The Qt 4 Item Views

"Interview"

Marius Bugge Monsen,

mmonsen@trolltech.com

The Qt 4 Item Views

Contents:

- A Brief Introduction To MVC
- The Qt 4 Item Views Concepts
- The Qt 4 Item Views Architecture
- Examples (Code)

Model-View-Controller

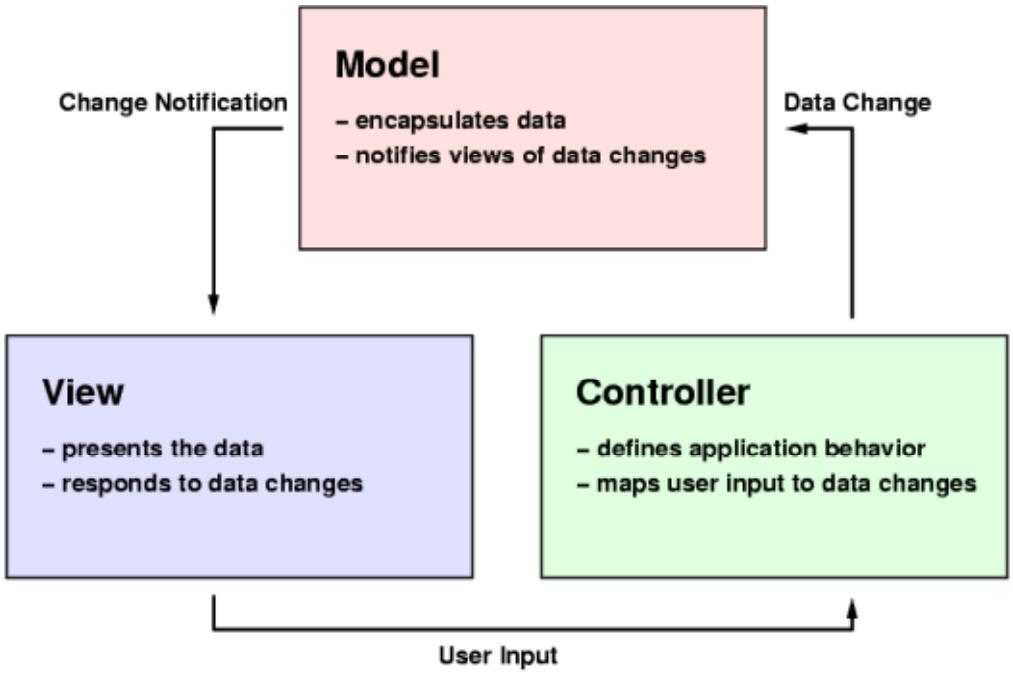
A Brief Introduction To MVC

Model-View-Controller

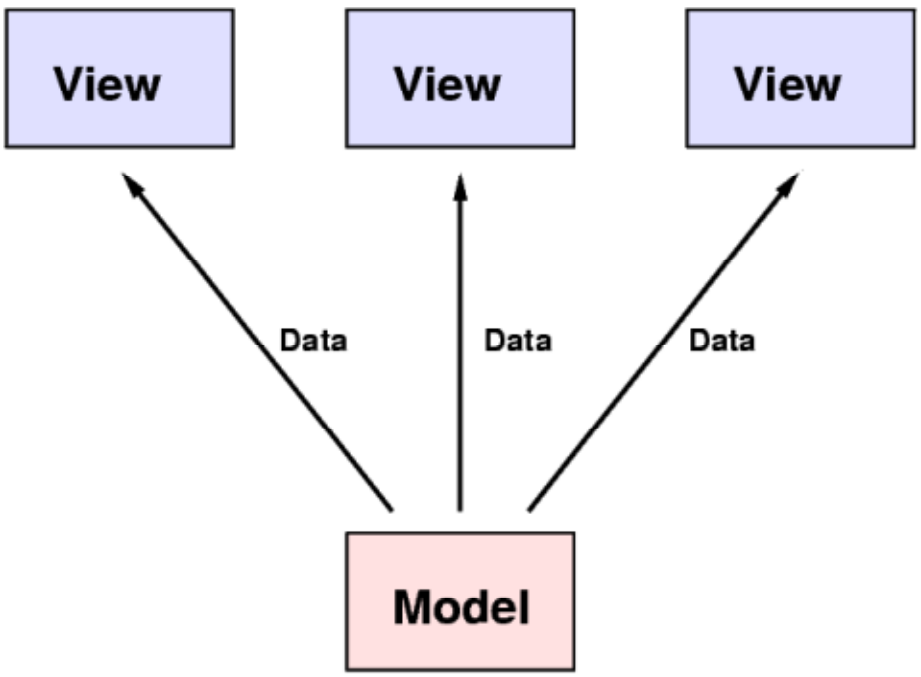
"MVC consist of three kinds of objects. The Model is the application object, the View is its screen representation, and the Controller defines the way the user interface reacts to user input. Before MVC, user interface designes tended to lump these objects together. MVC decouples them to increase flexibility and reuse."

[Design Patterns, Gamma et al.]

Model-View-Controller



Model-View-Controller



Model-View-Controller

"We quickly discovered that this split didn't work well in practical terms because the view and controller parts of a component required a tight coupling."

