 of file `kdganttforwardingproxymodel.cpp`.

Referenced by [KDGantt::ForwardingProxyModel::setSourceModel\(\)](#).

```
182 {
183     Q_UNUSED( parentIdx );
184     Q_UNUSED( start );
185     Q_UNUSED( end );
186     endInsertColumns();
187 }
```

#### 12.24.3.14 void ForwardingProxyModel::sourceColumnsRemoved (const QModelIndex & *parentIdx*, int *start*, int *end*) [protected, virtual, slot, inherited]

Called after columns have been removed from the source model.

**See also:**

QAbstractItemModel::columnsRemoved()

Definition at line 202 of file `kdganttforwardingproxymodel.cpp`.

Referenced by `KDGantt::ForwardingProxyModel::setSourceModel()`.

```
203 {
204     Q_UNUSED( parentIdx );
205     Q_UNUSED( start );
206     Q_UNUSED( end );
207     endRemoveColumns();
208 }
```

#### 12.24.3.15 void SummaryHandlingProxyModel::sourceDataChanged (const QModelIndex & *from*, const QModelIndex & *to*) [protected, virtual]

Called when the data in an existing item in the source model changes.

**See also:**

QAbstractItemModel::dataChanged()

Reimplemented from [KDGantt::ForwardingProxyModel](#).

Definition at line 167 of file `kdganttsummaryhandlingproxymodel.cpp`.

References `d`, `KDGantt::ItemTypeRole`, `KDGantt::ForwardingProxyModel::mapFromSource()`, and `KDGantt::TypeSummary`.

```
168 {
169     QAbstractItemModel* model = sourceModel();
170     QModelIndex parentIdx = from;
171     do {
172         const QModelIndex& dataIdx = parentIdx;
173         if ( model->data( dataIdx, ItemTypeRole )==TypeSummary ) {
174             //qDebug() << "removing " << parentIdx << "from cache";
175             d->removeFromCache( dataIdx );
176             QModelIndex proxyDataIdx = mapFromSource( dataIdx );
177             emit dataChanged( proxyDataIdx, proxyDataIdx );
178         }
179     } while ( ( parentIdx=model->parent( parentIdx ) ) != QModelIndex() );
180
181     BASE::sourceDataChanged( from, to );
182 }
```

#### 12.24.3.16 void ForwardingProxyModel::sourceLayoutAboutToBeChanged () [protected, virtual, slot, inherited]

Called just before the layout of the source model is changed.

**See also:**

[QAbstractItemModel::layoutAboutToBeChanged\(\)](#)

Definition at line 144 of file `kdganttforwardingproxymodel.cpp`.

Referenced by `KDGantt::ForwardingProxyModel::setSourceModel()`.

```
145 {
146     //qDebug() << "ForwardingProxyModel::sourceLayoutAboutToBeChanged() ";
147     emit layoutAboutToBeChanged();
148 }
```

**12.24.3.17 void SummaryHandlingProxyModel::sourceLayoutChanged ()** [protected, virtual]

Called when the layout of the source model has changed.

**See also:**

[QAbstractItemModel::layoutChanged\(\)](#)

Reimplemented from [KD Gantt::ForwardingProxyModel](#).

Definition at line 161 of file `kdganttsummaryhandlingproxymodel.cpp`.

References `d`.

```
162 {
163     d->clearCache();
164     BASE::sourceLayoutChanged();
165 }
```

**12.24.3.18 void ForwardingProxyModel::sourceModelAboutToBeReset ()** [protected, virtual, slot, inherited]

Called when the source model is about to be reset.

**See also:**

[QAbstractItemModel::modelAboutToBeReset\(\)](#)

Definition at line 126 of file `kdganttforwardingproxymodel.cpp`.

Referenced by `KDGantt::ForwardingProxyModel::setSourceModel()`.

```
127 {
128     // The matching signal is emitted be reset()
129 }
```

**12.24.3.19 void SummaryHandlingProxyModel::sourceModelReset ()** [protected, virtual]

Called when the source model is reset

**See also:**

[QAbstractItemModel::modelReset\(\)](#)

Reimplemented from [KDGantt::ForwardingProxyModel](#).

Definition at line 155 of file `kdganttsummaryhandlingproxymodel.cpp`.

References d.

```
156 {
157     d->clearCache();
158     BASE::sourceModelReset();
159 }
```

**12.24.3.20 void SummaryHandlingProxyModel::sourceRowsAboutToBeInserted (const QModelIndex & idx, int start, int end) [protected, virtual]**

Called just before rows are inserted into the source model.

**See also:**

[QAbstractItemModel::rowsAboutToBeInserted\(\)](#)

Reimplemented from [KDGantt::ForwardingProxyModel](#).

Definition at line 200 of file `kdganttsummaryhandlingproxymodel.cpp`.

References d.

```
201 {
202     BASE::sourceRowsAboutToBeInserted( parentIdx, start, end );
203     d->clearCache();
204 }
```

**12.24.3.21 void SummaryHandlingProxyModel::sourceRowsAboutToBeRemoved (const QModelIndex & parentIdx, int start, int end) [protected, virtual]**

Called just before rows are removed from the source model.

**See also:**

[QAbstractItemModel::rowsAboutToBeRemoved\(\)](#)

Reimplemented from [KDGantt::ForwardingProxyModel](#).

Definition at line 206 of file `kdganttsummaryhandlingproxymodel.cpp`.

References d.

```
207 {
208     BASE::sourceRowsAboutToBeRemoved( parentIdx, start, end );
209     d->clearCache();
210 }
```

**12.24.3.22 void ForwardingProxyModel::sourceRowsInserted (const QModelIndex & *parentIdx*, int *start*, int *end*)** [protected, virtual, slot, inherited]

Called after rows have been inserted into the source model.

**See also:**

QAbstractItemModel::rowsInserted()

Definition at line 221 of file `kdganttforwardingproxymodel.cpp`.

Referenced by `KDGantt::ForwardingProxyModel::setSourceModel()`.

```
222 {
223     Q_UNUSED( parentIdx );
224     Q_UNUSED( start );
225     Q_UNUSED( end );
226     endInsertRows();
227 }
```

**12.24.3.23 void ForwardingProxyModel::sourceRowsRemoved (const QModelIndex & *parentIdx*, int *start*, int *end*)** [protected, virtual, slot, inherited]

Called after rows have been removed from the source model.

**See also:**

QAbstractItemModel::rowsRemoved()

Definition at line 240 of file `kdganttforwardingproxymodel.cpp`.

Referenced by `KDGantt::ForwardingProxyModel::setSourceModel()`.

```
241 {
242     Q_UNUSED( parentIdx );
243     Q_UNUSED( start );
244     Q_UNUSED( end );
245     endRemoveRows();
246 }
```

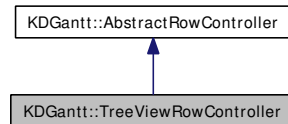
The documentation for this class was generated from the following files:

- [kdganttsummaryhandlingproxymodel.h](#)
- [kdganttsummaryhandlingproxymodel.cpp](#)

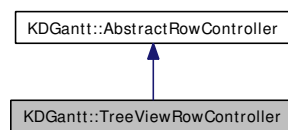
## 12.25 KDGantt::TreeViewRowController Class Reference

```
#include <kdgantttreeviewrowcontroller.h>
```

Inheritance diagram for KDGantt::TreeViewRowController:



Collaboration diagram for KDGantt::TreeViewRowController:



### 12.25.1 Detailed Description

Definition at line 34 of file kdgantttreeviewrowcontroller.h.

#### Public Member Functions

- int [headerHeight](#) () const
- QModelIndex [indexAbove](#) (const QModelIndex &idx) const
- QModelIndex [indexAt](#) (int height) const
- QModelIndex [indexBelow](#) (const QModelIndex &idx) const
- bool [isRowVisible](#) (const QModelIndex &idx) const
- int [maximumItemHeight](#) () const
- [Span rowGeometry](#) (const QModelIndex &idx) const
- [TreeViewRowController](#) (QTreeView \*tv, [QAbstractProxyModel](#) \*proxy)
- virtual [~TreeViewRowController](#) ()

### 12.25.2 Constructor & Destructor Documentation

#### 12.25.2.1 TreeViewRowController::TreeViewRowController (QTreeView \* tv, [QAbstractProxyModel](#) \* proxy)

Definition at line 40 of file kdgantttreeviewrowcontroller.cpp.

```

42   : _d( new Private )
43   {
44       _d->treeview = static_cast<Private::HackTreeView*>(tv);
45       _d->proxy = proxy;
46   }
  
```

### 12.25.2.2 TreeViewRowController::~~TreeViewRowController () [virtual]

Definition at line 48 of file kdgantttreeviewrowcontroller.cpp.

```
49 {
50     delete _d; _d=0;
51 }
```

## 12.25.3 Member Function Documentation

### 12.25.3.1 int TreeViewRowController::headerHeight () const [virtual]

#### Returns:

The height of the header part of the view.

Implement this to control how much space is reserved at the top of the view for a header

Implements [KDGantt::AbstractRowController](#).

Definition at line 55 of file kdgantttreeviewrowcontroller.cpp.

References [d](#).

```
56 {
57     //return d->treeview->header()->sizeHint().height();
58     return d->treeview->viewport()->y()-d->treeview->frameWidth();
59 }
```

### 12.25.3.2 QModelIndex TreeViewRowController::indexAbove (const QModelIndex & *idx*) const [virtual]

#### Returns:

The modelindex for the previous row before *idx*.

#### See also:

[QTreeView::indexAbove](#)

Implements [KDGantt::AbstractRowController](#).

Definition at line 87 of file kdgantttreeviewrowcontroller.cpp.

References [d](#).

```
88 {
89     const QModelIndex idx = d->proxy->mapToSource( _idx );
90     return d->proxy->mapFromSource( d->treeview->indexAbove( idx ) );
91 }
```

### 12.25.3.3 QModelIndex TreeViewRowController::indexAt (int *height*) const [virtual]

Implements [KDGantt::AbstractRowController](#).

Definition at line 82 of file kdgantttreeviewrowcontroller.cpp.

References [d](#).

```

83 {
84     return d->proxy->mapFromSource( d->treeview->indexAt( QPoint( 1,height ) ) );
85 }

```

#### 12.25.3.4 `QModelIndex TreeViewRowController::indexBelow( const QModelIndex & idx ) const` [virtual]

##### Returns:

The modelindex for the next row after *idx*.

##### See also:

`QTreeView::indexBelow`

Implements [KDGantt::AbstractRowController](#).

Definition at line 93 of file `kdgantttreeviewrowcontroller.cpp`.

References `d`.

```

94 {
95     const QModelIndex idx = d->proxy->mapToSource( _idx );
96     return d->proxy->mapFromSource( d->treeview->indexBelow( idx ) );
97 }

```

#### 12.25.3.5 `bool TreeViewRowController::isRowVisible( const QModelIndex & idx ) const` [virtual]

##### Returns:

true if the row containing index *idx* is visible in the view.

Implement this to allow [KDGantt](#) to optimise how items on screen are created. It is not harmful to always return true here, but the [View](#) will not perform optimally.

Implements [KDGantt::AbstractRowController](#).

Definition at line 66 of file `kdgantttreeviewrowcontroller.cpp`.

References `d`.

```

67 {
68     //qDebug() << _idx.model()<<d->proxy << d->treeview->model();
69     const QModelIndex idx = d->proxy->mapToSource( _idx );
70     assert( idx.isValid() ? ( idx.model() == d->treeview->model() ):( true ) );
71     return d->treeview->visualRect( idx ).isValid();
72 }

```

#### 12.25.3.6 `int TreeViewRowController::maximumItemHeight () const` [virtual]

Implements [KDGantt::AbstractRowController](#).

Definition at line 61 of file `kdgantttreeviewrowcontroller.cpp`.

References `d`.

```

62 {
63     return d->treeview->fontMetrics().height();
64 }

```

### 12.25.3.7 [Span](#) TreeViewRowController::rowGeometry (const QModelIndex & *idx*) const [virtual]

#### Returns:

A [Span](#) consisting of the row offset and height for the row containing *idx*. A simple implementation might look like

```
Span MyRowCtrlr::rowGeometry(const QModelIndex& idx)
{
    return Span(idx.row()*10,10);
}
```

Implements [KDGantt::AbstractRowController](#).

Definition at line 74 of file `kdgantttreeviewrowcontroller.cpp`.

References `d`, and `r`.

```
75 {
76     const QModelIndex idx = d->proxy->mapToSource( _idx );
77     assert( idx.isValid() ? ( idx.model() == d->treeview->model() ):( true ) );
78     QRect r = d->treeview->visualRect(idx).translated( QPoint( 0, d->treeview->verticalOffset() ) );
79     return Span( r.y(), r.height() );
80 }
```

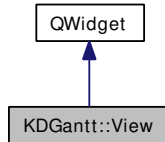
The documentation for this class was generated from the following files:

- [kdgantttreeviewrowcontroller.h](#)
- [kdgantttreeviewrowcontroller.cpp](#)

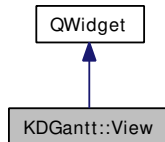
## 12.26 KDGantt::View Class Reference

```
#include <KDGanttView>
```

Inheritance diagram for KDGantt::View:



Collaboration diagram for KDGantt::View:



### 12.26.1 Detailed Description

This widget that consists of a QTreeView and a [GraphicsView](#).

This is the easy to use, complete gantt chart widget. It consists of a QTreeView on the left and a [KDGantt::GraphicsView](#) on the right separated by a QSplitter. The two views share the same model.

Definition at line 45 of file kdganttview.h.

#### Public Slots

- void [setConstraintModel](#) ([ConstraintModel](#) \*)
- void [setGrid](#) ([AbstractGrid](#) \*)
- void [setItemDelegate](#) ([ItemDelegate](#) \*)
- void [setModel](#) ([QAbstractItemModel](#) \*model)
- void [setRootIndex](#) (const [QModelIndex](#) &idx)
- void [setSelectionModel](#) ([QItemSelectionModel](#) \*smodel)

#### Public Member Functions

- [ConstraintModel](#) \* [constraintModel](#) () const
- [QAbstractProxyModel](#) \* [ganttProxyModel](#) ()
- const [QAbstractProxyModel](#) \* [ganttProxyModel](#) () const
- [GraphicsView](#) \* [graphicsView](#) ()
- const [GraphicsView](#) \* [graphicsView](#) () const
- [AbstractGrid](#) \* [grid](#) () const
- [QModelIndex](#) [indexAt](#) (const [QPoint](#) &pos) const
- [ItemDelegate](#) \* [itemDelegate](#) () const
- [QAbstractItemView](#) \* [leftView](#) ()

- const [QAbstractItemView](#) \* [leftView](#) () const
- [QAbstractItemModel](#) \* [model](#) () const
- void [print](#) ([QPainter](#) \*painter, const [QRectF](#) &target=[QRectF](#)(), bool drawRowLabels=true)
- [QModelIndex](#) [rootIndex](#) () const
- const [AbstractRowController](#) \* [rowController](#) () const
- [AbstractRowController](#) \* [rowController](#) ()
- [QItemSelectionModel](#) \* [selectionModel](#) () const
- void [setLeftView](#) ([QAbstractItemView](#) \*)
- void [setRowController](#) ([AbstractRowController](#) \*)
- [View](#) ([QWidget](#) \*parent=0)
- virtual [~View](#) ()

## Protected Member Functions

- void [resizeEvent](#) ([QResizeEvent](#) \*)

## 12.26.2 Constructor & Destructor Documentation

### 12.26.2.1 View::View ([QWidget](#) \*parent = 0) [explicit]

Constructor. Creates a [View](#) with parent *parent*, a [DateTimeGrid](#) as default grid implementation and no model etc.

Definition at line 190 of file `kdganttview.cpp`.

```

191     : QWidget (parent),
192       _d(new Private(this))
193 {
194     #if defined KDAB_EVAL
195         EvalDialog::checkEvalLicense( "KD Gantt" );
196     #endif
197     _d->init();
198 }
```

### 12.26.2.2 View::~View () [virtual]

Definition at line 200 of file `kdganttview.cpp`.

```

201 {
202     delete _d;
203 }
```

## 12.26.3 Member Function Documentation

### 12.26.3.1 [ConstraintModel](#) \* View::constraintModel () const

#### Returns:

the [KDGantt::ConstraintModel](#) displayed by this view.

Definition at line 396 of file `kdganttview.cpp`.

References [d](#).

```
397 {
398     return d->constraintProxy.sourceModel();
399 }
```

### 12.26.3.2 [QAbstractProxyModel](#) \* [View::ganttProxyModel](#) ()

Definition at line 406 of file kdganttview.cpp.

References d.

```
407 {
408     return &(amp;d->ganttProxyModel );
409 }
```

### 12.26.3.3 [const QAbstractProxyModel](#) \* [View::ganttProxyModel](#) () [const](#)

Definition at line 401 of file kdganttview.cpp.

References d.

Referenced by `KDAB_SCOPED_UNITTEST_SIMPLE()`.

```
402 {
403     return &(amp;d->ganttProxyModel );
404 }
```

### 12.26.3.4 [GraphicsView](#) \* [View::graphicsView](#) ()

Definition at line 296 of file kdganttview.cpp.

References d.

```
297 {
298     return &d->gfxview;
299 }
```

### 12.26.3.5 [const GraphicsView](#) \* [KDGantt::View::graphicsView](#) () [const](#)

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

Definition at line 288 of file kdganttview.cpp.

References d.

```
289 {
290     return &d->gfxview;
291 }
```

### 12.26.3.6 [AbstractGrid](#) \* [View::grid](#) () const

**Returns:**

the [AbstractGrid](#) used by this view.

Definition at line 348 of file `kdganttview.cpp`.

References [d](#).

Referenced by `setGrid()`.

```
349 {  
350     return d->gfxview.grid();  
351 }
```

### 12.26.3.7 [QModelIndex](#) [View::indexAt](#) (const [QPoint](#) & *pos*) const

**Returns:**

The [QModelIndex](#) for the item located at position *pos* in the view or an invalid index if no item was present at that position.

**See also:**

[GraphicsView::indexAt](#)

Definition at line 422 of file `kdganttview.cpp`.

References [d](#).

```
423 {  
424     return d->gfxview.indexAt ( pos );  
425 }
```

### 12.26.3.8 [ItemDelegate](#) \* [View::itemDelegate](#) () const

**Returns:**

the [ItemDelegate](#) used by this view to render items

Definition at line 371 of file `kdganttview.cpp`.

References [d](#).

```
372 {  
373     return d->gfxview.itemDelegate();  
374 }
```

### 12.26.3.9 [QAbstractItemView](#) \* [View::leftView](#) ()

Definition at line 280 of file `kdganttview.cpp`.

References [d](#).

```
281 {  
282     return d->leftWidget;  
283 }
```

### 12.26.3.10 `const QAbstractItemView * KDGantt::View::leftView () const`

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

Definition at line 272 of file `kdganttview.cpp`.

References `d`.

Referenced by `model()`, `rootIndex()`, `selectionModel()`, `setItemDelegate()`, `setModel()`, `setRootIndex()`, and `setSelectionModel()`.

```
273 {
274     return d->leftWidget;
275 }
```

### 12.26.3.11 `QAbstractItemModel * View::model () const`

#### Returns:

the current model displayed by this view

Definition at line 303 of file `kdganttview.cpp`.