[http://java.sun.com/products/jfc/tsc/articles/architecture/]

Concepts

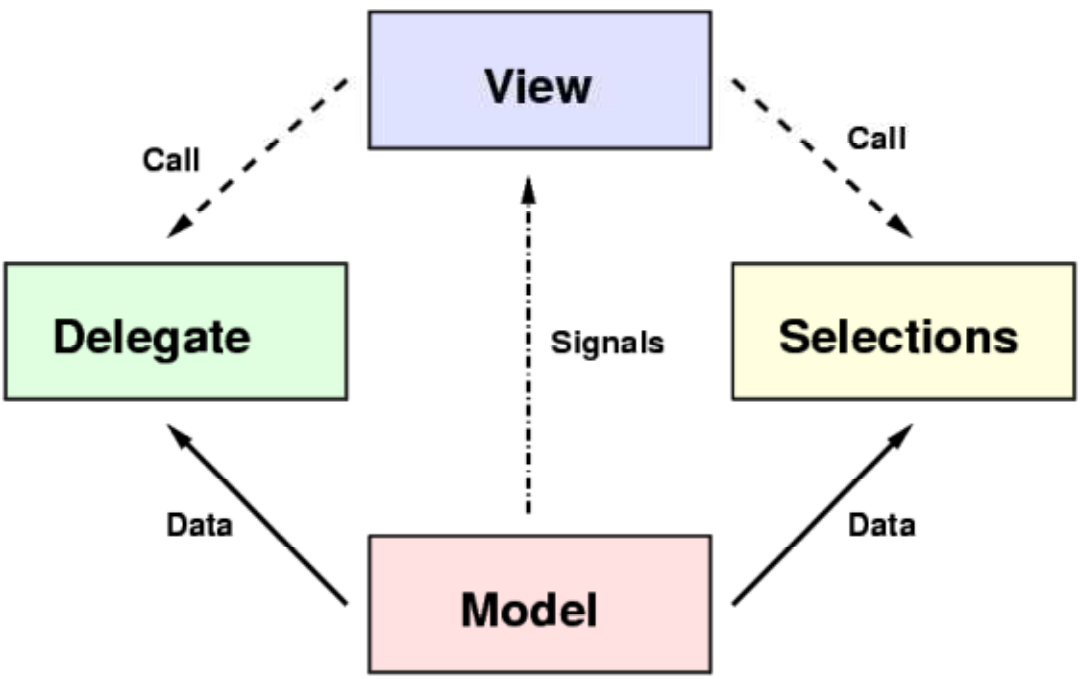
The Qt 4 Item Views Concepts

Concepts

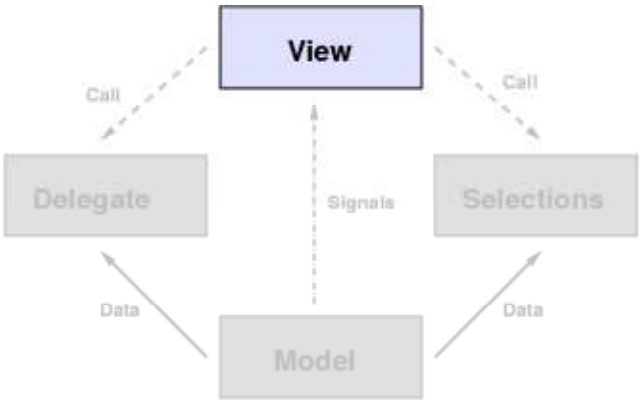
The Qt 4 Item View Components:

- Model
- View
- Delegate
- Selections

Concepts



Concepts

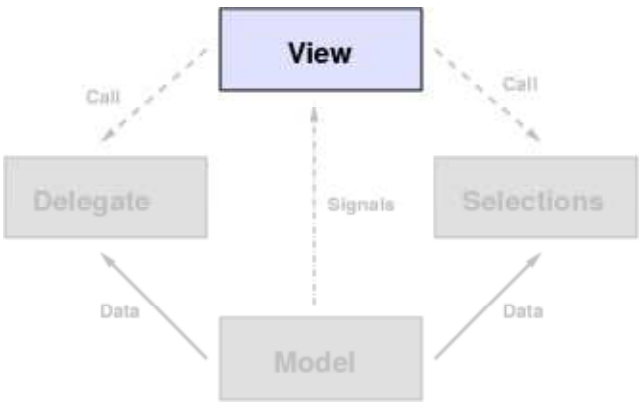


The View:

- Handles user input
- Does item layout
- Calls **Delegate** for drawing and editing the items

1/2

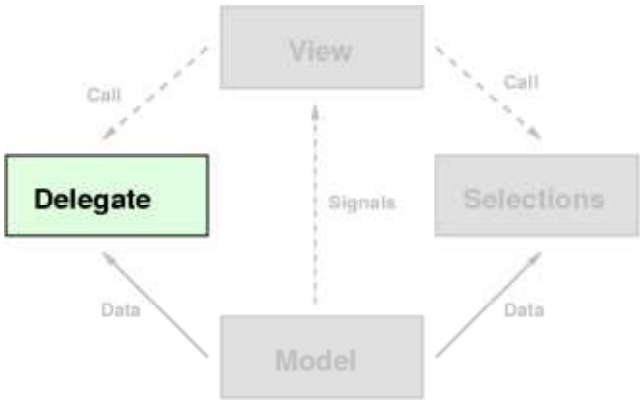
Concepts



- Calls **Selections** to do selection of items
- Implements view specific behavior

2/2

Concepts

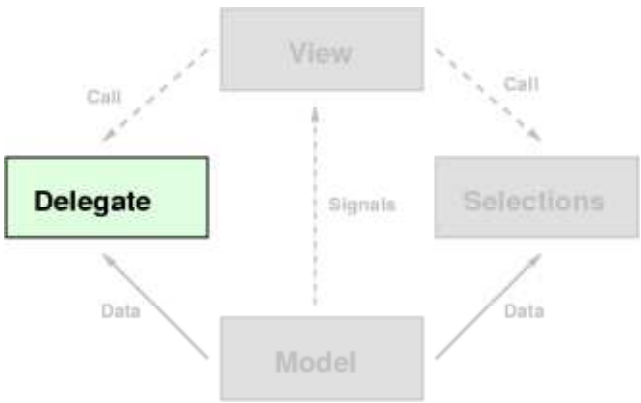


The Delegate:

- Paints each item
- Performs internal layout of each item
- Gives the size hint for each item

1/2

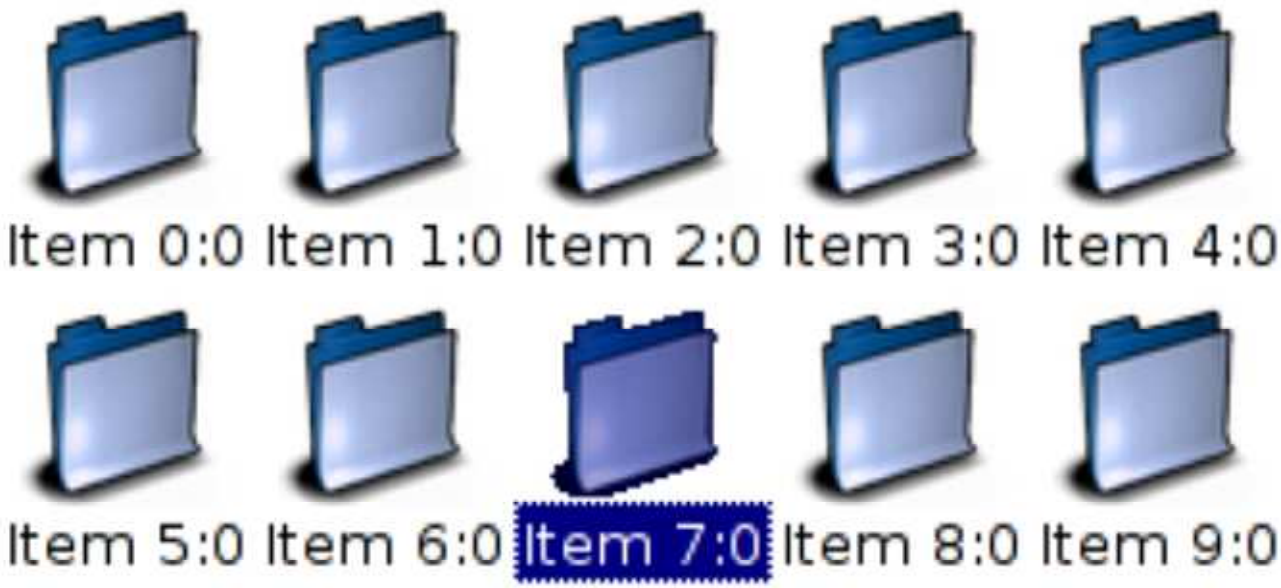
Concepts



- Provides an editor for each item
- Manages the editor geometry for each item

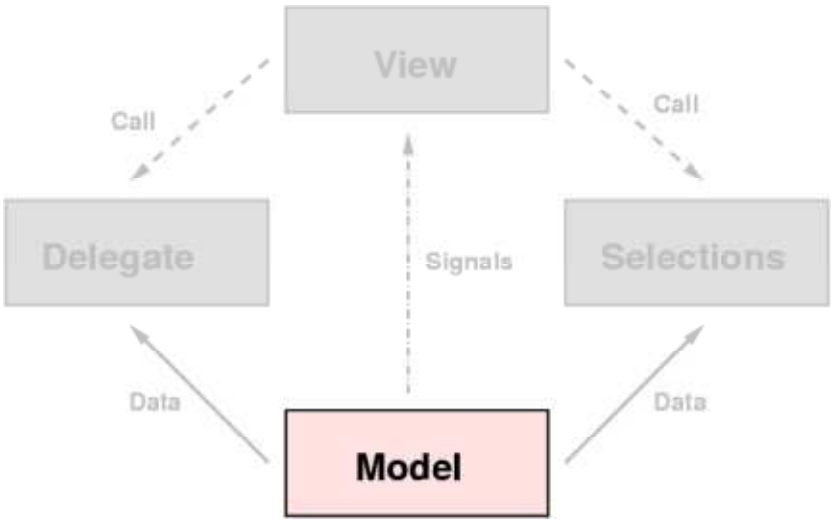
2/2

Concepts



The delegate is responsible for rendering and editing single items.

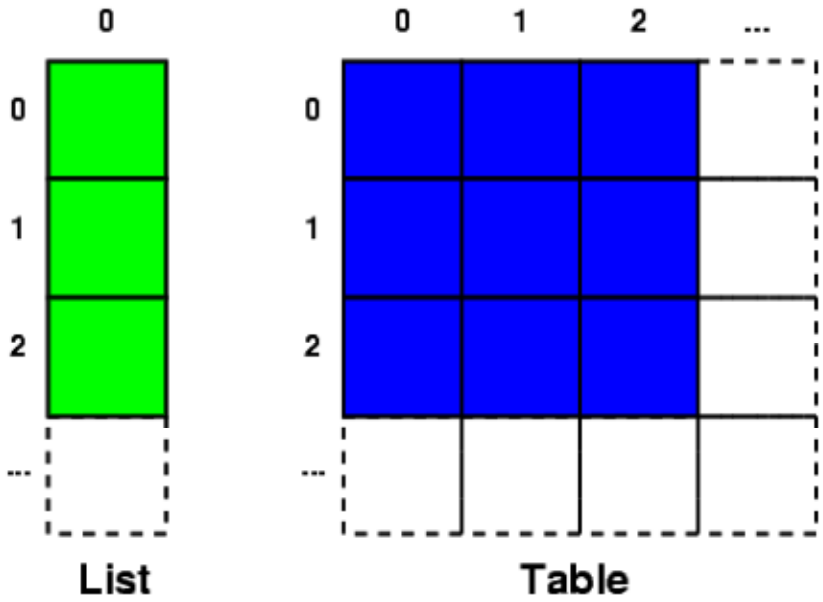
Concepts



The Model:

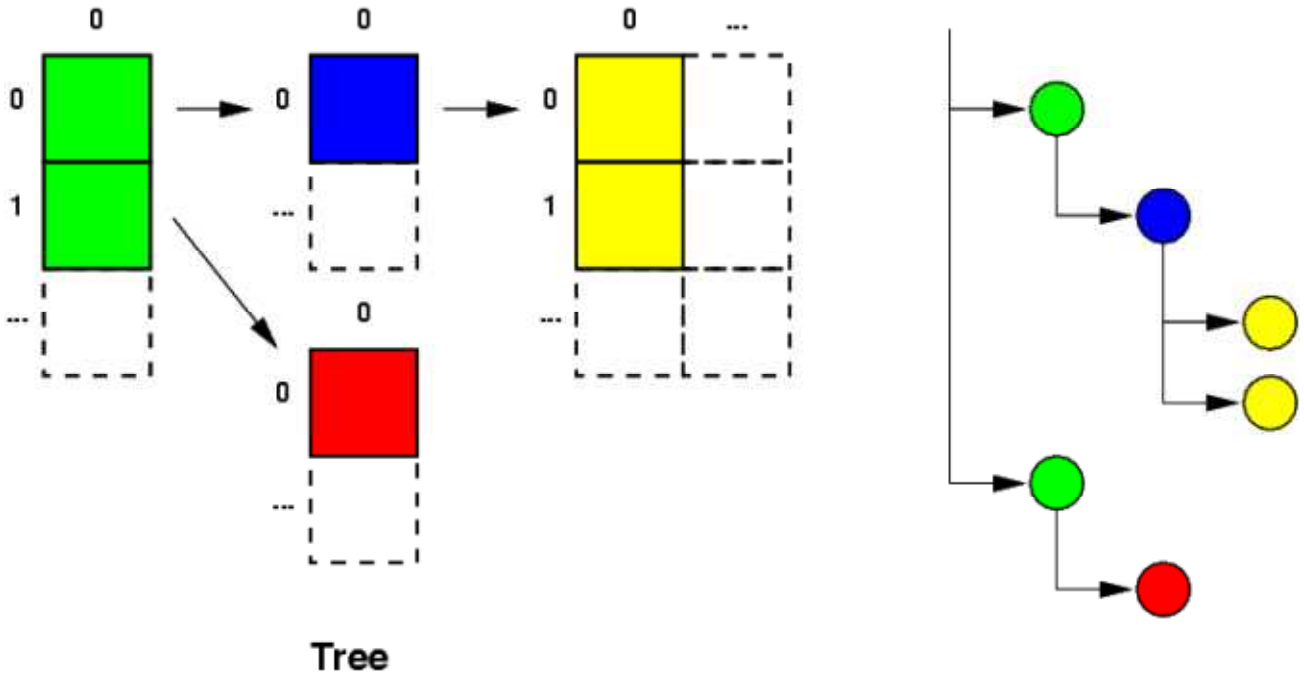
- Manages the Data
- Gives notice to the **View** of changes in the data

Concepts



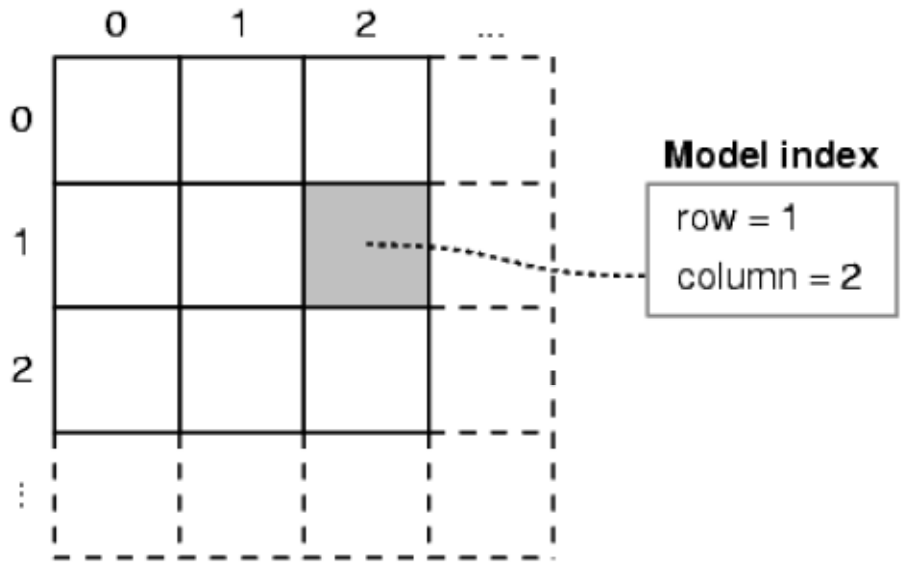
A "flat" model presents the data as if it was in a **table.**