References `leftView()`.

```
304 {
305     return leftView()->model();
306 }
```

### 12.26.3.12 `void View::print (QPainter * painter, const QRectF & target = QRectF(), bool drawRowLabels = true)`

Render the GanttView inside the rectangle *target* using the painter *painter*.

#### See also:

[KDGantt::GraphicsView::print\(QPainter\\* painter, const QRectF& target,bool drawRowLabels\)](#)

Definition at line 430 of file `kdganttview.cpp`.

References `d`.

```
431 {
432 #if 0
433     QRectF targetRect = target;
434     if( targetRect.isNull() )
435         targetRect.setRect(0, 0,
436                             painter->device()->width(),
437                             painter->device()->height());
438
439     QPixmap leftw = QPixmap::grabWidget(d->leftWidget->viewport());
440
441     /* Now figure out the size of the whole thing to fit it into
442      * targetRect */
443     QRectF targetLeft( leftw.rect() );
444     QRectF targetRight( d->gfxview.scene()->sceneRect() );
445     qreal sw = (targetLeft.width()+targetRight.width())/targetRect.width();
```

```

446     qreal sh = (targetLeft.height()+targetRight.height())/targetRect.height();
447     if( sw > 1. || sh > 1. ) {
448         // we need to scale
449         qreal s = qMax(sw,sh);
450         targetLeft.setWidth(targetLeft.width()/s);
451         targetLeft.setHeight(targetLeft.height()/s);
452         targetRight.setWidth(targetRight.width()/s);
453         targetRight.setHeight(targetRight.height()/s);
454     }
455     targetRight.translate( targetLeft.width(),0 );
456
457     painter->drawPixmap( targetLeft, leftw, leftw.rect() );
458     d->gfxview.scene()->render( painter,
459                               targetRight);
460 #endif
461     d->gfxview.print( painter,
462                     target,
463                     drawRowLabels);
464 }

```

### 12.26.3.13 void View::resizeEvent (QResizeEvent \*) [protected]

Definition at line 411 of file kdganttview.cpp.

```

412 {
413     QWidget::resizeEvent(ev);
414 }

```

### 12.26.3.14 QModelIndex View::rootIndex () const

#### Returns:

the rootindex for this view.

Definition at line 355 of file kdganttview.cpp.

References [leftView\(\)](#).

```

356 {
357     return leftView()->rootIndex();
358 }

```

### 12.26.3.15 const AbstractRowController \* View::rowController () const

Definition at line 263 of file kdganttview.cpp.

References [d](#).

```

264 {
265     return d->rowController;
266 }

```

**12.26.3.16** [AbstractRowController](#) \* [KDGantt::View::rowController](#) ()

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

Definition at line 256 of file `kdganttview.cpp`.

References `d`.

```
257 {
258     return d->rowController;
259 }
```

**12.26.3.17** [QItemSelectionModel](#) \* [View::selectionModel](#) () const

**Returns:**

the `QItemSelectionModel` used by this view

Definition at line 322 of file `kdganttview.cpp`.

References `leftView()`.

```
323 {
324     return leftView()->selectionModel();
325 }
```

**12.26.3.18** void [View::setConstraintModel](#) ([ConstraintModel](#) \* *cm*) [slot]

Sets the constraintmodel displayed by this view.

**See also:**

[KDGantt::ConstraintModel](#).

Definition at line 388 of file `kdganttview.cpp`.

References `d`.

```
389 {
390     d->constraintProxy.setSourceModel( cm );
391     d->gfxview.setConstraintModel( &d->mappedConstraintModel );
392 }
```

**12.26.3.19** void [View::setGrid](#) ([AbstractGrid](#) \* *grid*) [slot]

Sets the [AbstractGrid](#) for this view. The grid is an object that controls how `QModelIndexes` are mapped to and from the view and how the background and header is rendered.

**See also:**

[AbstractGrid](#) and [DateTimeGrid](#).

Definition at line 341 of file `kdganttview.cpp`.

References `d`, and `grid()`.

```

342 {
343     d->gfxview.setGrid( grid );
344 }

```

### 12.26.3.20 void View::setItemDelegate (ItemDelegate \* delegate) [slot]

Sets the [KDGantt::ItemDelegate](#) used for rendering items on this view.

#### See also:

[ItemDelegate](#) and [QAbstractItemDelegate](#).

Definition at line 379 of file `kdganttview.cpp`.

References `d`, and `leftView()`.

```

380 {
381     leftView()->setItemDelegate( delegate );
382     d->gfxview.setItemDelegate( delegate );
383 }

```

### 12.26.3.21 void View::setLeftView (QAbstractItemView \* aiv)

#### Parameters:

*aiv* The view to be used to the left, instead of the default tree view

#### See also:

[setRowController\(\)](#)

Definition at line 210 of file `kdganttview.cpp`.

References `d`.

Referenced by `KDAB_SCOPED_UNITTEST_SIMPLE()`.

```

211 {
212     assert( aiv );
213     if ( aiv==d->leftWidget ) return;
214     if ( !d->leftWidget.isNull() ) {
215         d->leftWidget->disconnect( this );
216         d->leftWidget->hide();
217         d->leftWidget->verticalScrollBar()->disconnect( d->gfxview.verticalScrollBar() );
218         d->gfxview.verticalScrollBar()->disconnect( d->leftWidget->verticalScrollBar() );
219     }
220
221     d->leftWidget = aiv;
222     d->splitter.insertWidget( 0, d->leftWidget );
223
224     if( qobject_cast<QTreeView*>(d->leftWidget) ) {
225         connect( d->leftWidget, SIGNAL( collapsed( const QModelIndex& ) ),
226                this, SLOT( slotCollapsed( const QModelIndex& ) ) );
227         connect( d->leftWidget, SIGNAL( expanded( const QModelIndex& ) ),
228                this, SLOT( slotExpanded( const QModelIndex& ) ) );
229     }
230
231     connect( d->gfxview.verticalScrollBar(), SIGNAL( valueChanged( int ) ),
232            d->leftWidget->verticalScrollBar(), SLOT( setValue( int ) ) );

```

```

233     connect( d->leftWidget->verticalScrollBar(), SIGNAL( valueChanged( int ) ),
234             d->gfxview.verticalScrollBar(), SLOT( setValue( int ) ) );
235     connect( d->leftWidget->verticalScrollBar(), SIGNAL( rangeChanged( int, int ) ),
236             this, SLOT( slotLeftWidgetVerticalRangeChanged( int, int ) ) );
237     connect( d->gfxview.verticalScrollBar(), SIGNAL( rangeChanged( int, int ) ),
238             this, SLOT( slotGfxViewVerticalRangeChanged( int, int ) ) );
239 }

```

### 12.26.3.22 void View::setModel (QAbstractItemModel \* model) [slot]

Sets the QAbstractItemModel to be displayed in this view to *model*.

See also:

[GraphicsView::setModel](#)

Definition at line 313 of file kdganttview.cpp.

References [d](#), and [leftView\(\)](#).

```

314 {
315     leftView()->setModel( model );
316     d->gantttProxyModel.setSourceModel( model );
317     d->gfxview.setModel( &d->gantttProxyModel );
318 }

```

### 12.26.3.23 void View::setRootIndex (const QModelIndex & idx) [slot]

Sets the root index of the model displayed by this view. Similar to QAbstractItemView::setRootIndex, default is QModelIndex().

Definition at line 363 of file kdganttview.cpp.

References [d](#), and [leftView\(\)](#).

```

364 {
365     leftView()->setRootIndex( idx );
366     d->gfxview.setRootIndex( idx );
367 }

```

### 12.26.3.24 void View::setRowController (AbstractRowController \* ctrl)

Sets *ctrl* to be the rowcontroller used by this [View](#). The default rowcontroller is owned by [KDGantt::View](#) and is suitable for the default treeview in the left part of the view. You probably only want to change this if you replace the treeview.

Definition at line 246 of file kdganttview.cpp.

References [d](#).

Referenced by [KDAB\\_SCOPED\\_UNITTEST\\_SIMPLE\(\)](#).

```

247 {
248     if ( ctrl == d->rowController ) return;
249     d->rowController = ctrl;
250     d->gfxview.setRowController( d->rowController );
251 }

```

**12.26.3.25 void View::setSelectionModel (QItemSelectionModel \* *smodel*)** [slot]

Sets the QItemSelectionModel used by this view to manage selections. Similar to QAbstractItemView::setSelectionModel

Definition at line 330 of file kdganttview.cpp.

References `d`, and `leftView()`.

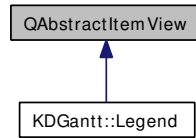
```
331 {  
332     leftView()->setSelectionModel( smodel );  
333     d->gfxview.setSelectionModel( new QItemSelectionModel( &(amp; d->gantttProxyModel ),this ) );  
334 }
```

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

- [kdganttview.h](#)
- [kdganttview.cpp](#)

## 12.27 QAbstractItemView Class Reference

Inheritance diagram for QAbstractItemView:

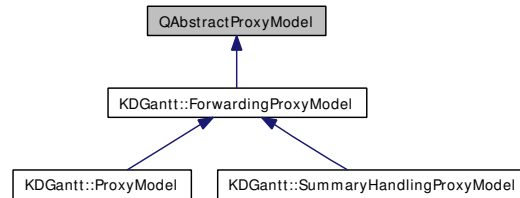


The documentation for this class was generated from the following file:

- [kdganttlegend.h](#)

## 12.28 QAbstractProxyModel Class Reference

Inheritance diagram for QAbstractProxyModel:

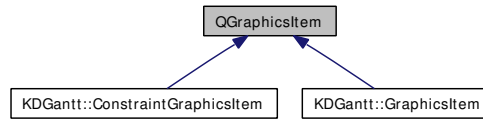


The documentation for this class was generated from the following file:

- [kdganttfowardngproxymodel.h](#)

## 12.29 QGraphicsItem Class Reference

Inheritance diagram for QGraphicsItem:

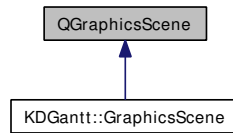


The documentation for this class was generated from the following file:

- [kdganttgraphicsitem.h](#)

## 12.30 QGraphicsScene Class Reference

Inheritance diagram for QGraphicsScene:

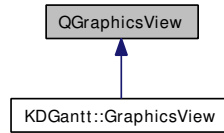


The documentation for this class was generated from the following file:

- [kdganttgraphicsscene.h](#)

## 12.31 QGraphicsView Class Reference

Inheritance diagram for QGraphicsView:

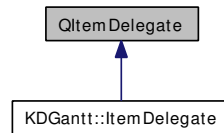


The documentation for this class was generated from the following file:

- [kdganttgraphicsview.h](#)

## 12.32 QItemDelegate Class Reference

Inheritance diagram for QItemDelegate:

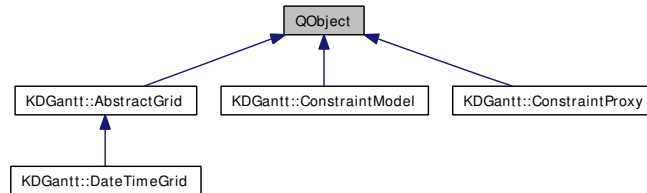


The documentation for this class was generated from the following file:

- [kdganttitemdelegate.h](#)

## 12.33 QObject Class Reference

Inheritance diagram for QObject:

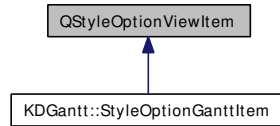


The documentation for this class was generated from the following file:

- [kdganttconstraintmodel.h](#)

## 12.34 QStyleOptionViewItem Class Reference

Inheritance diagram for QStyleOptionViewItem:

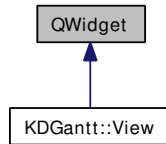


The documentation for this class was generated from the following file:

- [kdganttstyleoptionganttitem.h](#)

## 12.35 QWidget Class Reference

Inheritance diagram for QWidget:



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

- [kdgantview.h](#)

## Chapter 13

# KD Gantt 2 File Documentation

### 13.1 docs.h File Reference

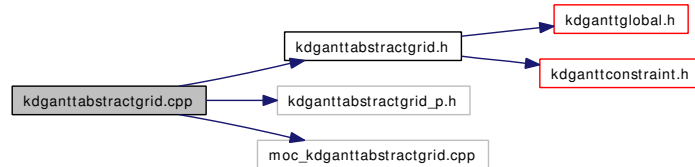
#### Namespaces

- namespace [KDGantt](#)

## 13.2 kdganttabstractgrid.cpp File Reference

```
#include "kdganttabstractgrid.h"
#include "kdganttabstractgrid_p.h"
#include "moc_kdganttabstractgrid.cpp"
```

Include dependency graph for kdganttabstractgrid.cpp:



### Defines

- #define `d_d_func()`

### 13.2.1 Define Documentation

#### 13.2.1.1 #define `d_d_func()`

Definition at line 53 of file kdganttabstractgrid.cpp.

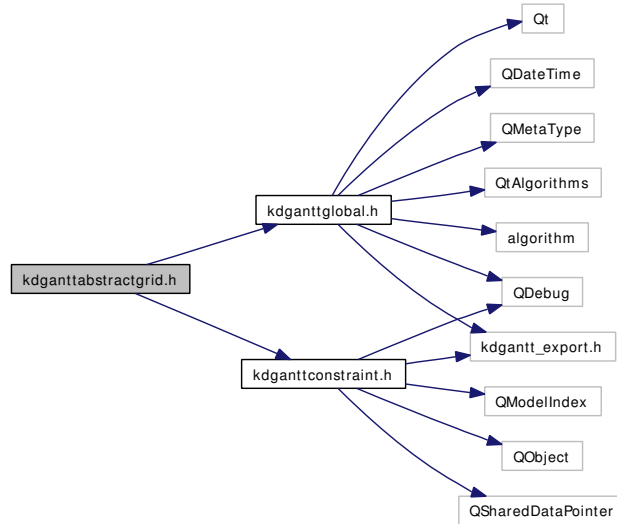
Referenced by `KDGantt::ConstraintModel::addConstraint()`, `KDGantt::ConstraintModel::cleanup()`, `KDGantt::GraphicsView::clearItems()`, `KDGantt::GraphicsScene::clearItems()`, `KDGantt::ProxyModel::column()`, `KDGantt::View::constraintModel()`, `KDGantt::GraphicsView::constraintModel()`, `KDGantt::GraphicsScene::constraintModel()`, `KDGantt::ConstraintModel::constraints()`, `KDGantt::ConstraintModel::constraintsForIndex()`, `KDGantt::SummaryHandlingProxyModel::data()`, `KDGantt::ProxyModel::data()`, `KDGantt::DateTimeGrid::dayWidth()`, `KDGantt::ItemDelegate::defaultBrush()`, `KDGantt::ItemDelegate::defaultPen()`, `KDGantt::GraphicsView::deleteSubtree()`, `KDGantt::GraphicsScene::dragSource()`, `KDGantt::GraphicsScene::drawBackground()`, `KDGantt::Legend::drawItem()`, `KDGantt::GraphicsScene::findConstraintItem()`, `KDGantt::GraphicsScene::findItem()`, `KDGantt::SummaryHandlingProxyModel::flags()`, `KDGantt::DateTimeGrid::freeDays()`, `KDGantt::View::gantProxyModel()`, `KDGantt::Legend::getStyleOption()`, `KDGantt::View::graphicsView()`, `KDGantt::View::grid()`, `KDGantt::GraphicsView::grid()`, `KDGantt::GraphicsScene::grid()`, `KDGantt::ConstraintModel::hasConstraint()`, `KDGantt::TreeViewRowController::headerHeight()`, `KDGantt::ListViewRowController::headerHeight()`, `KDGantt::TreeViewRowController::indexAbove()`, `KDGantt::ListViewRowController::indexAbove()`, `KDGantt::View::indexAt()`, `KDGantt::TreeViewRowController::indexAt()`, `KDGantt::ListViewRowController::indexAt()`, `KDGantt::GraphicsView::indexAt()`, `KDGantt::TreeViewRowController::indexBelow()`, `KDGantt::ListViewRowController::indexBelow()`, `KDGantt::GraphicsScene::insertItem()`, `KDGantt::GraphicsView::isReadOnly()`, `KDGantt::GraphicsScene::isReadOnly()`, `KDGantt::TreeViewRowController::isRowVisible()`, `KDGantt::ListViewRowController::isRowVisible()`, `KDGantt::View::itemDelegate()`, `KDGantt::GraphicsView::itemDelegate()`, `KDGantt::GraphicsScene::itemDelegate()`, `KDGantt::View::leftView()`, `KDGantt::DateTimeGrid::mapFromChart()`, `KDGantt::DateTimeGrid::mapToChart()`, `KDGantt::TreeViewRowController::maximumItemHeight()`, `KDGantt::ListViewRowController::maximumItemHeight()`, `KDGantt::Legend::measureItem()`, `KDGantt::GraphicsView::model()`, `KDGantt::GraphicsScene::model()`, `KDGantt::AbstractGrid::model()`, `KDGantt::DateTimeGrid::paintDayScaleHeader()`, `KDGantt::DateTimeGrid::paintGrid()`, `KDGantt::Date-`

TimeGrid::paintHourScaleHeader(), KDGantt::View::print(), KDGantt::GraphicsView::print(),  
 KDGantt::GraphicsScene::print(), KDGantt::ConstraintModel::removeConstraint(), KDGantt::Graphics-  
 Scene::removeItem(), KDGantt::GraphicsView::resizeEvent(), KDGantt::ProxyModel::role(),  
 KDGantt::GraphicsView::rootIndex(), KDGantt::GraphicsScene::rootIndex(), KDGantt::Abstract-  
 Grid::rootIndex(), KDGantt::View::rowController(), KDGantt::GraphicsView::rowController(),  
 KDGantt::GraphicsScene::rowController(), KDGantt::TreeViewRowController::rowGeometry(),  
 KDGantt::ListViewRowController::rowGeometry(), KDGantt::DateTimeGrid::rowSeparators(),  
 KDGantt::DateTimeGrid::scale(), KDGantt::GraphicsView::selectionModel(), KDGantt::Graphics-  
 Scene::selectionModel(), KDGantt::ProxyModel::setColumn(), KDGantt::View::setConstraint-  
 Model(), KDGantt::GraphicsView::setConstraintModel(), KDGantt::GraphicsScene::setConstraint-  
 Model(), KDGantt::SummaryHandlingProxyModel::setData(), KDGantt::ProxyModel::setData(),  
 KDGantt::DateTimeGrid::setDayWidth(), KDGantt::ItemDelegate::setDefaultBrush(), KDGantt::Item-  
 Delegate::setDefaultPen(), KDGantt::GraphicsScene::setDragSource(), KDGantt::DateTimeGrid::set-  
 FreeDays(), KDGantt::View::setGrid(), KDGantt::GraphicsView::setGrid(), KDGantt::Graphics-  
 Scene::setGrid(), KDGantt::View::setItemDelegate(), KDGantt::GraphicsView::setItemDelegate(),  
 KDGantt::GraphicsScene::setItemDelegate(), KDGantt::View::setLeftView(), KDGantt::View::set-  
 Model(), KDGantt::Legend::setModel(), KDGantt::GraphicsView::setModel(), KDGantt::Graphics-  
 Scene::setModel(), KDGantt::AbstractGrid::setModel(), KDGantt::GraphicsView::setReadOnly(),  
 KDGantt::GraphicsScene::setReadOnly(), KDGantt::ProxyModel::setRole(), KDGantt::View::set-  
 RootIndex(), KDGantt::GraphicsView::setRootIndex(), KDGantt::GraphicsScene::setRootIndex(),  
 KDGantt::AbstractGrid::setRootIndex(), KDGantt::View::setRowController(), KDGantt::Graphics-  
 View::setRowController(), KDGantt::GraphicsScene::setRowController(), KDGantt::Date-  
 Time-  
 Grid::setRowSeparators(), KDGantt::DateTimeGrid::setScale(), KDGantt::View::setSelectionModel(),  
 KDGantt::GraphicsView::setSelectionModel(), KDGantt::GraphicsScene::setSelectionModel(),  
 KDGantt::SummaryHandlingProxyModel::setSourceModel(), KDGantt::DateTimeGrid::setStart-  
 DateTime(), KDGantt::GraphicsView::setSummaryHandlingModel(), KDGantt::GraphicsScene::set-  
 SummaryHandlingModel(), KDGantt::DateTimeGrid::setWeekStart(), KDGantt::SummaryHandling-  
 ProxyModel::sourceColumnsAboutToBeInserted(), KDGantt::SummaryHandlingProxyModel::source-  
 ColumnsAboutToBeRemoved(), KDGantt::SummaryHandlingProxyModel::sourceDataChanged(),  
 KDGantt::SummaryHandlingProxyModel::sourceLayoutChanged(), KDGantt::SummaryHandling-  
 ProxyModel::sourceModelReset(), KDGantt::SummaryHandlingProxyModel::sourceRowsAbout-  
 ToBeInserted(), KDGantt::SummaryHandlingProxyModel::sourceRowsAboutToBeRemoved(),  
 KDGantt::DateTimeGrid::startDateTime(), KDGantt::GraphicsScene::summaryHandlingModel(),  
 KDGantt::GraphicsScene::updateItems(), KDGantt::GraphicsView::updateRow(), KDGantt::Graphics-  
 View::updateSceneRect(), and KDGantt::DateTimeGrid::weekStart().

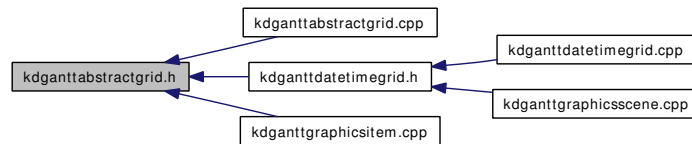
### 13.3 kdganttabstractgrid.h File Reference

```
#include "kdganttglobal.h"
#include "kdganttconstraint.h"
```

Include dependency graph for kdganttabstractgrid.h:



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



#### Namespaces

- namespace [KDGantt](#)

#### Classes

- class [KDGantt::AbstractGrid](#)

*Abstract baseclass for grids. A grid is used to convert between `QModelIndex`'es and gantt chart values (doubles) and to paint the background and header of the view.*

## 13.4 kdganttabstractrowcontroller.cpp File Reference

```
#include "kdganttabstractrowcontroller.h"
```

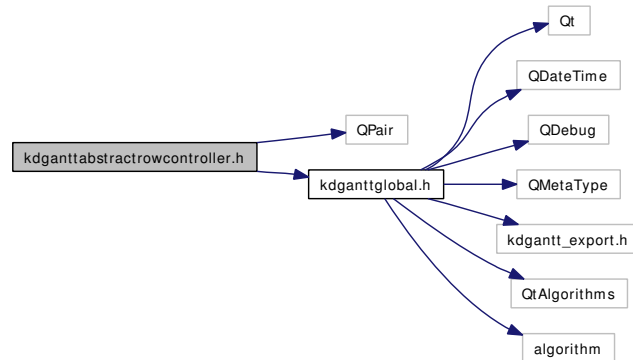
Include dependency graph for kdganttabstractrowcontroller.cpp:



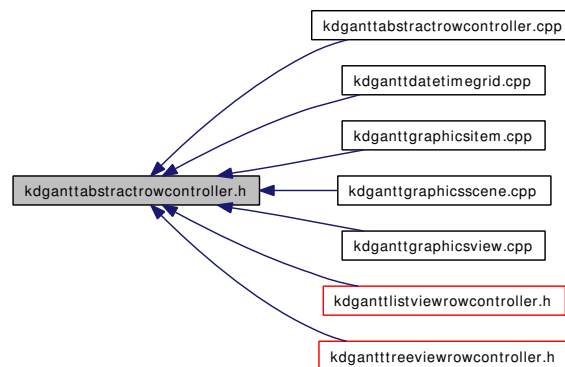
## 13.5 kdganttabstractrowcontroller.h File Reference

```
#include <QPair>
#include "kdganttglobal.h"
```

Include dependency graph for kdganttabstractrowcontroller.h:



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



### Namespaces

- namespace [KDGantt](#)

### Classes

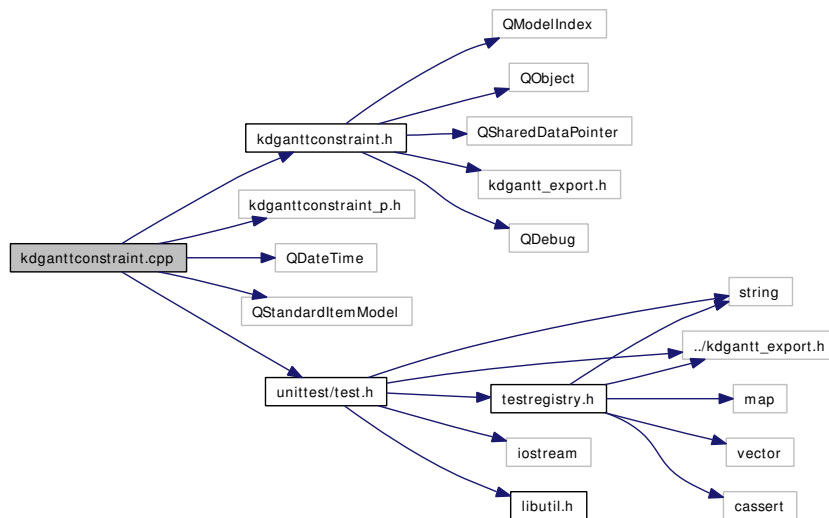
- class [KDGantt::AbstractRowController](#)

*Abstract baseclass for row controllers. A row controller is used by the [GraphicsView](#) to navigate the model and to determine the row geometries.*

## 13.6 kdganttconstraint.cpp File Reference

```
#include "kdganttconstraint.h"
#include "kdganttconstraint_p.h"
#include <QDateTime>
#include <QStandardItemModel>
#include "unittest/test.h"
```

Include dependency graph for kdganttconstraint.cpp:



### Functions

- [KDAB\\_SCOPED\\_UNITTEST\\_SIMPLE](#) (KDGantt, [Constraint](#), "test")
- [QDebug operator<<](#) (QDebug dbg, const [Constraint](#) &c)

### 13.6.1 Function Documentation

#### 13.6.1.1 KDAB\_SCOPED\_UNITTEST\_SIMPLE (KDGantt, [Constraint](#), "test")

Definition at line 173 of file kdganttconstraint.cpp.

References [assertEqual](#), [assertFalse](#), [assertNotEqual](#), [assertTrue](#), [KDGantt::qHash\(\)](#), [KDGantt::Constraint::type\(\)](#), and [KDGantt::Constraint::TypeSoft](#).

```

174 {
175     QStandardItemModel dummyModel( 100, 100 );
176     QModelIndex idx1 = dummyModel.index( 7, 17, QModelIndex() );
177     QModelIndex idx2 = dummyModel.index( 42, 17, QModelIndex() );
178
179     Constraint c1 = Constraint( QModelIndex(), QModelIndex(), Constraint::TypeSoft );
180     Constraint c2 = Constraint( QModelIndex(), QModelIndex(), Constraint::TypeSoft );
181     Constraint c3 = c2;
182     Constraint c4( idx1, idx2 );
183     Constraint c5( idx2, idx1 );

```

```
184
185     assertTrue( c1==c2 );
186     assertEquals( qHash( c1 ), qHash( c2 ) );
187     assertTrue( c1==c3 );
188     assertEquals( qHash( c1 ), qHash( c3 ) );
189     assertTrue( c2==c3 );
190     assertEquals( qHash( c2 ), qHash( c3 ) );
191
192     assertFalse( c2==c4 );
193     assertNotEqual( qHash( c2 ), qHash( c4 ) );
194
195     assertFalse( c4==c5 );
196
197     assertEquals( c3.type(), Constraint::TypeSoft );
198
199     dummyModel.removeRow( 8 );
200     assertFalse( c4==c5 );
201 }
```

### 13.6.1.2 QDebug operator<< (QDebug *dbg*, const **Constraint** & *c*)

Definition at line 154 of file `kdganttconstraint.cpp`.

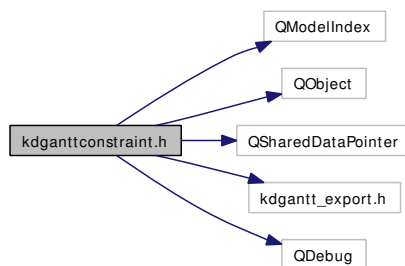
Referenced by operator<<().

```
155 {
156     return c.debug( dbg );
157 }
```

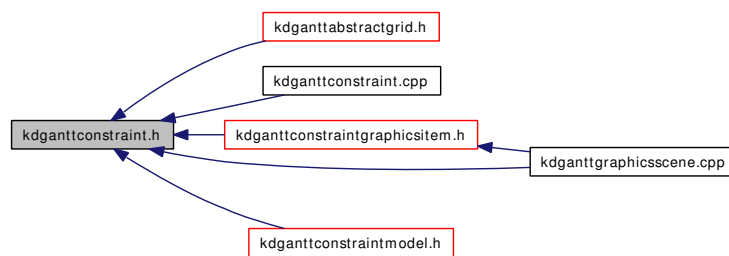
## 13.7 kdganttconstraint.h File Reference

```
#include <QModelIndex>
#include <QObject>
#include <QSharedPointer>
#include "kdgantt_export.h"
#include <QDebug>
```

Include dependency graph for kdganttconstraint.h:



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



### Namespaces

- namespace [KDGantt](#)

### Classes

- class [KDGantt::Constraint](#)  
*A class used to represent a dependency.*

### Functions

- `QDebug operator<< (QDebug dbg, const KDGantt::Constraint &c)`
- `uint KDGantt::qHash (const Constraint &c)`

## 13.7.1 Function Documentation

### 13.7.1.1 QDebug operator<< (QDebug *dbg*, const [KDGantt::Constraint](#) & *c*)

Definition at line 154 of file `kdganttconstraint.cpp`.

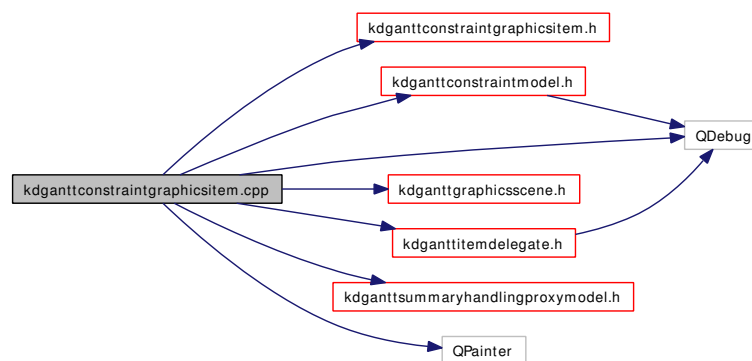
References `c`.

```
155 {  
156     return c.debug( dbg );  
157 }
```

## 13.8 kdganttconstraintgraphicsitem.cpp File Reference

```
#include "kdganttconstraintgraphicsitem.h"  
#include "kdganttconstraintmodel.h"  
#include "kdganttgraphicsscene.h"  
#include "kdganttitemdelegate.h"  
#include "kdganttsummaryhandlingproxymodel.h"  
#include <QPainter>  
#include <QDebug>
```

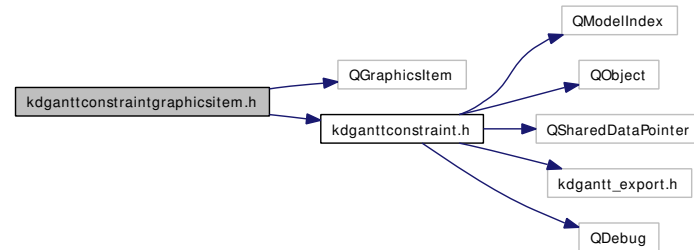
Include dependency graph for kdganttconstraintgraphicsitem.cpp:



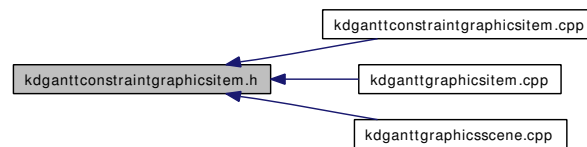
## 13.9 kdganttconstraintgraphicsitem.h File Reference

```
#include <QGraphicsItem>
#include "kdganttconstraint.h"
```

Include dependency graph for kdganttconstraintgraphicsitem.h:



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



### Namespaces

- namespace [KDGantt](#)

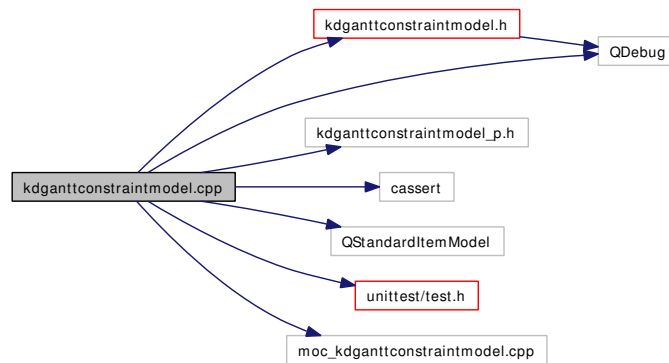
### Classes

- class [KDGantt::ConstraintGraphicsItem](#)

## 13.10 kdganttconstraintmodel.cpp File Reference

```
#include "kdganttconstraintmodel.h"
#include "kdganttconstraintmodel_p.h"
#include <QDebug>
#include <cassert>
#include <QStandardItemModel>
#include "unittest/test.h"
#include "moc_kdganttconstraintmodel.cpp"
```

Include dependency graph for kdganttconstraintmodel.cpp:



### Defines

- #define `d d_func()`

### Functions

- `KDAB_SCOPED_UNITTEST_SIMPLE` (`KDGantt`, `ConstraintModel`, "test")
- `std::ostream & operator<<` (`std::ostream &os`, `const QModelIndex &idx`)
- `QDebug operator<<` (`QDebug dbg`, `const KDGantt::ConstraintModel &model`)

#### 13.10.1 Define Documentation

##### 13.10.1.1 #define `d d_func()`

Definition at line 91 of file `kdganttconstraintmodel.cpp`.

#### 13.10.2 Function Documentation

##### 13.10.2.1 `KDAB_SCOPED_UNITTEST_SIMPLE` (`KDGantt`, `ConstraintModel`, "test")

Definition at line 235 of file `kdganttconstraintmodel.cpp`.

References `KDGantt::ConstraintModel::addConstraint()`, `assertEqual`, `assertFalse`, `assertTrue`, `KDGantt::ConstraintModel::constraints()`, `KDGantt::ConstraintModel::constraintsForIndex()`, `KDGantt::ConstraintModel::hasConstraint()`, and `KDGantt::ConstraintModel::removeConstraint()`.

```

236 {
237     QStandardItemModel dummyModel( 100, 100 );
238     ConstraintModel model;
239
240     QModelIndex invalidIndex;
241     assertEquals( invalidIndex, invalidIndex );
242
243     assertEquals( model.constraints().count(), 0 );
244
245     model.addConstraint( Constraint( QModelIndex(), QModelIndex() ) );
246     assertEquals( model.constraints().count(), 1 );
247
248     model.addConstraint( Constraint( QModelIndex(), QModelIndex() ) );
249     assertEquals( model.constraints().count(), 1 );
250
251     QPersistentModelIndex idx1 = dummyModel.index( 7, 17, QModelIndex() );
252     QPersistentModelIndex idx2 = dummyModel.index( 42, 17, QModelIndex() );
253
254     model.addConstraint( Constraint( idx1, idx2 ) );
255     assertEquals( model.constraints().count(), 2 );
256     assertTrue( model.hasConstraint( Constraint( idx1, idx2 ) ) );
257
258     assertEquals( model.constraintsForIndex( QModelIndex() ).count(), 1 );
259
260     assertEquals( model.constraints().count(), 2 );
261     model.removeConstraint( Constraint( QModelIndex(), QModelIndex() ) );
262     assertEquals( model.constraints().count(), 1 );
263     assertFalse( model.hasConstraint( Constraint( QModelIndex(), QModelIndex() ) ) );
264
265     model.removeConstraint( Constraint( QModelIndex(), QModelIndex() ) );
266     assertEquals( model.constraints().count(), 1 );
267
268     model.removeConstraint( Constraint( idx1, idx2 ) );
269     assertEquals( model.constraints().count(), 0 );
270     assertFalse( model.hasConstraint( Constraint( idx1, idx2 ) ) );
271
272     model.addConstraint( Constraint( idx1, idx2 ) );
273     assertTrue( model.hasConstraint( Constraint( idx1, idx2 ) ) );
274     dummyModel.removeRow( 8 );
275     assertTrue( model.hasConstraint( Constraint( idx1, idx2 ) ) );
276     dummyModel.removeRow( 7 );
277     assertTrue( model.hasConstraint( Constraint( idx1, idx2 ) ) );
278 }

```

### 13.10.2.2 `std::ostream& operator<< (std::ostream & os, const QModelIndex & idx)`

Definition at line 227 of file `kdganttconstraintmodel.cpp`.

```

228 {
229     QString str;
230     QDebug( &str )<<idx;
231     os<<str.toStdString();
232     return os;
233 }

```

### 13.10.2.3 `QDebug operator<< (QDebug dbg, const KDGantt::ConstraintModel & model)`

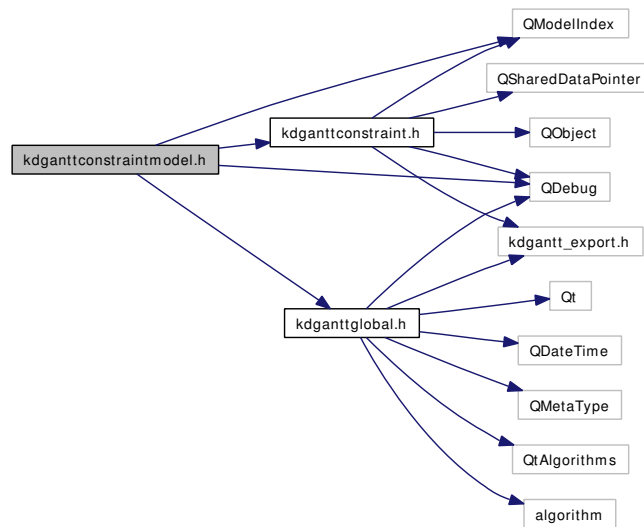
Definition at line 210 of file `kdganttconstraintmodel.cpp`.

```
211 {
212     dbg << "KDGantt::ConstraintModel[ " << static_cast<const QObject*>( &model ) << ":"
213         << model.constraints() << " ]";
214     return dbg;
215 }
```

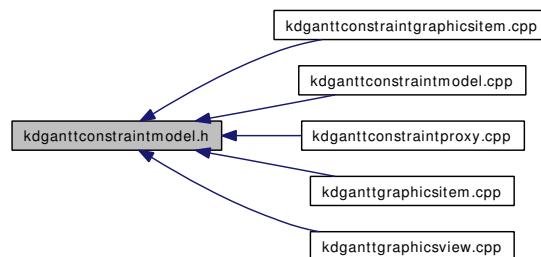
## 13.11 kdganttconstraintmodel.h File Reference

```
#include <QModelIndex>
#include <QDebug>
#include "kdganttglobal.h"
#include "kdganttconstraint.h"
```

Include dependency graph for kdganttconstraintmodel.h:



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



### Namespaces

- namespace [KDGantt](#)

### Classes

- class [KDGantt::ConstraintModel](#)

## Functions

- QDebug `operator<<` (QDebug dbg, [KDGantt::ConstraintModel](#) \*model)
- QDebug `operator<<` (QDebug dbg, const [KDGantt::ConstraintModel](#) &model)

### 13.11.1 Function Documentation

#### 13.11.1.1 QDebug `operator<<` (QDebug *dbg*, [KDGantt::ConstraintModel](#) \* *model*)

Definition at line 75 of file `kdganttconstraintmodel.h`.