Concepts



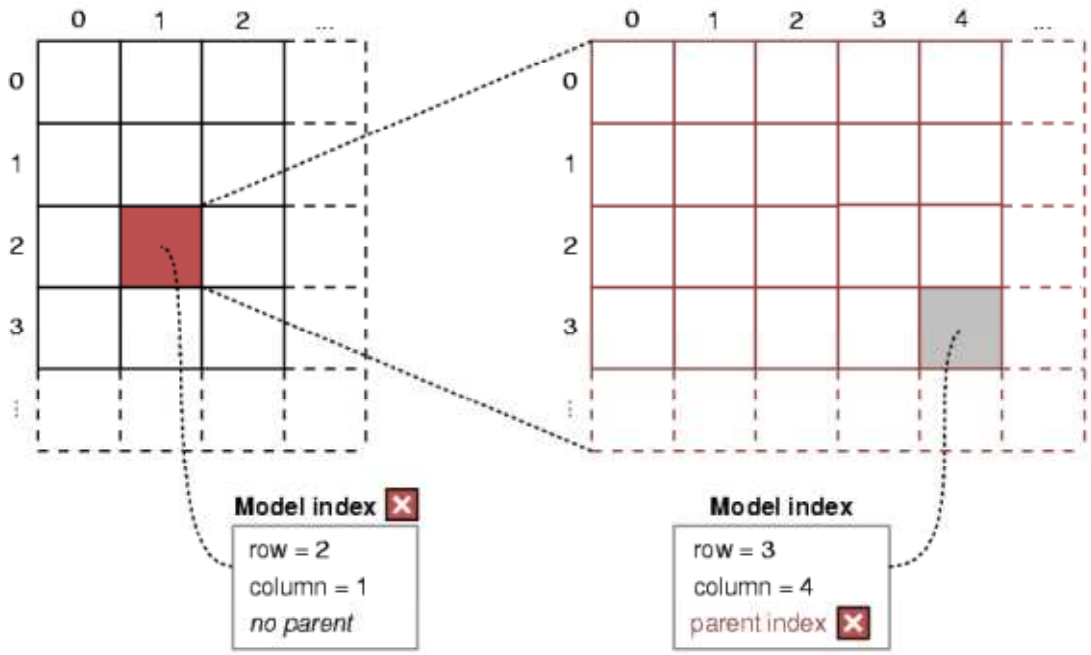
A **hierarchical** model presents a hierarchy of **tables**.

Concepts



The **Model Index** refers to a **cell** in the model.

Concepts



Concepts

The Model Data Roles:

- Each **item** in a table can have several data **elements**
- Each data **element** is assigned a **role**

Concepts

The Predefined Roles:

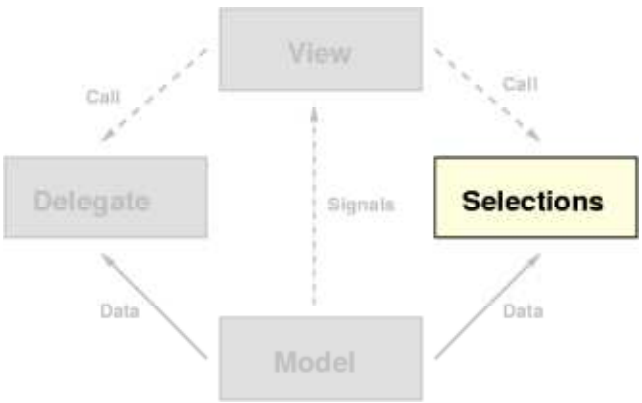
- Display *(typically the item text)*
- Decoration *(typically the item pixmap)*
- Edit *(shown in the editor)*

Concepts

- Tool Tip
- Status Tip
- What's This?

2/2

Concepts

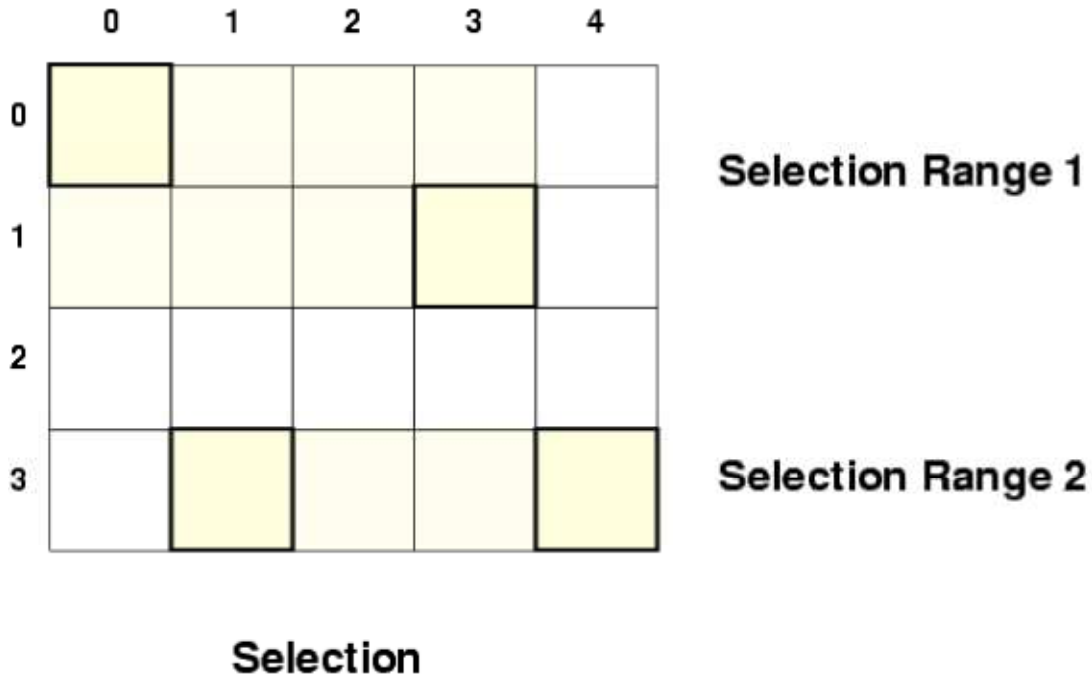


The Selections:

- Manage selections of items
- Manage current item

Gives notice to the **View** of changes in selections or current item

Concepts

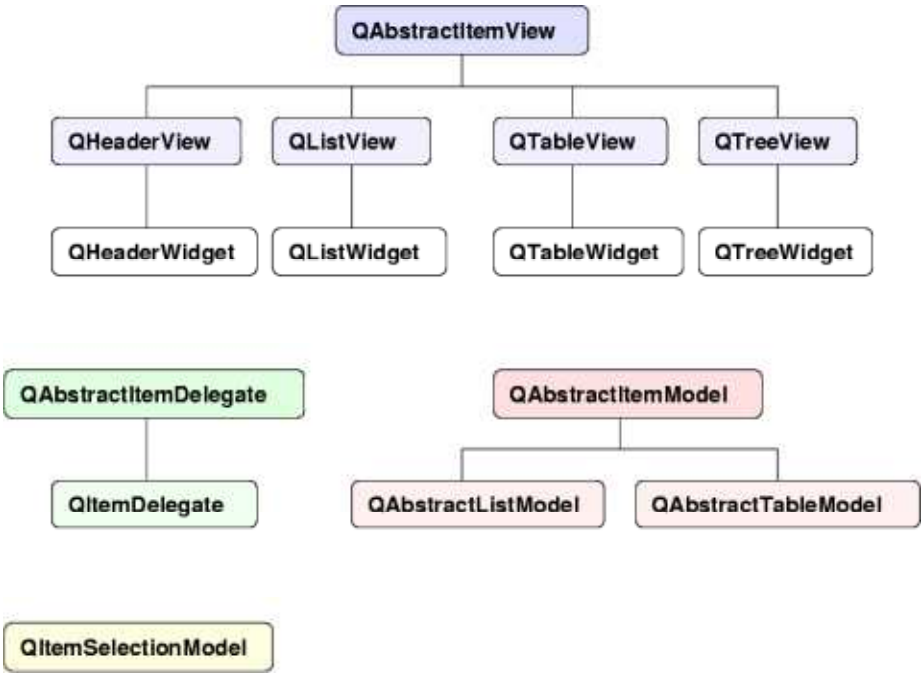


Selections are defined by **model indexes.**

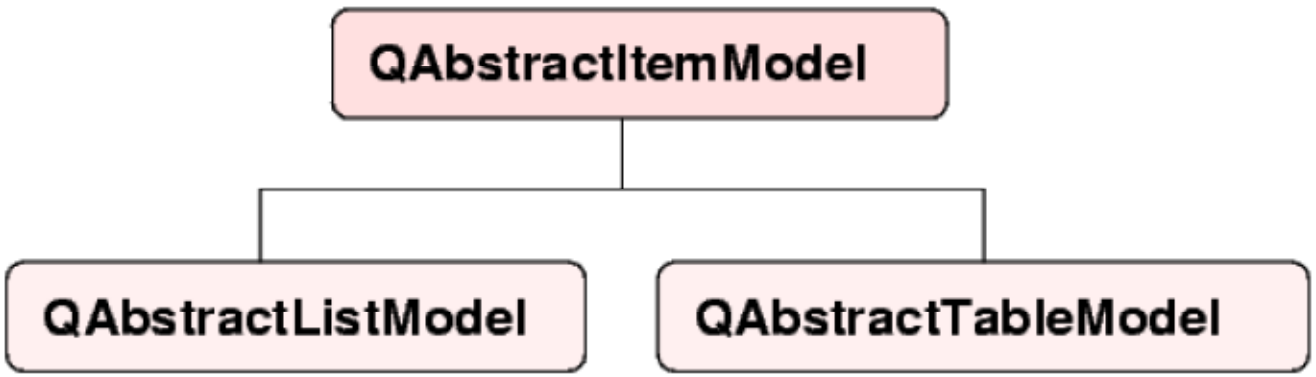
Architecture

The Qt 4 Item Views Architecture

Architecture



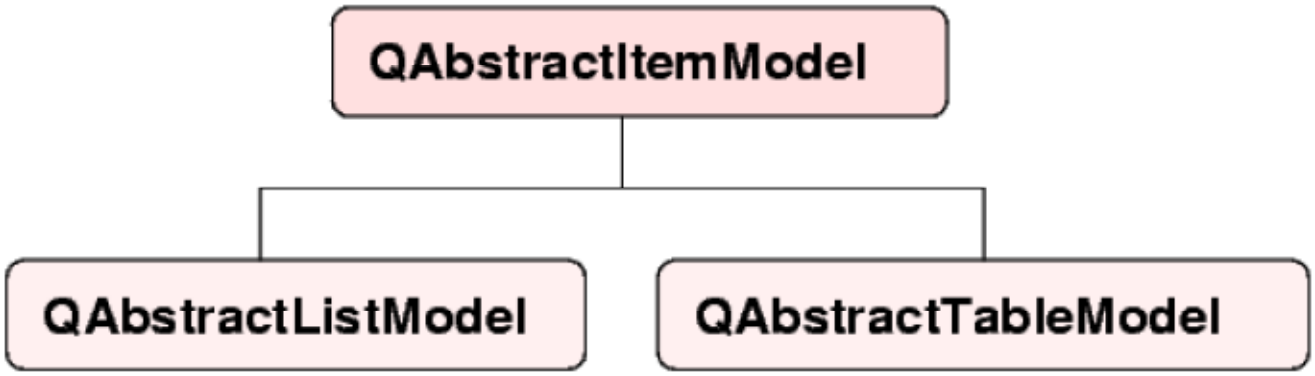
Architecture



The model encapsulates the data.