References `operator<<()`.

```
76 {  
77     return operator<<(dbg, *model);  
78 }
```

#### 13.11.1.2 QDebug `operator<<` (QDebug *dbg*, const [KDGantt::ConstraintModel](#) & *model*)

Definition at line 210 of file `kdganttconstraintmodel.cpp`.

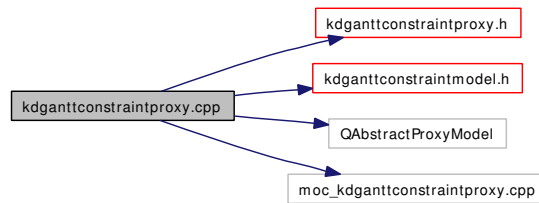
References `KDGantt::ConstraintModel::constraints()`.

```
211 {  
212     dbg << "KDGantt::ConstraintModel[ " << static_cast<const QObject*>( &model ) << " :"  
213         << model.constraints() << " ]";  
214     return dbg;  
215 }
```

## 13.12 kdganttconstraintproxy.cpp File Reference

```
#include "kdganttconstraintproxy.h"  
#include "kdganttconstraintmodel.h"  
#include <QAbstractProxyModel>  
#include "moc_kdganttconstraintproxy.cpp"
```

Include dependency graph for kdganttconstraintproxy.cpp:

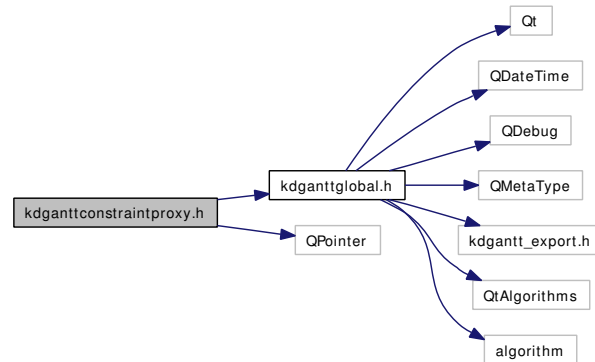


## 13.13 kdganttconstraintproxy.h File Reference

```
#include "kdganttglobal.h"
```

```
#include <QPointer>
```

Include dependency graph for kdganttconstraintproxy.h:



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



### Namespaces

- namespace [KDGantt](#)

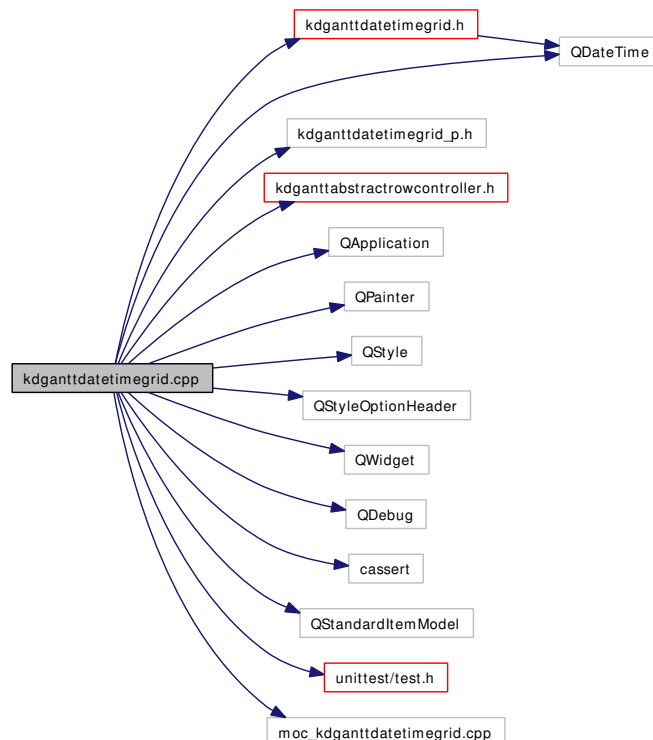
### Classes

- class [KDGantt::ConstraintProxy](#)

## 13.14 kdganttdateimegrid.cpp File Reference

```
#include "kdganttdateimegrid.h"
#include "kdganttdateimegrid_p.h"
#include "kdganttabstractrowcontroller.h"
#include <QApplication>
#include <QDateTime>
#include <QPainter>
#include <QStyle>
#include <QStyleOptionHeader>
#include <QWidget>
#include <QDebug>
#include <cassert>
#include <QStandardItemModel>
#include "unittest/test.h"
#include "moc_kdganttdateimegrid.cpp"
```

Include dependency graph for kdganttdateimegrid.cpp:



### Defines

- #define `d d_func()`

## Functions

- static void [debug\\_print\\_idx](#) (const QModelIndex &idx)
- [KDAB\\_SCOPED\\_UNITTEST\\_SIMPLE](#) (KDGantt, [DateTimeGrid](#), "test")
- std::ostream & [operator<<](#) (std::ostream &os, const QDateTime &dt)

### 13.14.1 Define Documentation

#### 13.14.1.1 #define d\_func()

Definition at line 76 of file kdganttdatetimegrid.cpp.

### 13.14.2 Function Documentation

#### 13.14.2.1 static void debug\_print\_idx (const QModelIndex & *idx*) [static]

Definition at line 228 of file kdganttdatetimegrid.cpp.

References [KDGantt::StartTimeRole](#).

```

229 {
230     if ( !idx.isValid() ) {
231         qDebug() << "[Invalid]";
232         return;
233     }
234     QDateTime st = idx.data( StartTimeRole ).toDateTime();
235     QDateTime et = idx.data( StartTimeRole ).toDateTime();
236     qDebug() << idx << "["<<st<<et<<"]";
237 }
```

#### 13.14.2.2 KDAB\_SCOPED\_UNITTEST\_SIMPLE (KDGantt, [DateTimeGrid](#), "test")

Definition at line 426 of file kdganttdatetimegrid.cpp.

References [assertEqual](#), [assertFalse](#), [assertTrue](#), [KDGantt::EndTimeRole](#), [KDGantt::AbstractGrid::isSatisfiedConstraint\(\)](#), [KDGantt::Span::length\(\)](#), [KDGantt::DateTimeGrid::mapFromChart\(\)](#), [KDGantt::DateTimeGrid::mapToChart\(\)](#), [KDGantt::Span::setEnd\(\)](#), [KDGantt::AbstractGrid::setModel\(\)](#), [KDGantt::DateTimeGrid::setStartDateTime\(\)](#), [KDGantt::Span::start\(\)](#), and [KDGantt::StartTimeRole](#).

```

426                                     {
427     QStandardItemModel model( 3, 2 );
428     DateTimeGrid grid;
429     QDateTime dt = QDateTime::currentDateTime();
430     grid.setModel( &model );
431     grid.setStartDateTime( dt.addDays( -10 ) );
432
433     model.setData( model.index( 0, 0 ), dt,                               StartTimeRole );
434     model.setData( model.index( 0, 0 ), dt.addDays( 17 ), EndTimeRole );
435
436     model.setData( model.index( 2, 0 ), dt.addDays( 18 ), StartTimeRole );
437     model.setData( model.index( 2, 0 ), dt.addDays( 19 ), EndTimeRole );
438
439     Span s = grid.mapToChart( model.index( 0, 0 ) );
440     //qDebug() << "span="<<s;
441
442     assertTrue( s.start()>0 );
```

```

443     assertTrue( s.length()>0 );
444
445     grid.mapFromChart( s, model.index( 1, 0 ) );
446
447     QDateTime s1 = model.data( model.index( 0, 0 ), StartTimeRole ).toDate();
448     QDateTime e1 = model.data( model.index( 0, 0 ), EndTimeRole ).toDate();
449     QDateTime s2 = model.data( model.index( 1, 0 ), StartTimeRole ).toDate();
450     QDateTime e2 = model.data( model.index( 1, 0 ), EndTimeRole ).toDate();
451
452     assertTrue( s1.isValid() );
453     assertTrue( e1.isValid() );
454     assertTrue( s2.isValid() );
455     assertTrue( e2.isValid() );
456
457     assertEquals( s1, s2 );
458     assertEquals( e1, e2 );
459
460     assertTrue( grid.isSatisfiedConstraint( Constraint( model.index( 0, 0 ), model.index( 2, 0 ) ) ) );
461     assertFalse( grid.isSatisfiedConstraint( Constraint( model.index( 2, 0 ), model.index( 0, 0 ) ) ) );
462
463     s = grid.mapToChart( model.index( 0, 0 ) );
464     s.setEnd( s.end()+100000. );
465     bool rc = grid.mapFromChart( s, model.index( 0, 0 ) );
466     assertTrue( rc );
467     assertEquals( s1, model.data( model.index( 0, 0 ), StartTimeRole ).toDate() );
468     Span newspan = grid.mapToChart( model.index( 0, 0 ) );
469     assertEquals( newspan.start(), s.start() );
470     assertEquals( newspan.length(), s.length() );
471
472     {
473         QDateTime startDateTime = QDateTime::currentDateTime();
474         qreal dayWidth = 100;
475         QDate currentDate = QDate::currentDate();
476         QDateTime dt( QDate(currentDate.year(), 1, 1), QTime( 0, 0, 0, 0 ) );
477         assert( dt.isValid() );
478         qreal result = startDateTime.date().daysTo(dt.date())*24.*60.*60.;
479         result += startDateTime.time().msecsTo(dt.time())/1000.;
480         result *= dayWidth/( 24.*60.*60. );
481
482         int days = static_cast<int>( result/dayWidth );
483         qreal secs = result*( 24.*60.*60. )/dayWidth;
484         QDateTime dt2 = startDateTime;
485         QDateTime result2 = dt2.addDays( days ).addSecs( static_cast<int>(secs-(days*24.*60.*60.) ) );
486
487         assertEquals( dt, result2 );
488     }
489 }

```

### 13.14.2.3 std::ostream& @13::operator<< (std::ostream & os, const QDateTime & dt)

[static]

Definition at line 419 of file kdganttdatetimegrid.cpp.

```

420     {
421         os << dt.toString().toStdString();
422         return os;
423     }

```

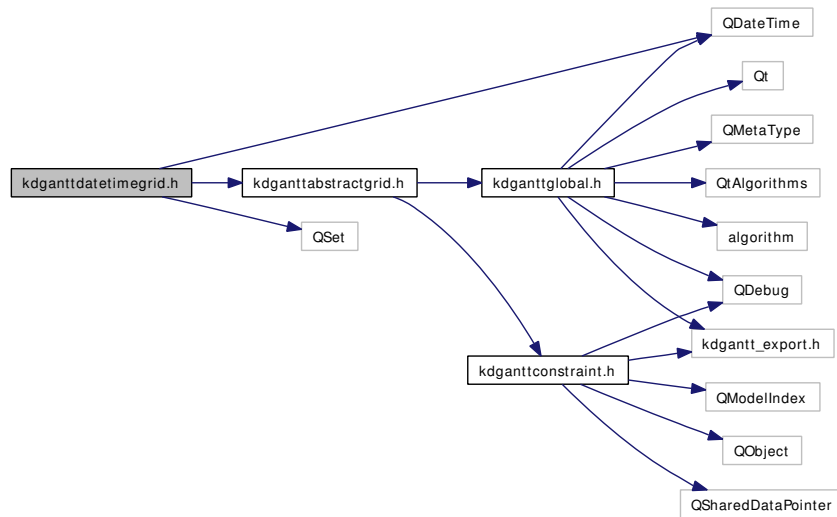
## 13.15 kdganttdatetimegrid.h File Reference

```
#include "kdganttabstractgrid.h"
```

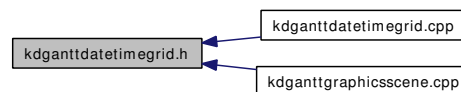
```
#include <QDateTime>
```

```
#include <QSet>
```

Include dependency graph for kdganttdatetimegrid.h:



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



### Namespaces

- namespace [KDGantt](#)

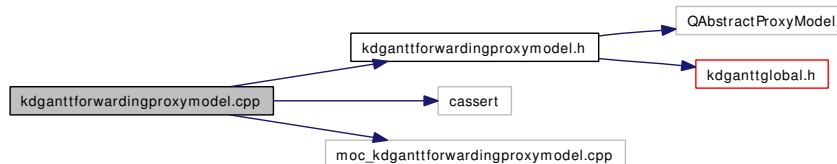
### Classes

- class [KDGantt::DateTimeGrid](#)

## 13.16 kdganttforwardingproxymodel.cpp File Reference

```
#include "kdganttforwardingproxymodel.h"
#include <cassert>
#include "moc_kdganttforwardingproxymodel.cpp"
```

Include dependency graph for kdganttforwardingproxymodel.cpp:



### Typedefs

- typedef [QAbstractProxyModel](#) **BASE**

#### 13.16.1 Typedef Documentation

##### 13.16.1.1 typedef [QAbstractProxyModel](#) **BASE**

Definition at line 31 of file kdganttforwardingproxymodel.cpp.

#### 13.16.2 Variable Documentation

##### 13.16.2.1 int **c**

Definition at line 60 of file kdganttforwardingproxymodel.cpp.

Referenced by `KDGantt::GraphicsView::addConstraint()`, `KDGantt::ConstraintModel::addConstraint()`, `KDGantt::ConstraintModel::cleanup()`, `KDGantt::ConstraintModel::clear()`, `KDGantt::ConstraintModel::constraintsForIndex()`, `KDGantt::GraphicsScene::findConstraintItem()`, `KDGantt::ConstraintModel::hasConstraint()`, `KDGantt::GraphicsScene::insertItem()`, `KDGantt::AbstractGrid::isSatisfiedConstraint()`, `KDGantt::DateTimeGrid::mapFromChart()`, `operator<<()`, `KDGantt::qHash()`, and `KDGantt::ConstraintModel::removeConstraint()`.

##### 13.16.2.2 const [QAbstractItemModel](#)\* **m**

Definition at line 62 of file kdganttforwardingproxymodel.cpp.

##### 13.16.2.3 void\* **p**

Definition at line 61 of file kdganttforwardingproxymodel.cpp.

Referenced by `KDGantt::Legend::paintEvent()`.

### 13.16.2.4 int r

Definition at line 60 of file kdganttforwardingproxymodel.cpp.

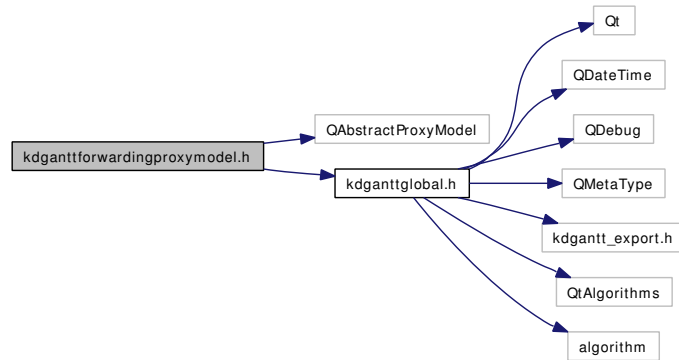
Referenced by `KDGantt::Legend::drawItem()`, `KDGantt::ItemDelegate::paintGanttItem()`, `KDGantt::GraphicsView::resizeEvent()`, `KDGantt::TreeViewRowController::rowGeometry()`, `KDGantt::ListViewRowController::rowGeometry()`, `KDGantt::GraphicsItem::updateItem()`, and `KDGantt::GraphicsView::updateSceneRect()`.

## 13.17 kdganttforwardingproxymodel.h File Reference

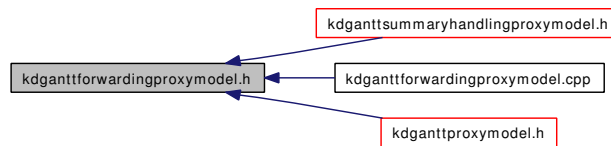
```
#include <QAbstractProxyModel>
```

```
#include "kdganttglobal.h"
```

Include dependency graph for kdganttforwardingproxymodel.h:



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



### Namespaces

- namespace [KDGantt](#)

### Classes

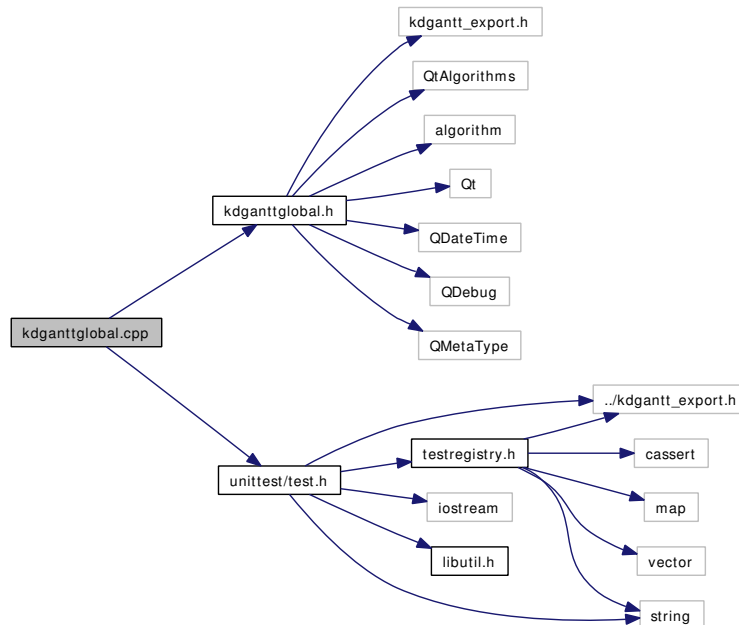
- class [KDGantt::ForwardingProxyModel](#)

## 13.18 kdganttglobal.cpp File Reference

```
#include "kdganttglobal.h"
```

```
#include "unittest/test.h"
```

Include dependency graph for kdganttglobal.cpp:



### Functions

- `KDAB_SCOPED_UNITTEST_SIMPLE (KDGantt, DateTimeSpan, "test")`
- `KDAB_SCOPED_UNITTEST_SIMPLE (KDGantt, Span, "test")`
- `std::ostream & operator<< (std::ostream &os, const DateTimeSpan &span)`
- `std::ostream & operator<< (std::ostream &os, const Span &span)`
- `QDebug operator<< (QDebug dbg, const KDGantt::DateTimeSpan &s)`
- `QDebug operator<< (QDebug dbg, const KDGantt::Span &s)`
- `QDebug operator<< (QDebug dbg, KDGantt::ItemDataRole r)`

### 13.18.1 Function Documentation

#### 13.18.1.1 KDAB\_SCOPED\_UNITTEST\_SIMPLE (KDGantt, DateTimeSpan, "test")

Definition at line 164 of file kdganttglobal.cpp.