- QAbstractItemModel
  - Hierarchy
- QAbstractListModel

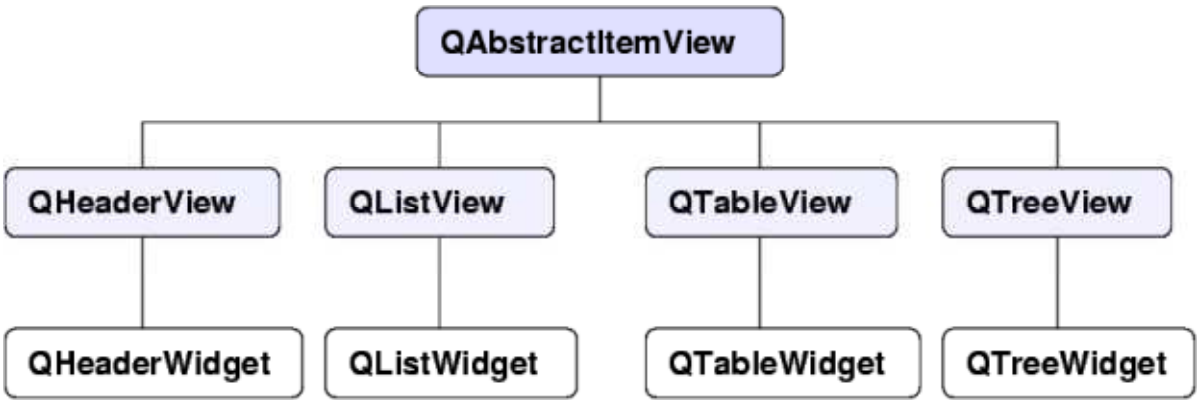
Architecture



- Single column *(list)*
- QAbstractTableModel
  - Rows and columns *(table)*

2/2

Architecture



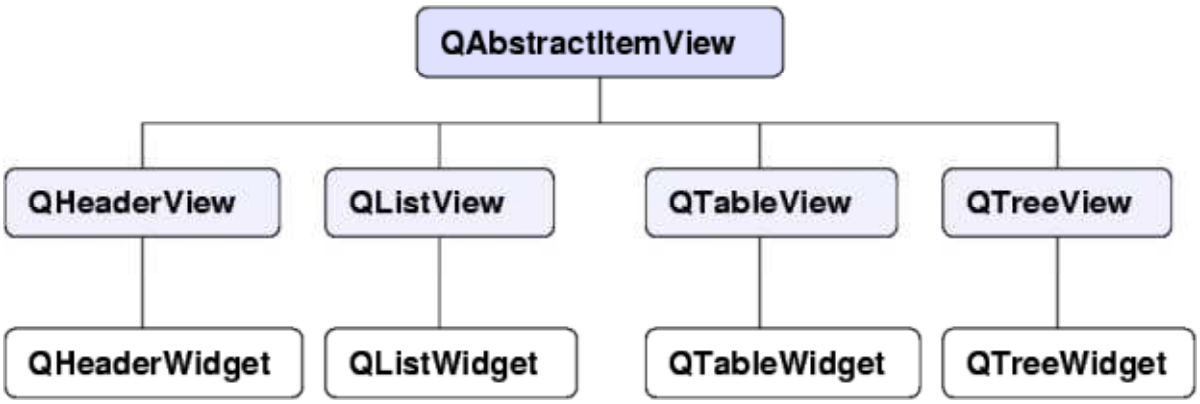
The view presents the data.

Three levels:

- Abstract
  - Common view functionality

1/3

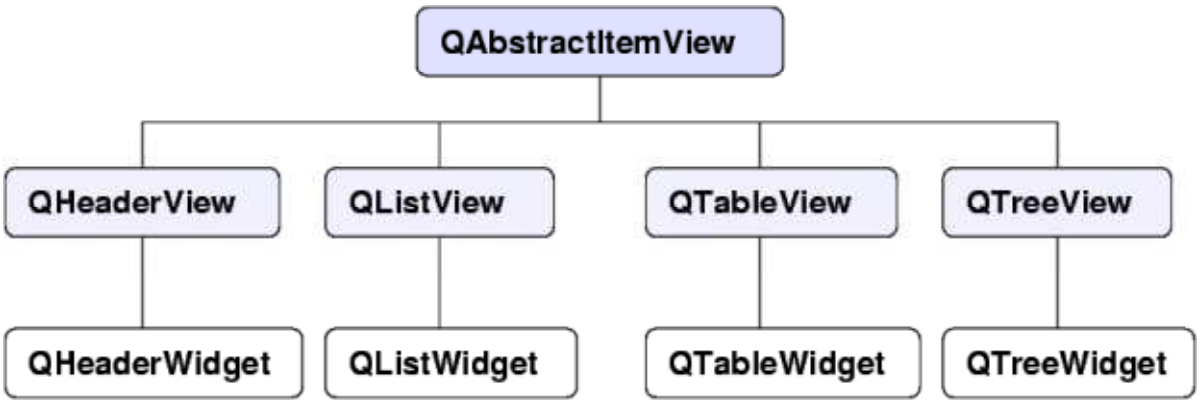
Architecture



- Generic
  - Uses Model/View
- Convenience
  - View items
  - Internal model

2/3

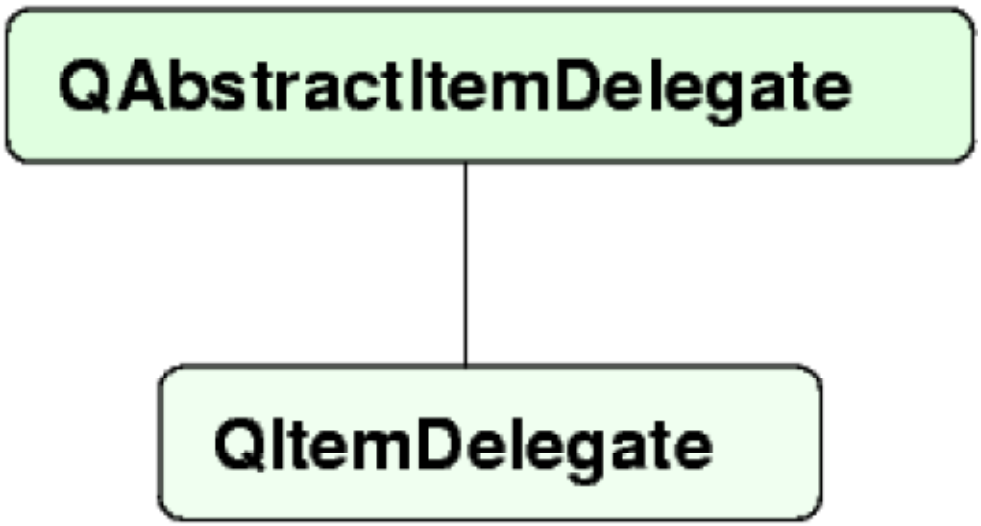
Architecture



- **No** Model/View

3/3

Architecture



The delegate is responsible for rendering and editing single items.