References `assertEqual`, `assertFalse`, `assertNotEqual`, `assertTrue`, `KDGantt::DateTimeSpan::isValid()`, `KDGantt::DateTimeSpan::setEnd()`, and `KDGantt::DateTimeSpan::setStart()`.

```

164                                     {
165     DateTimeSpan s1;
166     assertFalse( s1.isValid() );
167     QDateTime dt = QDateTime::currentDateTime();
168     s1.setStart( dt );

```

```

169     assertTrue( dt.isValid() );
170     s1.setEnd( dt.addDays( 1 ) );
171
172     DateTimeSpan s2( dt, dt.addDays( 1 ) );
173
174     assertEquals( s1, s2 );
175
176     DateTimeSpan s3;
177
178     assertNotEqual( s1, s3 );
179 }

```

### 13.18.1.2 KDAB\_SCOPED\_UNITTEST\_SIMPLE (KDGantt, [Span](#), "test")

Definition at line 154 of file kdganttglobal.cpp.

References [assertEquals](#), [assertFalse](#), [KDGantt::Span::isValid\(\)](#), [KDGantt::Span::length\(\)](#), [KDGantt::Span::setLength\(\)](#), [KDGantt::Span::setStart\(\)](#), and [KDGantt::Span::start\(\)](#).

```

154                                     {
155     Span s1;
156     assertFalse( s1.isValid() );
157     s1.setStart( 10. );
158     s1.setLength( 2. );
159
160     Span s2( s1.start(), s1.length() );
161     assertEquals( s1, s2 );
162 }

```

### 13.18.1.3 [std::ostream& @17::operator<<](#) (std::ostream & os, const [DateTimeSpan](#) & span) [static]

Definition at line 146 of file kdganttglobal.cpp.

References [KDGantt::DateTimeSpan::end\(\)](#), and [KDGantt::DateTimeSpan::start\(\)](#).

```

147     {
148         os << "DateTimeSpan[ start=" <<span.start().toString().toString()
149             << ", end=" <<span.end().toString().toString() << " ]";
150         return os;
151     }

```

### 13.18.1.4 [std::ostream& @17::operator<<](#) (std::ostream & os, const [Span](#) & span) [static]

Definition at line 141 of file kdganttglobal.cpp.

References [KDGantt::Span::length\(\)](#), and [KDGantt::Span::start\(\)](#).

```

142     {
143         os << "Span[ start=" <<span.start()<< ", length=" <<span.length()<< " ]";
144         return os;
145     }

```

### 13.18.1.5 QDebug operator<< (QDebug *dbg*, const [KDGantt::DateTimeSpan](#) & *s*)

Definition at line 129 of file kdganttglobal.cpp.

```
130 {
131     dbg << "KDGantt::DateTimeSpan[ start="<<s.start ()<<" end="<<s.end ()<<"]";
132     return dbg;
133 }
```

### 13.18.1.6 QDebug operator<< (QDebug *dbg*, const [KDGantt::Span](#) & *s*)

Definition at line 124 of file kdganttglobal.cpp.

```
125 {
126     dbg << "KDGantt::Span[ start="<<s.start ()<<" length="<<s.length ()<<"]";
127     return dbg;
128 }
```

### 13.18.1.7 QDebug operator<< (QDebug *dbg*, [KDGantt::ItemDataRole](#) *r*)

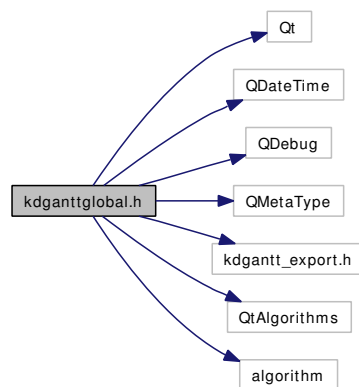
Definition at line 111 of file kdganttglobal.cpp.

```
112 {
113     switch(r){
114     case KDGantt::StartTimeRole:     dbg << "KDGantt::StartTimeRole"; break;
115     case KDGantt::EndTimeRole:      dbg << "KDGantt::EndTimeRole"; break;
116     case KDGantt::TaskCompletionRole: dbg << "KDGantt::TaskCompletionRole"; break;
117     case KDGantt::ItemTypeRole:     dbg << "KDGantt::ItemTypeRole"; break;
118     case KDGantt::LegendRole:       dbg << "KDGantt::LegendRole"; break;
119     default: dbg << static_cast<Qt::ItemDataRole>(r);
120     }
121     return dbg;
122 }
```

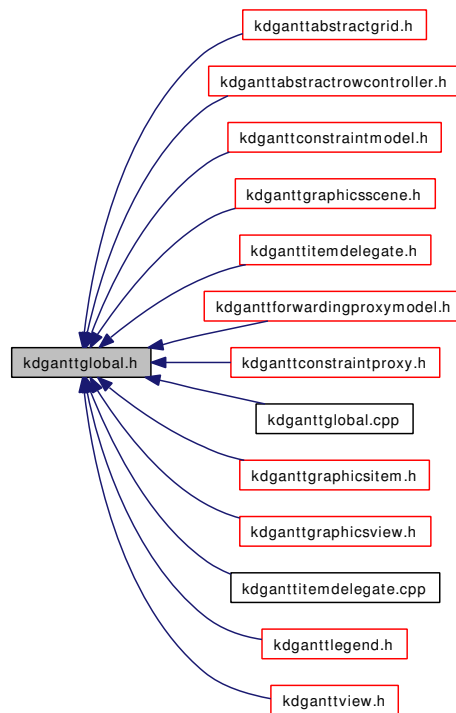
## 13.19 kdganttglobal.h File Reference

```
#include <Qt>
#include <QDateTime>
#include <QDebug>
#include <QMetaType>
#include "kdgantt_export.h"
#include <QtAlgorithms>
#include <algorithm>
```

Include dependency graph for kdganttglobal.h:



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



## Namespaces

- namespace [KDGantt](#)

## Classes

- class [KDGantt::DateTimeSpan](#)
- class [KDGantt::Span](#)

*A class representing a start point and a length.*

## Defines

- #define [KDAB\\_SET\\_OBJECT\\_NAME\(x\)](#) `__kdab__dereference_for_methodcall( x ).setObjectName( QLatin1String( #x ) )`
- #define [KDGANTT\\_DECLARE\\_PRIVATE\\_BASE\\_POLYMORPHIC\(X\)](#)
- #define [KDGANTT\\_DECLARE\\_PRIVATE\\_BASE\\_POLYMORPHIC\\_QWIDGET\(X\)](#)
- #define [KDGANTT\\_DECLARE\\_PRIVATE\\_BASE\\_VALUE\(X\)](#)
- #define [KDGANTT\\_DECLARE\\_PRIVATE\\_DERIVED\(X\)](#)
- #define [KDGANTT\\_DECLARE\\_PRIVATE\\_DERIVED\\_PARENT\(X, ParentType\)](#)
- #define [KDGANTT\\_DECLARE\\_PRIVATE\\_DERIVED\\_QWIDGET\(X\)](#) `KDGANTT_DECLARE_PRIVATE_DERIVED_PARENT( X, QWidget* )`
- #define [KDGANTT\\_DECLARE\\_SWAP\\_BASE\(X\)](#)
- #define [KDGANTT\\_DECLARE\\_SWAP\\_DERIVED\(X\)](#) `void swap( X& other ) { doSwap( other ); }`
- #define [KDGANTT\\_DECLARE\\_SWAP\\_SPECIALISATION\(X\)](#)

- `#define KDGANTT_DECLARE_SWAP_SPECIALISATION_DERIVED(X) KDGANTT_DECLARE_SWAP_SPECIALISATION( X )`

## Enumerations

- `enum KDGantt::ItemDataRole {`  
`KDGantt::KDGanttRoleBase = Qt::UserRole + 1174,`  
`KDGantt::StartTimeRole = KDGanttRoleBase + 1,`  
`KDGantt::EndTimeRole = KDGanttRoleBase + 2,`  
`KDGantt::TaskCompletionRole = KDGanttRoleBase + 3,`  
`KDGantt::ItemTypeRole = KDGanttRoleBase + 4,`  
`KDGantt::LegendRole = KDGanttRoleBase + 5 }`
- `enum KDGantt::ItemType {`  
`KDGantt::TypeNone = 0,`  
`KDGantt::TypeEvent = 1,`  
`KDGantt::TypeTask = 2,`  
`KDGantt::TypeSummary = 3,`  
`KDGantt::TypeMulti = 4,`  
`KDGantt::TypeUser = 1000 }`

## Functions

- `template<typename T> T & __kdab__dereference_for_methodcall (T *o)`
- `template<typename T> T & __kdab__dereference_for_methodcall (T &o)`
- `bool KDGantt::operator!=(const DateTimeSpan &s1, const DateTimeSpan &s2)`
- `bool KDGantt::operator!=(const Span &s1, const Span &s2)`
- `QDebug operator<< (QDebug dbg, const KDGantt::DateTimeSpan &s)`
- `QDebug operator<< (QDebug dbg, const KDGantt::Span &s)`
- `QDebug operator<< (QDebug dbg, KDGantt::ItemDataRole r)`
- `bool KDGantt::operator==(const DateTimeSpan &s1, const DateTimeSpan &s2)`
- `bool KDGantt::operator==(const Span &s1, const Span &s2)`
- `Q_DECLARE_METATYPE (KDGantt::ItemType)`
- `Q_DECLARE_TYPEINFO (KDGantt::DateTimeSpan, Q_MOVABLE_TYPE)`
- `Q_DECLARE_TYPEINFO (KDGantt::Span, Q_MOVABLE_TYPE)`

### 13.19.1 Define Documentation

- 13.19.1.1** `#define KDAB_SET_OBJECT_NAME(x) __kdab__dereference_for_methodcall( x ).setObjectName( QLatin1String( #x ) )`

Definition at line 45 of file `kdganttglobal.h`.

**13.19.1.2 #define KDGANTT\_DECLARE\_PRIVATE\_BASE\_POLYMORPHIC(X)****Value:**

```
protected:                                     \
    class Private;                             \
        friend class Private;                 \
    Private * d_func() { return _d; }          \
    const Private * d_func() const { return _d; } \
    explicit inline X( Private * );           \
private:                                       \
    void init();                               \
    Private * _d;
```

Definition at line 121 of file kdganttglobal.h.

**13.19.1.3 #define KDGANTT\_DECLARE\_PRIVATE\_BASE\_POLYMORPHIC\_QWIDGET(X)****Value:**

```
protected:                                     \
    class Private;                             \
        friend class Private;                 \
    Private * d_func() { return _d; }          \
    const Private * d_func() const { return _d; } \
    explicit inline X( Private *, QWidget* ); \
private:                                       \
    void init();                               \
    Private * _d;
```

Definition at line 146 of file kdganttglobal.h.

**13.19.1.4 #define KDGANTT\_DECLARE\_PRIVATE\_BASE\_VALUE(X)****Value:**

```
public:                                       \
    inline void swap( X & other ) { qSwap( _d, other._d ); } \
protected:                                   \
    class Private;                             \
        friend class Private;                 \
    Private * d_func() { return _d; }          \
    const Private * d_func() const { return _d; } \
private:                                       \
    void init();                               \
    Private * _d;
```

Definition at line 96 of file kdganttglobal.h.

**13.19.1.5 #define KDGANTT\_DECLARE\_PRIVATE\_DERIVED(X)****Value:**

```
protected:                                     \
    class Private;                             \
        friend class Private;
```

```

inline Private * d_func();           \
inline const Private * d_func() const; \
explicit inline X( Private * );     \
private:                             \
    void init();

```

Definition at line 59 of file kdganttglobal.h.

### 13.19.1.6 #define KDGANTT\_DECLARE\_PRIVATE\_DERIVED\_PARENT(X, ParentType)

#### Value:

```

protected:                             \
    class Private;                       \
        friend class Private;           \
    inline Private * d_func();           \
    inline const Private * d_func() const; \
    explicit inline X( Private *, ParentType ); \
private:                                 \
    void init();

```

Definition at line 82 of file kdganttglobal.h.

### 13.19.1.7 #define KDGANTT\_DECLARE\_PRIVATE\_DERIVED\_QWIDGET(X) KDGANTT\_DECLARE\_PRIVATE\_DERIVED\_PARENT( X, [QWidget\\*](#) )

Definition at line 93 of file kdganttglobal.h.

### 13.19.1.8 #define KDGANTT\_DECLARE\_SWAP\_BASE(X)

#### Value:

```

protected: \
    void doSwap( X& other ) \
    { qSwap( _d, other._d); }

```

Definition at line 177 of file kdganttglobal.h.

### 13.19.1.9 #define KDGANTT\_DECLARE\_SWAP\_DERIVED(X) void swap( X& other ) { doSwap( other ); }

Definition at line 182 of file kdganttglobal.h.

### 13.19.1.10 #define KDGANTT\_DECLARE\_SWAP\_SPECIALISATION(X)

#### Value:

```

template <> inline void qSwap<X>( X & lhs, X & rhs ) \
{ lhs.swap( rhs ); } \
namespace std { \
    template <> inline void swap<X>( X & lhs, X & rhs ) \
    { lhs.swap( rhs ); } \
}

```

Definition at line 161 of file kdganttglobal.h.

```
13.19.1.11 #define KDGANTT_DECLARE_SWAP_SPECIALISATION_  
           DERIVED(X) KDGANTT_DECLARE_SWAP_SPECIALISATION( X  
           )
```

Definition at line 174 of file kdganttglobal.h.

## 13.19.2 Function Documentation

### 13.19.2.1 `template<typename T> T& __kdab__dereference_for_methodcall (T * o)`

Definition at line 41 of file kdganttglobal.h.

```
41         {  
42     return *o;  
43 }
```

### 13.19.2.2 `template<typename T> T& __kdab__dereference_for_methodcall (T & o)`

Definition at line 36 of file kdganttglobal.h.

```
36         {  
37     return o;  
38 }
```

### 13.19.2.3 `QDebug operator<< (QDebug dbg, const KDGantt::DateTimeSpan & s)`

Definition at line 129 of file kdganttglobal.cpp.

References [KDGantt::DateTimeSpan::end\(\)](#), and [KDGantt::DateTimeSpan::start\(\)](#).

```
130 {  
131     dbg << "KDGantt::DateTimeSpan[ start="<<s.start ()<<" end="<<s.end ()<<"]";  
132     return dbg;  
133 }
```

### 13.19.2.4 `QDebug operator<< (QDebug dbg, const KDGantt::Span & s)`

Definition at line 124 of file kdganttglobal.cpp.

References [KDGantt::Span::length\(\)](#), and [KDGantt::Span::start\(\)](#).

```
125 {  
126     dbg << "KDGantt::Span[ start="<<s.start ()<<" length="<<s.length ()<<"]";  
127     return dbg;  
128 }
```

### 13.19.2.5 QDebug operator<< (QDebug *dbg*, [KDGantt::ItemDataRole](#) *r*)

Definition at line 111 of file `kdganttglobal.cpp`.

References [KDGantt::EndTimeRole](#), [KDGantt::ItemTypeRole](#), [KDGantt::LegendRole](#), [KDGantt::StartTimeRole](#), and [KDGantt::TaskCompletionRole](#).

```
112 {
113     switch(r) {
114         case KDGantt::StartTimeRole:      dbg << "KDGantt::StartTimeRole"; break;
115         case KDGantt::EndTimeRole:       dbg << "KDGantt::EndTimeRole"; break;
116         case KDGantt::TaskCompletionRole: dbg << "KDGantt::TaskCompletionRole"; break;
117         case KDGantt::ItemTypeRole:      dbg << "KDGantt::ItemTypeRole"; break;
118         case KDGantt::LegendRole:        dbg << "KDGantt::LegendRole"; break;
119         default: dbg << static_cast<Qt::ItemDataRole>(r);
120     }
121     return dbg;
122 }
```

### 13.19.2.6 Q\_DECLARE\_METATYPE ([KDGantt::ItemType](#))

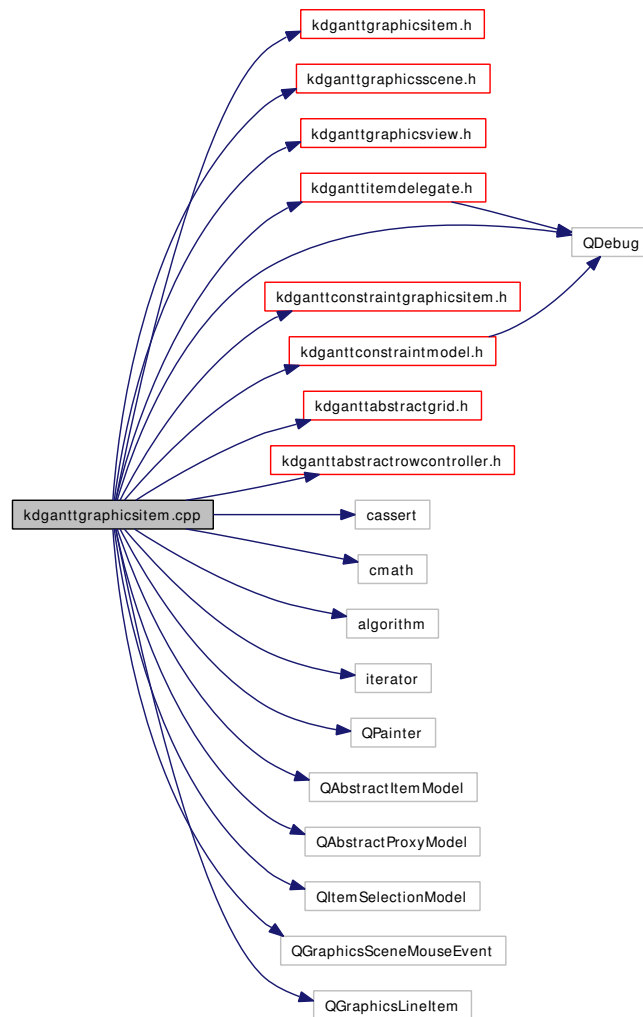
### 13.19.2.7 Q\_DECLARE\_TYPEINFO ([KDGantt::DateTimeSpan](#), Q\_MOVABLE\_TYPE)

### 13.19.2.8 Q\_DECLARE\_TYPEINFO ([KDGantt::Span](#), Q\_MOVABLE\_TYPE)

## 13.20 kdganttgraphicsitem.cpp File Reference

```
#include "kdganttgraphicsitem.h"  
#include "kdganttgraphicsscene.h"  
#include "kdganttgraphicsview.h"  
#include "kdganttitemdelegate.h"  
#include "kdganttconstraintgraphicsitem.h"  
#include "kdganttconstraintmodel.h"  
#include "kdganttabstractgrid.h"  
#include "kdganttabstractrowcontroller.h"  
#include <cassert>  
#include <cmath>  
#include <algorithm>  
#include <iterator>  
#include <QPainter>  
#include <QAbstractItemModel>  
#include <QAbstractProxyModel>  
#include <QItemSelectionModel>  
#include <QGraphicsSceneMouseEvent>  
#include <QGraphicsLineItem>  
#include <QDebug>
```

Include dependency graph for kdganttgraphicsitem.cpp:



## Typedefs

- typedef [QGraphicsItem](#) BASE

### 13.20.1 Typedef Documentation

#### 13.20.1.1 typedef [QGraphicsItem](#) BASE

Definition at line 54 of file kdganttgraphicsitem.cpp.

### 13.20.2 Variable Documentation

#### 13.20.2.1 bool [oldval](#)

Definition at line 59 of file kdganttgraphicsitem.cpp.

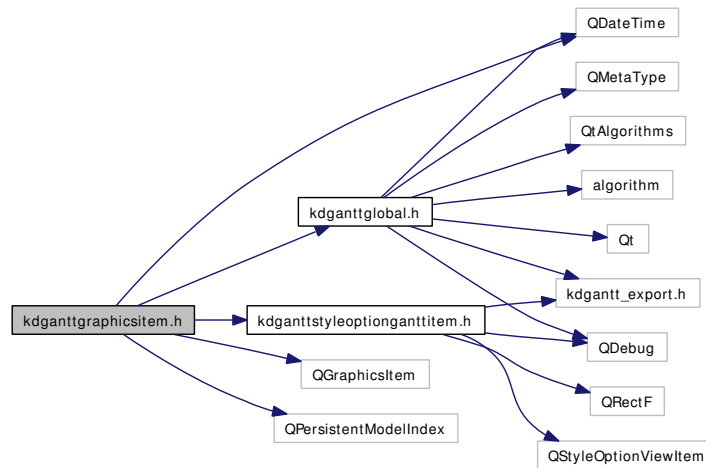
13.20.2.2 `bool* u_ptr`

Definition at line 58 of file kdganttgraphicsitem.cpp.

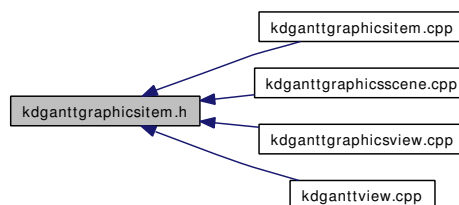
## 13.21 kdganttgraphicsitem.h File Reference

```
#include "kdganttglobal.h"
#include "kdganttstyleoptionganttitem.h"
#include <QGraphicsItem>
#include <QDateTime>
#include <QPersistentModelIndex>
```

Include dependency graph for kdganttgraphicsitem.h:



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



### Namespaces

- namespace [KDGantt](#)

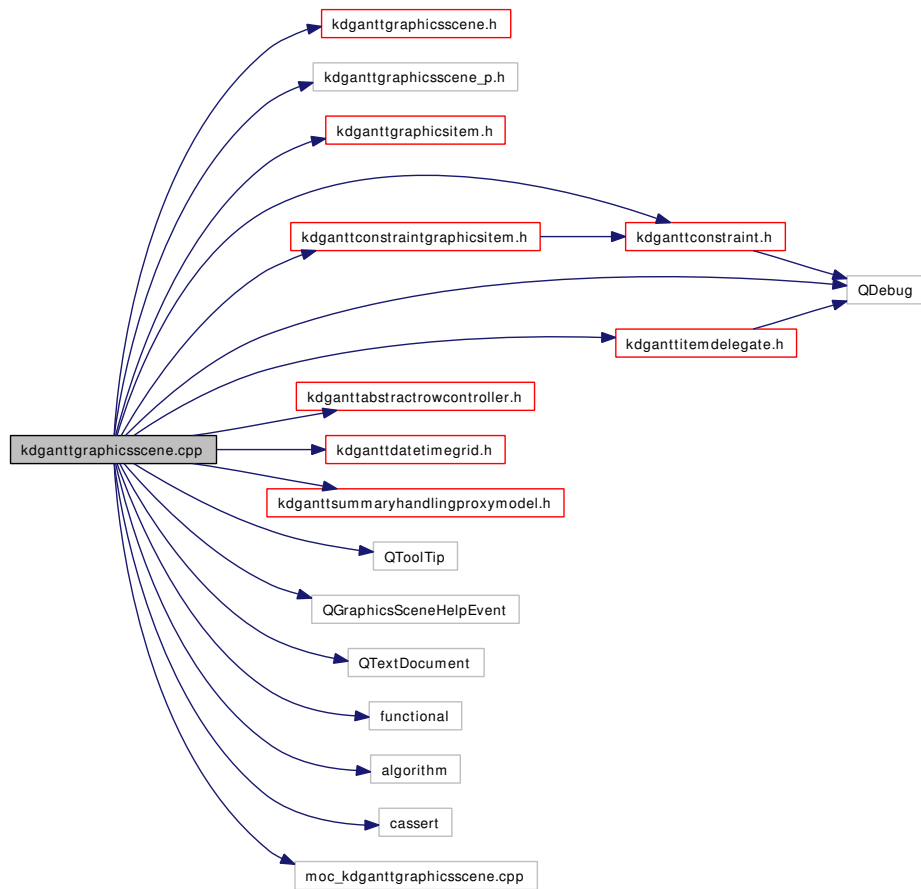
### Classes

- class [KDGantt::GraphicsItem](#)

## 13.22 kdganttgraphicsscene.cpp File Reference

```
#include "kdganttgraphicsscene.h"  
#include "kdganttgraphicsscene_p.h"  
#include "kdganttgraphicsitem.h"  
#include "kdganttconstraint.h"  
#include "kdganttconstraintgraphicsitem.h"  
#include "kdganttitemdelegate.h"  
#include "kdganttabstractrowcontroller.h"  
#include "kdganttdateetimegrid.h"  
#include "kdganttsummaryhandlingproxymodel.h"  
#include <QToolTip>  
#include <QGraphicsSceneHelpEvent>  
#include <QTextDocument>  
#include <QDebug>  
#include <functional>  
#include <algorithm>  
#include <cassert>  
#include "moc_kdganttgraphicsscene.cpp"
```

Include dependency graph for kdganttgraphicsscene.cpp:



## Defines

- `#define d d_func()`

## Functions

- `template<typename Operation1, typename Operation2> unary_compose< Operation1, Operation2 > compose1 (const Operation1 &f, const Operation2 &g)`

### 13.22.1 Define Documentation

#### 13.22.1.1 #define d d\_func()

Definition at line 151 of file kdganttgraphicsscene.cpp.

## 13.22.2 Function Documentation

### 13.22.2.1 `template<typename Operation1, typename Operation2> unary_compose<Operation1,Operation2> @21::compose1 (const Operation1 &f, const Operation2 &g) [static]`

Definition at line 119 of file kdganttgraphicsscene.cpp.

```
120     {  
121         return unary_compose<Operation1,Operation2>( f, g );  
122     }
```

## 13.22.3 Variable Documentation

### 13.22.3.1 `Operation1 _f`

Definition at line 114 of file kdganttgraphicsscene.cpp.

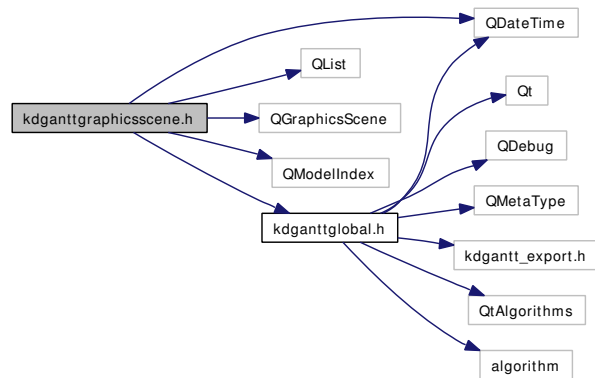
### 13.22.3.2 `Operation2 _g`

Definition at line 115 of file kdganttgraphicsscene.cpp.

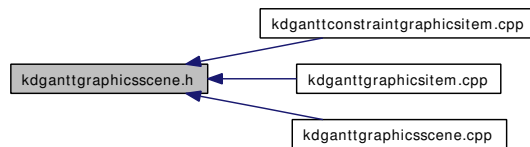
## 13.23 kdganttgraphicsscene.h File Reference

```
#include <QDateTime>
#include <QList>
#include <QGraphicsScene>
#include <QModelIndex>
#include "kdganttglobal.h"
```

Include dependency graph for kdganttgraphicsscene.h:



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



### Namespaces

- namespace [KDGantt](#)

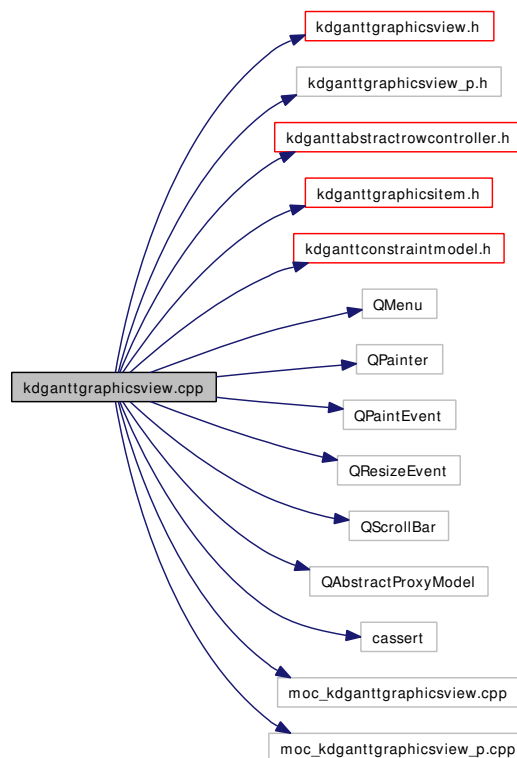
### Classes

- class [KDGantt::GraphicsScene](#)

## 13.24 kdganttgraphicsview.cpp File Reference

```
#include "kdganttgraphicsview.h"  
#include "kdganttgraphicsview_p.h"  
#include "kdganttabstractrowcontroller.h"  
#include "kdganttgraphicsitem.h"  
#include "kdganttconstraintmodel.h"  
#include <QMenu>  
#include <QPainter>  
#include <QPaintEvent>  
#include <QResizeEvent>  
#include <QScrollBar>  
#include <QAbstractProxyModel>  
#include <cassert>  
#include "moc_kdganttgraphicsview.cpp"  
#include "moc_kdganttgraphicsview_p.cpp"
```

Include dependency graph for kdganttgraphicsview.cpp:



## Defines

- #define [d\\_d\\_func\(\)](#)

## Typedefs

- typedef [QGraphicsView](#) [BASE](#)

### 13.24.1 Define Documentation

#### 13.24.1.1 #define [d\\_d\\_func\(\)](#)

Definition at line 325 of file `kdganttgraphicsview.cpp`.

### 13.24.2 Typedef Documentation

#### 13.24.2.1 typedef [QGraphicsView](#) [BASE](#)

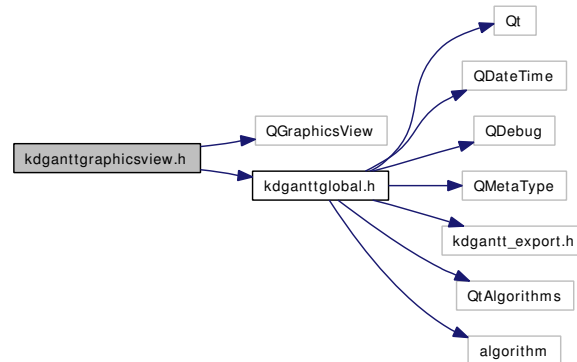
```
void QGraphicsView::pressed( const QModelIndex & index );
```

Definition at line 289 of file `kdganttgraphicsview.cpp`.

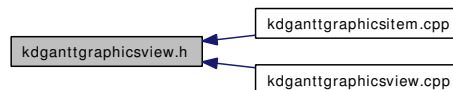
## 13.25 kdganttgraphicsview.h File Reference

```
#include <QGraphicsView>
#include "kdganttglobal.h"
```

Include dependency graph for kdganttgraphicsview.h:



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



### Namespaces

- namespace [KDGantt](#)

### Classes

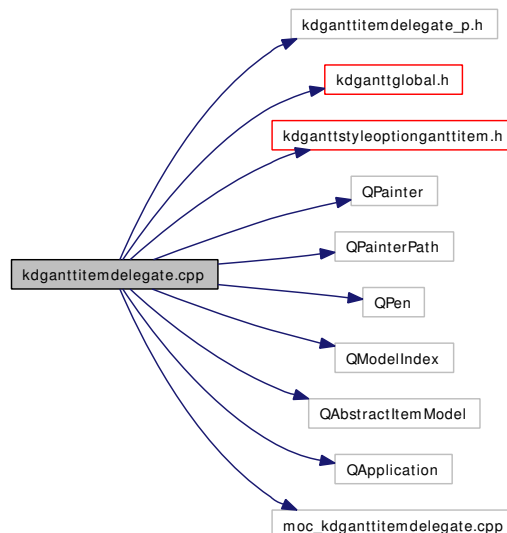
- class [KDGantt::GraphicsView](#)

*The [GraphicsView](#) class provides a model/view implementation of a gantt chart.*

## 13.26 kdganttitemdelegate.cpp File Reference

```
#include "kdganttitemdelegate_p.h"
#include "kdganttglobal.h"
#include "kdganttstyleoptionganttitem.h"
#include <QPainter>
#include <QPainterPath>
#include <QPen>
#include <QModelIndex>
#include <QAbstractItemModel>
#include <QApplication>
#include "moc_kdganttitemdelegate.cpp"
```

Include dependency graph for kdganttitemdelegate.cpp:



### Defines

- #define `d_d_func()`
- #define `PRINT_INTERACTIONSTATE(x)` case x: dbg << #x; break;

### Functions

- static void `adjust_for_horizontal_alignment` (QRectF \*rect, Qt::Alignment a)
- QDebug `operator<<` (QDebug dbg, `KDGantt::ItemDelegate::InteractionState` state)

### Variables

- static const qreal `PW` = 1.5
- static const qreal `TURN` = 10.

## 13.26.1 Define Documentation

### 13.26.1.1 `#define d_d_func()`

Definition at line 113 of file kdganttitemdelegate.cpp.

### 13.26.1.2 `#define PRINT_INTERACTIONSTATE(x) case x: dbg << #x; break;`

Definition at line 38 of file kdganttitemdelegate.cpp.

Referenced by operator<<().

## 13.26.2 Function Documentation

### 13.26.2.1 `static void adjust_for_horizontal_alignment (QRectF * rect, Qt::Alignment a)` [static]

Definition at line 221 of file kdganttitemdelegate.cpp.

```
222 {
223     Q_UNUSED( rect );
224     Q_UNUSED( a );
225 }
```

### 13.26.2.2 `QDebug operator<< (QDebug dbg, KDGantt::ItemDelegate::InteractionState state)`

Definition at line 42 of file kdganttitemdelegate.cpp.

```
43 {
44     switch( state ) {
45         PRINT_INTERACTIONSTATE( KDGantt::ItemDelegate::State_None );
46         PRINT_INTERACTIONSTATE( KDGantt::ItemDelegate::State_Move );
47         PRINT_INTERACTIONSTATE( KDGantt::ItemDelegate::State_ExtendLeft );
48         PRINT_INTERACTIONSTATE( KDGantt::ItemDelegate::State_ExtendRight );
49     default:
50         break;
51     }
52     return dbg;
53 }
```

## 13.26.3 Variable Documentation

### 13.26.3.1 `const qreal PW = 1.5` [static]

Definition at line 344 of file kdganttitemdelegate.cpp.

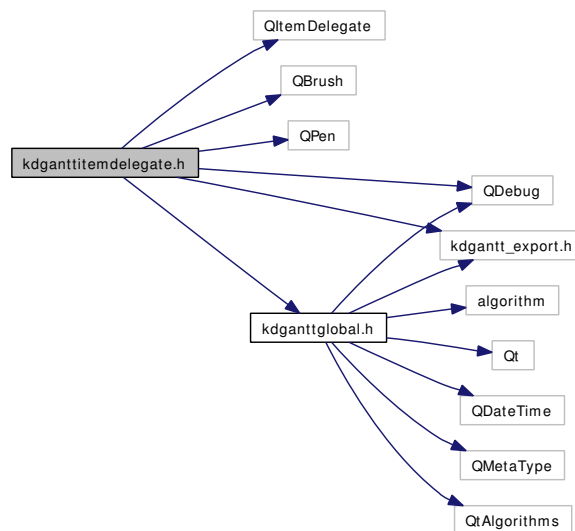
### 13.26.3.2 `const qreal TURN = 10.` [static]

Definition at line 343 of file kdganttitemdelegate.cpp.

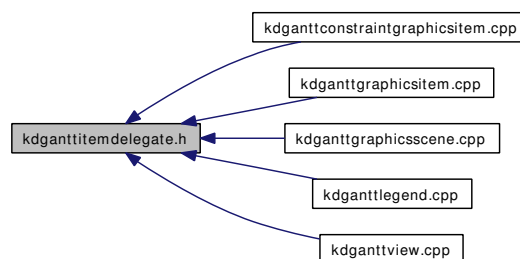
## 13.27 kdganttitemdelegate.h File Reference

```
#include <QItemDelegate>
#include <QBrush>
#include <QPen>
#include <QDebug>
#include "kdgantt_export.h"
#include "kdganttglobal.h"
```

Include dependency graph for kdganttitemdelegate.h:



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



### Namespaces

- namespace [KDGantt](#)

### Classes

- class [KDGantt::ItemDelegate](#)

*Class used to render gantt items in a [KDGantt::GraphicsView](#).*

## Functions

- [QDebug operator<<](#) ([QDebug dbg](#), [KDGantt::ItemDelegate::InteractionState](#))

### 13.27.1 Function Documentation

#### 13.27.1.1 [QDebug operator<<](#) ([QDebug dbg](#), [KDGantt::ItemDelegate::InteractionState](#))

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

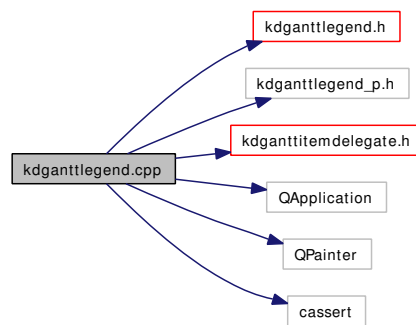
References [PRINT\\_INTERACTIONSTATE](#), [KDGantt::ItemDelegate::State\\_ExtendLeft](#), [KDGantt::ItemDelegate::State\\_ExtendRight](#), [KDGantt::ItemDelegate::State\\_Move](#), and [KDGantt::ItemDelegate::State\\_None](#).

```
43 {
44     switch( state ) {
45         PRINT_INTERACTIONSTATE ( KDGantt::ItemDelegate::State_None );
46         PRINT_INTERACTIONSTATE ( KDGantt::ItemDelegate::State_Move );
47         PRINT_INTERACTIONSTATE ( KDGantt::ItemDelegate::State_ExtendLeft );
48         PRINT_INTERACTIONSTATE ( KDGantt::ItemDelegate::State_ExtendRight );
49     default:
50         break;
51     }
52     return dbg;
53 }
```

## 13.28 kdganttlegend.cpp File Reference

```
#include "kdganttlegend.h"  
#include "kdganttlegend_p.h"  
#include "kdganttitemdelegate.h"  
#include <QApplication>  
#include <QPainter>  
#include <cassert>
```

Include dependency graph for kdganttlegend.cpp:



### Defines

- #define `d d_func()`

#### 13.28.1 Define Documentation

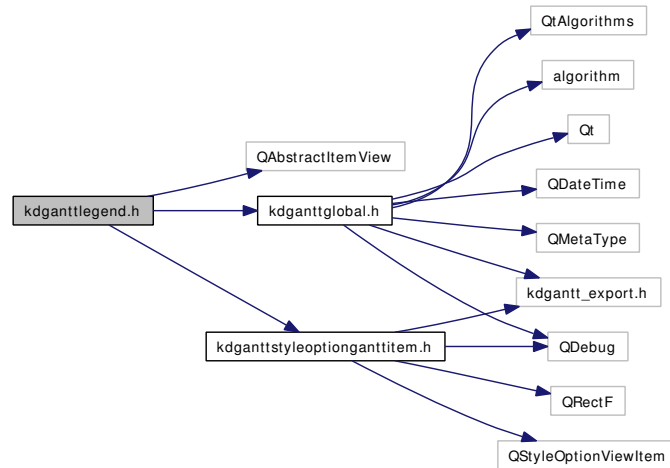
##### 13.28.1.1 #define `d d_func()`

Definition at line 61 of file kdganttlegend.cpp.