- QItemDelegate is the **default** delegate in the views

Architecture

The selection model stores the selections.

- Current item
- Current selection
- Selection ranges

Examples

Examples (Code)

Examples

An example of how to use the QTableWidget convenience view:

```
int main(int argc, char *argv[])
{
    QApplication app(argc, argv);
    QTableWidget *table = new QTableWidget(page);
    table->setRowCount(1000);
    table->setColumnCount(1000);

    QTableWidgetItem *item = new QTableWidgetItem();
```

Examples

```
    item->setBackgroundColor(Qt::blue);
    item->setTextColor(Qt::red);
    item->setText("Hello");
    item->setFont(QFont("Helvetica", 12, QFont::Bold));
    item->setIcon(QPixmap("data/services.png"));
    table->setItem(1, 1, item);
    table->show();

    app.setMainWidget(table);
    return app.exec();
}
```

Examples

$HOME/examples/convenience/main.cpp

Examples

How to implement a minimal model:

```
class MiniModel : public QAbstractListModel
{
    Q_OBJECT
public:
    MiniModel(QObject *parent = 0);
    ~MiniModel();

    int rowCount() const;
    QVariant data(const QModelIndex &index, int role) const;
```

Examples

```
};
```

$HOME/examples/minimal/minimodel.h

Examples

```
MiniModel::MiniModel(QObject *parent)
    : QAbstractListModel(parent)
{}

MiniModel::~MiniModel()
{}

int MiniModel::rowCount() const
{
    return 10;
}
```

1/2

Examples

```
QVariant MiniModel::data(const QModelIndex &index, int role) const
{
    if (role == DisplayRole)
        return "Item " + QString::number(index.row());
    return QVariant();
}
```

$HOME/examples/minimal/minimodel.cpp

2/2

Examples

How to view a model in several views, and share a selection model:

```
...
QDirModel *model = QDirModel(QDir::home(), splitter);


QTreeView *tree = new QTreeView(splitter);
QListView *list = new QListView(splitter);


tree->setModel(model);
list->setModel(model);
```

1/2

Examples

```
QItemSelectionModel *selection = new QItemSelectionModel(model,
splitter);


tree->setSelectionModel(selection);
list->setSelectionModel(selection);
...
```

$QTDIR/doc/src/snippets/shareddirmodel/main.cpp

2/2

Examples

A simple table model:

```
class PlasmaModel : public QAbstractTableModel
{
public:
    PlasmaModel(int rows, int cols, QObject *parent = 0);
    ~PlasmaModel();

    int rowCount() const;
    int columnCount() const;
```

1/2

Examples

```
    QVariant data(const QModelIndex &index, int role) const;

protected:
    void timerEvent(QTimerEvent *e);

private:
   ... // private data is not shown ...
};
```

$QTDIR/demos/interview/plasmatable/plasmamodel.h

2/2

Examples

```
int PlasmaModel::rowCount() const
{
    return rows;
}

int PlasmaModel::columnCount() const
{
    return cols;
}
```

1/2

Examples

$QTDIR/demos/interview/plasmatable/plasmamodel.cpp

2/2

Examples

```
QVariant PlasmaModel::data(const QModelIndex &index, int role)
const
{
    if (role == DisplayRole)
        return value(index.row(), index.column()); // return the
color value
    return QVariant();
}
```

$QTDIR/demos/interview/plasmatable/plasmamodel.cpp

Examples

```
void PlasmaModel::timerEvent(QTimerEvent *e)
{
    if (e->timerId() != timer)
        return;

... // calculating new colors here

    QModelIndex topLeft = index(0, 0);
    QModelIndex bottomRight = index(rows - 1, cols - 1);
```

Examples

```
    emit dataChanged(topLeft, bottomRight); // this is important!
}
```

$QTDIR/demos/interview/plasmatable/plasmamodel.cpp

2/2

Examples

A simple delegate:

```
class PlasmaDelegate : public QAbstractItemDelegate
{
public:
    PlasmaDelegate(QObject *parent = 0);
    ~PlasmaDelegate();

    void paint(QPainter *painter,
               const QStyleOptionViewItem &option,
               const QAbstractItemModel *model,
```

1/2

Examples

```
                const QModelIndex &index) const;


    QSize sizeHint(const QFontMetrics &fontMetrics,
                   const QStyleOptionViewItem &option,
                   const QAbstractItemModel *model,
                   const QModelIndex &index) const;
};
```

$QTDIR/demos/interview/plasmatable/plasmadelegate.h

2/2

Examples

```
void PlasmaDelegate::paint(QPainter *painter,
                           const QStyleOptionViewItem &option,
                           const QAbstractItemModel *model,
                           const QModelIndex &index) const
{
    unsigned char s = option.state & QStyle::Style_Selected ? 63 :
0;
    unsigned int color = model->data(index,
QAbstractItemModel::DisplayRole).toInt();
    unsigned char r = ((color & 0x00FF0000) >> 16) + s;
    unsigned char g = ((color & 0x0000FF00) >> 8) + s;
```

1/2

Examples

```
    unsigned char b = (color & 0x000000FF) + s;

    painter->setPen(Qt::NoPen);
    painter->setBrush(QColor(r, g, b));
    painter->drawRect(option.rect);
}
```

$QTDIR/demos/interview/plasmatable/plasmadelegate.cpp

2/2

Examples

```
QSize PlasmaDelegate::sizeHint(const QFontMetrics &,
                               const QStyleOptionViewItem &,
                               const QAbstractItemModel *,
                               const QModelIndex &) const
{
    return QSize(4, 4); // our cell size
}
```

$QTDIR/demos/interview/plasmatable/plasmadelegate.cpp

Thank you!