## 13.29 kdganttlegend.h File Reference

```
#include <QAbstractItemView>
#include "kdganttglobal.h"
#include "kdganttstyleoptionganttitem.h"
```

Include dependency graph for kdganttlegend.h:



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



### Namespaces

- namespace [KDGantt](#)

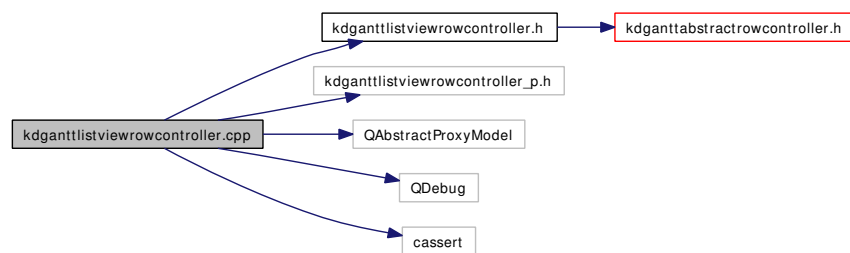
### Classes

- class [KDGantt::Legend](#)  
*Legend showing an image and a description for Gantt items.*

## 13.30 kdganttlistviewrowcontroller.cpp File Reference

```
#include "kdganttlistviewrowcontroller.h"  
#include "kdganttlistviewrowcontroller_p.h"  
#include <QAbstractProxyModel>  
#include <QDebug>  
#include <cassert>
```

Include dependency graph for kdganttlistviewrowcontroller.cpp:



### Defines

- `#define d d_func()`

#### 13.30.1 Define Documentation

##### 13.30.1.1 `#define d d_func()`

Definition at line 52 of file `kdganttlistviewrowcontroller.cpp`.

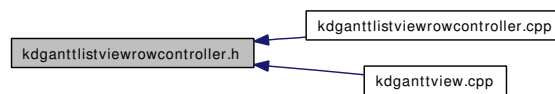
## 13.31 kdganttlistviewrowcontroller.h File Reference

```
#include "kdganttabstractrowcontroller.h"
```

Include dependency graph for kdganttlistviewrowcontroller.h:



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



### Namespaces

- namespace [KDGantt](#)

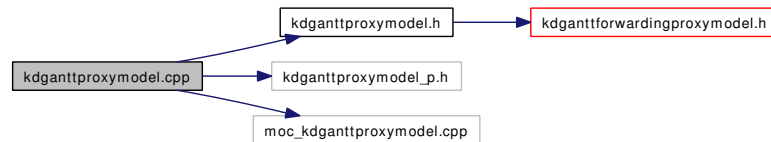
### Classes

- class [KDGantt::ListViewRowController](#)

## 13.32 kdganttproxymodel.cpp File Reference

```
#include "kdganttproxymodel.h"  
#include "kdganttproxymodel_p.h"  
#include "moc_kdganttproxymodel.cpp"
```

Include dependency graph for kdganttproxymodel.cpp:



### Defines

- #define [d d\\_func\(\)](#)

### Typedefs

- typedef [ForwardingProxyModel](#) [BASE](#)

#### 13.32.1 Define Documentation

##### 13.32.1.1 #define [d d\\_func\(\)](#)

Definition at line 66 of file kdganttproxymodel.cpp.

#### 13.32.2 Typedef Documentation

##### 13.32.2.1 typedef [ForwardingProxyModel](#) [BASE](#)

Definition at line 31 of file kdganttproxymodel.cpp.

## 13.33 kdganttproxymodel.h File Reference

```
#include "kdganttforwardingproxymodel.h"
```

Include dependency graph for kdganttproxymodel.h:



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



### Namespaces

- namespace [KDGantt](#)

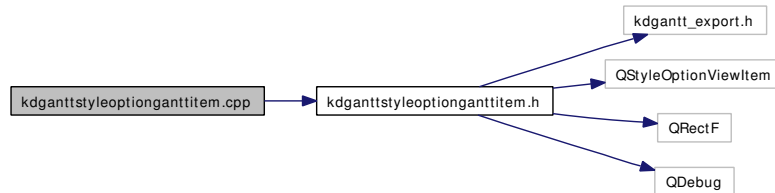
### Classes

- class [KDGantt::ProxyModel](#)

## 13.34 kdganttstyleoptionganttitem.cpp File Reference

```
#include "kdganttstyleoptionganttitem.h"
```

Include dependency graph for kdganttstyleoptionganttitem.cpp:



### Typedefs

- typedef [QStyleOptionViewItem](#) BASE

### Functions

- [QDebug](#) operator<< ([QDebug](#) dbg, const [KD Gantt::StyleOptionGanttItem](#) &s)

#### 13.34.1 Typedef Documentation

##### 13.34.1.1 typedef [QStyleOptionViewItem](#) BASE

Definition at line 34 of file kdganttstyleoptionganttitem.cpp.

#### 13.34.2 Function Documentation

##### 13.34.2.1 [QDebug](#) operator<< ([QDebug](#) *dbg*, const [KD Gantt::StyleOptionGanttItem](#) & *s*)

Definition at line 66 of file kdganttstyleoptionganttitem.cpp.

References [KD Gantt::StyleOptionGanttItem::boundingRect](#), [KD Gantt::StyleOptionGanttItem::displayPosition](#), [KD Gantt::StyleOptionGanttItem::grid](#), [KD Gantt::StyleOptionGanttItem::itemRect](#), and [KD Gantt::StyleOptionGanttItem::text](#).

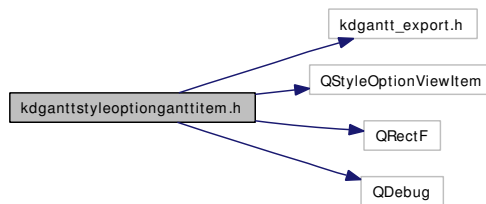
```

67 {
68     dbg << "KD Gantt::StyleOptionGanttItem[ boundingRect="<<s.boundingRect
69         <<", itemRect="<<s.itemRect
70         <<", displayPosition="<<s.displayPosition
71         <<", grid="<<s.grid
72         <<", text="<<s.text
73         <<"]";
74     return dbg;
75 }
```

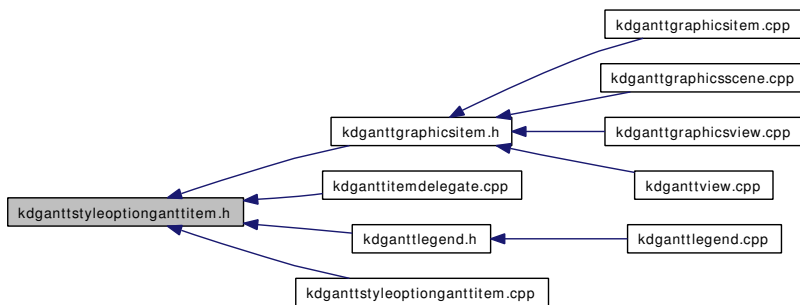
## 13.35 kdganttstyleoptiongantitem.h File Reference

```
#include "kdgantt_export.h"
#include <QStyleOptionViewItem>
#include <QRectF>
#include <QDebug>
```

Include dependency graph for kdganttstyleoptiongantitem.h:



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



### Namespaces

- namespace [KDGantt](#)

### Classes

- class [KDGantt::StyleOptionGanttItem](#)  
*QStyleOption subclass for gantt items.*

### Functions

- QDebug [operator<<](#) (QDebug dbg, const [KDGantt::StyleOptionGanttItem](#) &s)

## 13.35.1 Function Documentation

### 13.35.1.1 QDebug operator<< (QDebug *dbg*, const [KDGantt::StyleOptionGanttItem](#) & *s*)

Definition at line 66 of file `kdganttstyleoptionganttitem.cpp`.

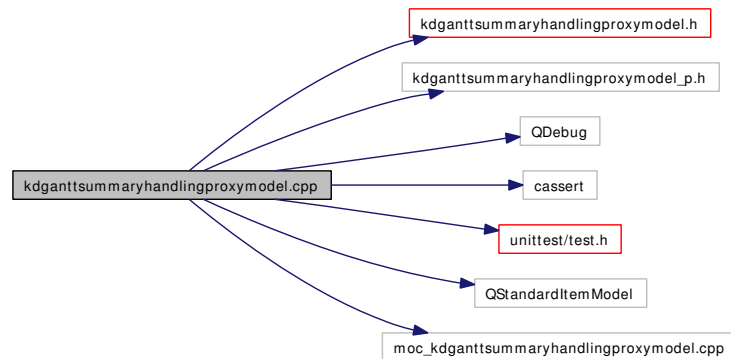
References [KDGantt::StyleOptionGanttItem::boundingRect](#), [KDGantt::StyleOptionGanttItem::displayPosition](#), [KDGantt::StyleOptionGanttItem::grid](#), [KDGantt::StyleOptionGanttItem::itemRect](#), and [KDGantt::StyleOptionGanttItem::text](#).

```
67 {
68     dbg << "KDGantt::StyleOptionGanttItem[ boundingRect="<<s.boundingRect
69         <<", itemRect="<<s.itemRect
70         <<", displayPosition="<<s.displayPosition
71         <<", grid="<<s.grid
72         <<", text="<<s.text
73         <<"]";
74     return dbg;
75 }
```

## 13.36 kdganttsummaryhandlingproxymodel.cpp File Reference

```
#include "kdganttsummaryhandlingproxymodel.h"
#include "kdganttsummaryhandlingproxymodel_p.h"
#include <QDebug>
#include <cassert>
#include "unittest/test.h"
#include <QStandardItemModel>
#include "moc_kdganttsummaryhandlingproxymodel.cpp"
```

Include dependency graph for kdganttsummaryhandlingproxymodel.cpp:



### Defines

- `#define d d_func()`

### Typedefs

- typedef [ForwardingProxyModel](#) BASE

### Functions

- `KDAB_SCOPED_UNITTEST_SIMPLE` (KDGantt, [SummaryHandlingProxyModel](#), "test")
- `std::ostream & operator<<` (std::ostream &os, const QDateTime &dt)

#### 13.36.1 Define Documentation

##### 13.36.1.1 `#define d d_func()`

Definition at line 126 of file kdganttsummaryhandlingproxymodel.cpp.

## 13.36.2 Typedef Documentation

### 13.36.2.1 typedef [ForwardingProxyModel](#) BASE

Definition at line 50 of file `kdganttsummaryhandlingproxymodel.cpp`.

## 13.36.3 Function Documentation

### 13.36.3.1 `KDAB_SCOPED_UNITTEST_SIMPLE` ([KDGantt](#), [SummaryHandlingProxyModel](#), "test")

Definition at line 282 of file `kdganttsummaryhandlingproxymodel.cpp`.

References `assertEqual`, `assertFalse`, `assertTrue`, `KDGantt::SummaryHandlingProxyModel::data()`, `KDGantt::EndTimeRole`, `KDGantt::SummaryHandlingProxyModel::flags()`, `KDGantt::ForwardingProxyModel::index()`, `KDGantt::ItemTypeRole`, `KDGantt::SummaryHandlingProxyModel::setSourceModel()`, `KDGantt::StartTimeRole`, `KDGantt::TypeSummary`, and `KDGantt::TypeTask`.

```

282                                     {
283     SummaryHandlingProxyModel model;
284     QStandardItemModel sourceModel;
285
286     model.setSourceModel( &sourceModel );
287
288     QStandardItem* topitem = new QStandardItem( QString::fromLatin1( "Summary" ) );
289     topitem->setData( KDGantt::TypeSummary, KDGantt::ItemTypeRole );
290     sourceModel.appendRow( topitem );
291
292     QStandardItem* task1 = new QStandardItem( QString::fromLatin1( "Task1" ) );
293     task1->setData( KDGantt::TypeTask, KDGantt::ItemTypeRole );
294     QStandardItem* task2 = new QStandardItem( QString::fromLatin1( "Task2" ) );
295     task2->setData( KDGantt::TypeTask, KDGantt::ItemTypeRole );
296     topitem->appendRow( task1 );
297     topitem->appendRow( task2 );
298
299
300     QDateTime startdt = QDateTime::currentDateTime();
301     QDateTime enddt = startdt.addDays( 1 );
302
303
304     task1->setData( startdt, KDGantt::StartTimeRole );
305     task1->setData( enddt, KDGantt::EndTimeRole );
306     task2->setData( startdt, KDGantt::StartTimeRole );
307     task2->setData( enddt, KDGantt::EndTimeRole );
308
309     const QModelIndex topidx = model.index( 0, 0, QModelIndex() );
310
311     assertEquals( model.data( topidx, KDGantt::ItemTypeRole ).toInt(), KDGantt::TypeSummary );
312     assertEquals( model.data( model.index( 0, 0, topidx ), KDGantt::ItemTypeRole ).toInt(), KDGantt::Ty
313
314     QDateTime task1startdt = model.data( model.index( 0, 0, topidx ), KDGantt::StartTimeRole ).toDateT
315     assertEquals( task1startdt, startdt );
316
317     QDateTime summarystartdt = model.data( topidx, KDGantt::StartTimeRole ).toDateTime();
318     assertEquals( summarystartdt, startdt );
319     assertTrue( model.flags( model.index( 0, 0, topidx ) ) & Qt::ItemIsEditable );
320     assertFalse( model.flags( topidx ) & Qt::ItemIsEditable );
321 }

```

### 13.36.3.2 `std::ostream& @35::operator<< (std::ostream & os, const QDateTime & dt)` [static]

Definition at line 275 of file kdganttsummaryhandlingproxymodel.cpp.

```
276     {  
277         os << dt.toString().toString();  
278         return os;  
279     }
```

## 13.36.4 Variable Documentation

### 13.36.4.1 `int c`

Definition at line 139 of file kdganttsummaryhandlingproxymodel.cpp.

### 13.36.4.2 `const QAbstractItemModel* m`

Definition at line 141 of file kdganttsummaryhandlingproxymodel.cpp.

### 13.36.4.3 `void* p`

Definition at line 140 of file kdganttsummaryhandlingproxymodel.cpp.

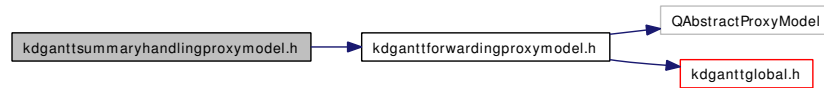
### 13.36.4.4 `int r`

Definition at line 139 of file kdganttsummaryhandlingproxymodel.cpp.

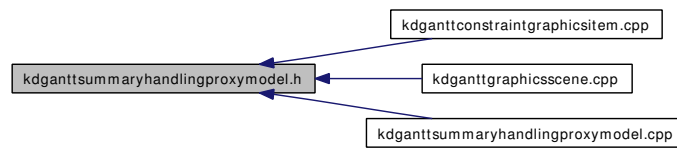
## 13.37 kdganttsummaryhandlingproxymodel.h File Reference

```
#include "kdganttforwardingproxymodel.h"
```

Include dependency graph for kdganttsummaryhandlingproxymodel.h:



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



### Namespaces

- namespace [KDGantt](#)

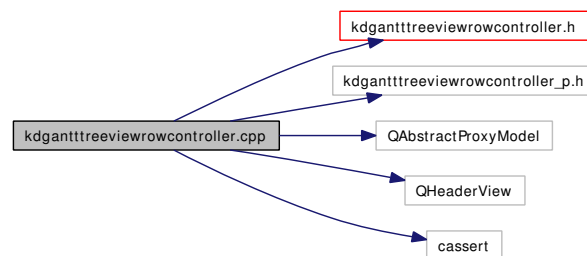
### Classes

- class [KDGantt::SummaryHandlingProxyModel](#)  
*Proxy model that supports summary gantt items.*

## 13.38 kdgantttreeviewrowcontroller.cpp File Reference

```
#include "kdgantttreeviewrowcontroller.h"  
#include "kdgantttreeviewrowcontroller_p.h"  
#include <QAbstractProxyModel>  
#include <QHeaderView>  
#include <cassert>
```

Include dependency graph for kdgantttreeviewrowcontroller.cpp:



### Defines

- #define `d_d_func()`

### 13.38.1 Define Documentation

#### 13.38.1.1 #define `d_d_func()`

Definition at line 53 of file kdgantttreeviewrowcontroller.cpp.

### 13.39 kdgantttreeviewrowcontroller.h File Reference

```
#include "kdganttabstractrowcontroller.h"
```

Include dependency graph for kdgantttreeviewrowcontroller.h:



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



#### Namespaces

- namespace [KDGantt](#)

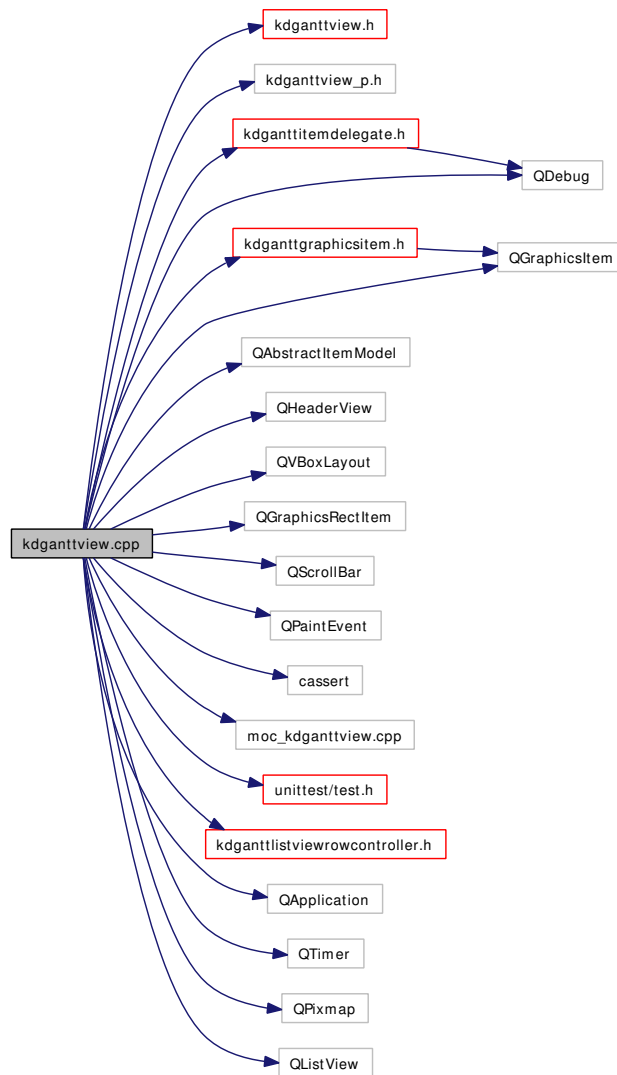
#### Classes

- class [KDGantt::TreeViewRowController](#)

## 13.40 kdganttview.cpp File Reference

```
#include "kdganttview.h"  
#include "kdganttview_p.h"  
#include "kdganttitemdelegate.h"  
#include "kdganttgraphicsitem.h"  
#include <QAbstractItemModel>  
#include <QHeaderView>  
#include <QVBoxLayout>  
#include <QGraphicsItem>  
#include <QGraphicsRectItem>  
#include <QScrollBar>  
#include <QPaintEvent>  
#include <QDebug>  
#include <cassert>  
#include "moc_kdganttview.cpp"  
#include "unittest/test.h"  
#include "kdganttlistviewrowcontroller.h"  
#include <QApplication>  
#include <QTimer>  
#include <QPixmap>  
#include <QListView>
```

Include dependency graph for kdganttview.cpp:



## Defines

- `#define d_d_func()`

## Functions

- `KDAB_SCOPED_UNITTEST_SIMPLE` (KDGantt, View, "test")
- `std::ostream & operator<<` (std::ostream &os, const QImage &img)

### 13.40.1 Define Documentation

#### 13.40.1.1 `#define d_d_func()`

Definition at line 205 of file kdganttview.cpp.

## 13.40.2 Function Documentation

### 13.40.2.1 KDAB\_SCOPED\_UNITTEST\_SIMPLE (KDGantt, View, "test")

Definition at line 486 of file kdganttview.cpp.

References `assertEqual`, `KDGantt::View::ganttProxyModel()`, `KDGantt::View::setLeftView()`, and `KDGantt::View::setRowController()`.

```

486                                     {
487     View view( 0 );
488 #if 0 // GUI tests do not work well on the server
489     QTimer::singleShot( 1000, qApp, SLOT( quit() ) );
490     view.show();
491
492     qApp->exec();
493     QPixmap screenshot1 = QPixmap::grabWidget( &view );
494
495     QTreeView* tv = new QTreeView;
496     view.setLeftView( tv );
497     view.setRowController( new TreeViewRowController(tv,view.ganttProxyModel()) );
498
499     QTimer::singleShot( 1000, qApp, SLOT( quit() ) );
500
501     qApp->exec();
502     QPixmap screenshot2 = QPixmap::grabWidget( &view );
503
504     assertEquals( screenshot1.toImage(), screenshot2.toImage() );
505
506     QListView* lv = new QListView;
507     view.setLeftView(lv);
508     view.setRowController( new ListViewRowController(lv,view.ganttProxyModel()) );
509     view.show();
510     QTimer::singleShot( 1000, qApp, SLOT( quit() ) );
511     qApp->exec();
512 #endif
513 }
```

### 13.40.2.2 std::ostream& @39::operator<< (std::ostream & os, const QImage & img) [static]

Definition at line 479 of file kdganttview.cpp.

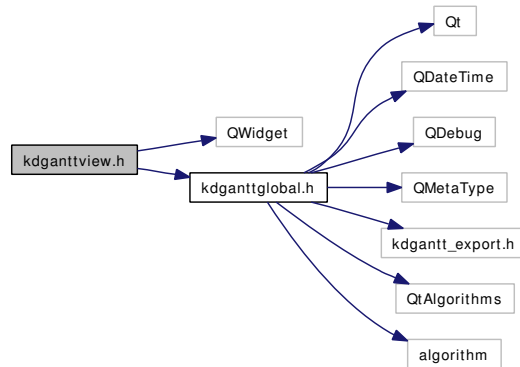
```

480     {
481         os << "QImage[ size=( "<<img.width()<<" , "<<img.height()<<" )]";
482         return os;
483     }
```

## 13.41 kdganttview.h File Reference

```
#include <QWidget>
#include "kdganttglobal.h"
```

Include dependency graph for kdganttview.h:



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



### Namespaces

- namespace [KDGantt](#)

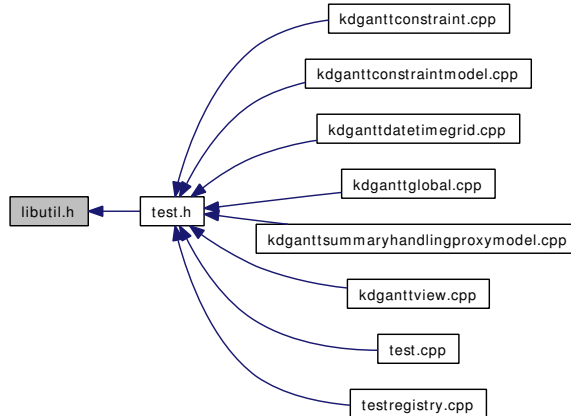
### Classes

- class [KDGantt::View](#)

*This widget that consists of a [QTreeView](#) and a [GraphicsView](#).*

## 13.42 libutil.h File Reference

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



### Defines

- #define [KDAB\\_EXPORT\\_STATIC\\_SYMBOLS\(ID\)](#) int \_\_init\_##ID##\_static\_symbols(){ return 0; }
- #define [KDAB\\_IMPORT\\_STATIC\\_SYMBOLS\(ID\)](#)

#### 13.42.1 Define Documentation

- 13.42.1.1** #define [KDAB\\_EXPORT\\_STATIC\\_SYMBOLS\(ID\)](#) int \_\_init\_##ID##\_static\_symbols(){ return 0; }

Definition at line 11 of file libutil.h.

- 13.42.1.2** #define [KDAB\\_IMPORT\\_STATIC\\_SYMBOLS\(ID\)](#)

#### Value:

```
extern int __init_##ID##_static_symbols(); \
static int fake_init##ID = __init_##ID##_static_symbols();
```

Definition at line 12 of file libutil.h.

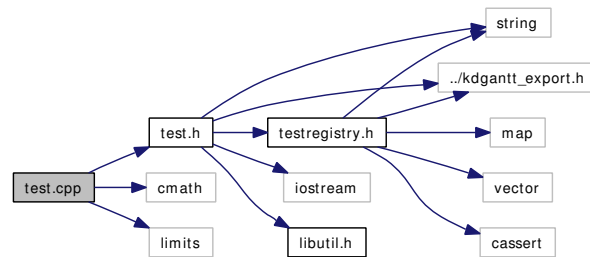
### 13.43 test.cpp File Reference

```
#include "test.h"
```

```
#include <cmath>
```

```
#include <limits>
```

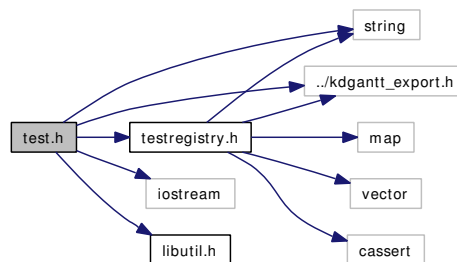
Include dependency graph for test.cpp:



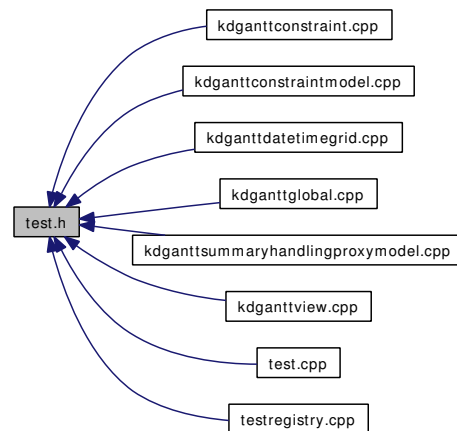
## 13.44 test.h File Reference

```
#include <string>
#include <iostream>
#include "../kdgantt_export.h"
#include "testregistry.h"
#include "libutil.h"
```

Include dependency graph for test.h:



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



### Namespaces

- namespace [KDAB](#)
- namespace [KDAB::UnitTest](#)

### Classes

- class [KDAB::UnitTest::GenericFactory< T\\_Test >](#)
- class [KDAB::UnitTest::Test](#)
- class [KDAB::UnitTest::TestFactory](#)

## Defines

- #define `assertDoesNotThrowException(x, E)`
- #define `assertEqual(x, y) _assertEqual( (x), (y), #x, #y, __FILE__, __LINE__ )`
- #define `assertEqualWithEpsilons(x, y, z) _assertEqualWithEpsilons( (x), (y), (z), #x, #y, #z, __FILE__, __LINE__ )`
- #define `assertFalse(x) _assertFalse( (x), #x, __FILE__, __LINE__ )`
- #define `assertNearEqual(x, y, z)`
- #define `assertNotEqual(x, y) _assertNotEqual( (x), (y), #x, #y, __FILE__, __LINE__ )`
- #define `assertNotNull(x) _assertNotNull( (x), #x, __FILE__, __LINE__ )`
- #define `assertNull(x) _assertNull( (x), #x, __FILE__, __LINE__ )`
- #define `assertThrowsException(x, E) assertThrowsExceptionWithCode( x, E, do{ }while(0) )`
- #define `assertThrowsExceptionWithCode(x, E, code)`
- #define `assertTrue(x) _assertTrue( (x), #x, __FILE__, __LINE__ )`
- #define `KDAB_EXPORT_SCOPED_UNITTEST(Namespace, Class, Group)`
- #define `KDAB_EXPORT_UNITTEST(Class, Group)`
- #define `KDAB_IMPORT_UNITTEST(Class) KDAB_IMPORT_STATIC_SYMBOLS( Class )`
- #define `KDAB_SCOPED_UNITTEST_SIMPLE(Namespace, Class, Group)`
- #define `KDAB_UNITTEST_SIMPLE(Class, Group)`

### 13.44.1 Define Documentation

#### 13.44.1.1 #define `assertDoesNotThrowException(x, E)`

##### Value:

```
do {
    try {
        x;
        success();
    } catch ( E & ) {
        fail( __FILE__, __LINE__ ) \
        << "\"" #x "\" threw \"" #E "\" , but shouldn't" << std::endl; \
    } catch ( ... ) {
        fail( __FILE__, __LINE__ ) \
        << "\"" #x "\" threw something, but it wasn't \"" #E "\" << std::endl; \
    }
} while ( false )
```

Definition at line 46 of file test.h.

#### 13.44.1.2 #define `assertEqual(x, y) _assertEqual( (x), (y), #x, #y, __FILE__, __LINE__ )`

Definition at line 18 of file test.h.

Referenced by `KDAB_SCOPED_UNITTEST_SIMPLE()`.

#### 13.44.1.3 #define `assertEqualWithEpsilons(x, y, z) _assertEqualWithEpsilons( (x), (y), (z), #x, #y, #z, __FILE__, __LINE__ )`

Definition at line 22 of file test.h.

**13.44.1.4 #define assertFalse(x) \_assertFalse( (x), #x, \_\_FILE\_\_, \_\_LINE\_\_ )**

Definition at line 17 of file test.h.

Referenced by KDAB\_SCOPED\_UNITTEST\_SIMPLE().

**13.44.1.5 #define assertNearEqual(x, y, z)**

Definition at line 21 of file test.h.

**13.44.1.6 #define assertNotEqual(x, y) \_assertNotEqual( (x), (y), #x, #y, \_\_FILE\_\_, \_\_LINE\_\_ )**

Definition at line 19 of file test.h.

Referenced by KDAB\_SCOPED\_UNITTEST\_SIMPLE().

**13.44.1.7 #define assertNotNull(x) \_assertNotNull( (x), #x, \_\_FILE\_\_, \_\_LINE\_\_ )**

Definition at line 14 of file test.h.

**13.44.1.8 #define assertNull(x) \_assertNull( (x), #x, \_\_FILE\_\_, \_\_LINE\_\_ )**

Definition at line 15 of file test.h.

**13.44.1.9 #define assertThrowsException(x, E) assertThrowsExceptionWithCode( x, E, do{}while(0) )**

Definition at line 44 of file test.h.

**13.44.1.10 #define assertThrowsExceptionWithCode(x, E, code)****Value:**

```
do {
    try {
        x;
        fail( __FILE__, __LINE__ ) \
            << "\"" #x "\" didn't throw \"" #E "\"" << std::endl; \
    } catch ( E & ppq_ut_thrown ) {
        success();
        ( void )ppq_ut_thrown;
        code;
    } catch ( ... ) {
        fail( __FILE__, __LINE__ ) \
            << "\"" #x "\" threw something, but it wasn't \"" #E "\"" << std::endl; \
    }
} while ( false )
```

Definition at line 28 of file test.h.

**13.44.1.11 #define assertTrue(x) \_assertTrue( (x), #x, \_\_FILE\_\_, \_\_LINE\_\_ )**

Definition at line 16 of file test.h.

Referenced by KDAB\_SCOPED\_UNITTEST\_SIMPLE().

**13.44.1.12 #define KDAB\_EXPORT\_SCOPED\_UNITTEST(Namespace, Class, Group)****Value:**

```
static const KDAB::UnitTest::GenericFactory< Namespace::Class > __##Class##_unittest( Group ); \
    KDAB_EXPORT_STATIC_SYMBOLS( Class )
```

Definition at line 144 of file test.h.

**13.44.1.13 #define KDAB\_EXPORT\_UNITTEST(Class, Group)****Value:**

```
static const KDAB::UnitTest::GenericFactory< Class > __##Class##_unittest( Group ); \
    KDAB_EXPORT_STATIC_SYMBOLS( Class )
```

Definition at line 140 of file test.h.

**13.44.1.14 #define KDAB\_IMPORT\_UNITTEST(Class) KDAB\_IMPORT\_STATIC\_SYMBOLS(Class)**

Definition at line 149 of file test.h.

**13.44.1.15 #define KDAB\_SCOPED\_UNITTEST\_SIMPLE(Namespace, Class, Group)****Value:**

```
namespace Namespace {
    class Class##Test : public KDAB::UnitTest::Test { \
    public:
        Class##Test() : Test( #Namespace "://" #Class ){} \
        void run(); \
    }; \
} \
KDAB_EXPORT_SCOPED_UNITTEST( Namespace, Class##Test, Group ) \
void Namespace::Class##Test::run()
```

Definition at line 162 of file test.h.

**13.44.1.16 #define KDAB\_UNITTEST\_SIMPLE(Class, Group)****Value:**

```
class Class##Test : public KDAB::UnitTest::Test { \
    public:
        Class##Test() : Test( #Class ) {} \
}
```

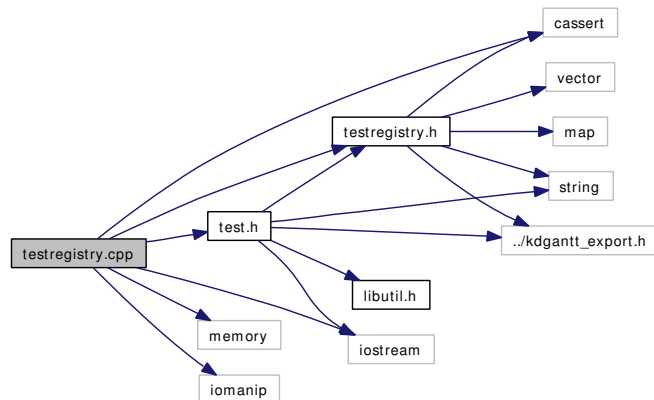
```
void run();
};
KDAB_EXPORT_UNITTEST( Class##Test, Group )
void Class##Test::run()
```

Definition at line 153 of file test.h.

## 13.45 testregistry.cpp File Reference

```
#include "testregistry.h"  
#include "test.h"  
#include <memory>  
#include <iostream>  
#include <iomanip>  
#include <cassert>
```

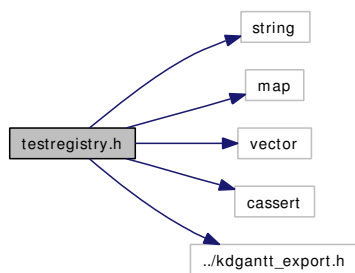
Include dependency graph for testregistry.cpp:



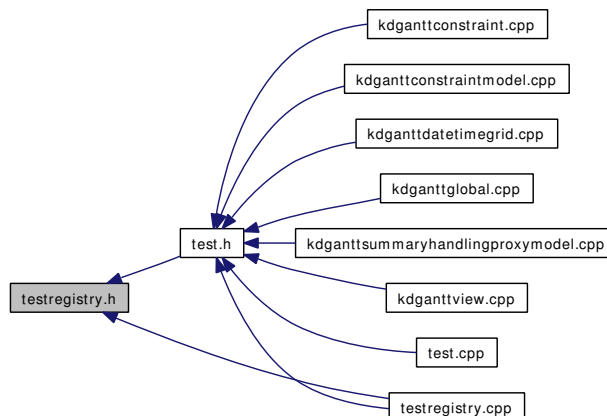
## 13.46 testregistry.h File Reference

```
#include <string>
#include <map>
#include <vector>
#include <cassert>
#include "../kdgantt_export.h"
```

Include dependency graph for testregistry.h:



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



### Namespaces

- namespace [KDAB](#)
- namespace [KDAB::UnitTest](#)

### Classes

- class [KDAB::UnitTest::Runner](#)
- class [KDAB::UnitTest::TestRegistry](#)



# Chapter 14

## KD Gantt 2 Page Documentation

### 14.1 Todo List

**Class [KDGantt::DateTimeGrid](#)** Extend to work with hours, minutes,... as units too.

**Member [KDGantt::ItemDelegate::defaultBrush\(ItemType type\) const](#)** Move this to GraphicsView to make delegate stateless.

**Member [KDGantt::ItemDelegate::defaultPen\(ItemType type\) const](#)** Move this to GraphicsView to make delegate stateless.

**Member [KDGantt::ItemDelegate::paintConstraintItem\(QPainter \\*p, const QStyleOptionGraphicsItem &opt, const QF](#)**  
Review *opt*'s type

**Member [KDGantt::ItemDelegate::setDefaultBrush\(ItemType type, const QBrush &brush\)](#)** Move this to GraphicsView to make delegate stateless.

**Member [KDGantt::ItemDelegate::setDefaultPen\(ItemType type, const QPen &pen\)](#)** Move this to GraphicsView to make delegate stateless.