Qt Item Views In Depth

About me

- Marius Bugge Monsen
- Senior Software Engineer
- 5 years for TT
- Msc. comp. sci. NTNU
- Current projects:
  - Itemviews
  - QtDBus
  - WebKit

TROLLTECH DEV DAYS '07

Contents

- Introduction
- Architecture
- Models In Depth
- Selections In Depth
- I Just Want To ...

TROLLTECH DEV DAYS '07

- **Models In Depth**
  - List
  - Tree
  - Editable Items
  - Inserting And Removing Items
  - Lazy Initialization
  - Drag And Drop
  - Testing

TROLLTECH DEV DAYS '07

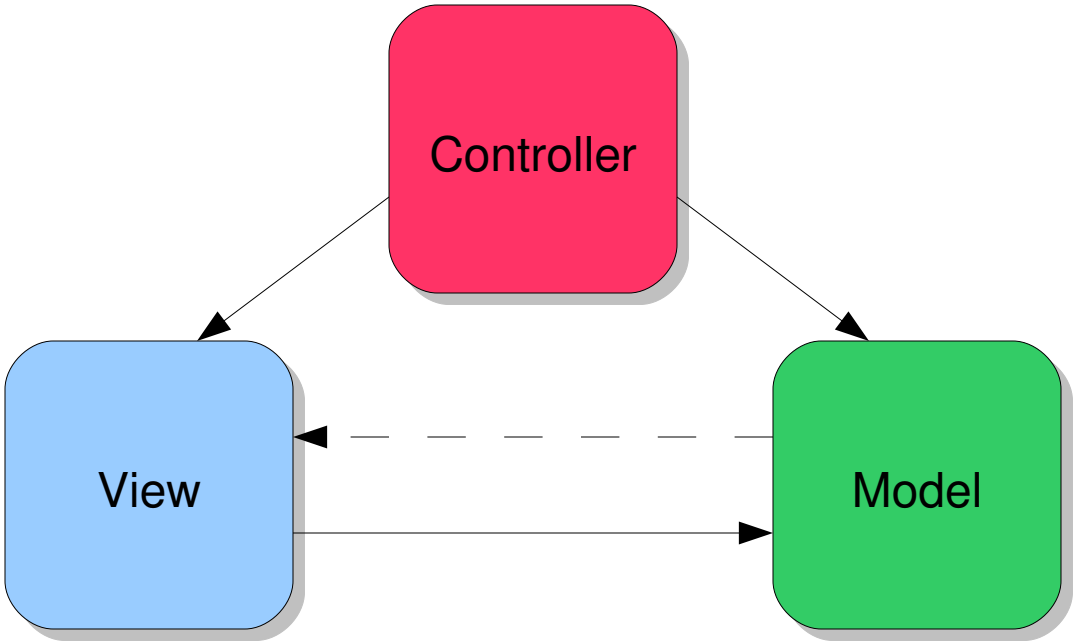
- **I Just Want To...**
  - Show Some Data
  - Insert Lots Of Items Fast
  - Do Sorting And Filtering
  - Have Checkable Items
  - Do Custom Item Painting
  - Have A Custom Item Editor
  - Animate Items

TROLLTECH DEV DAYS '07

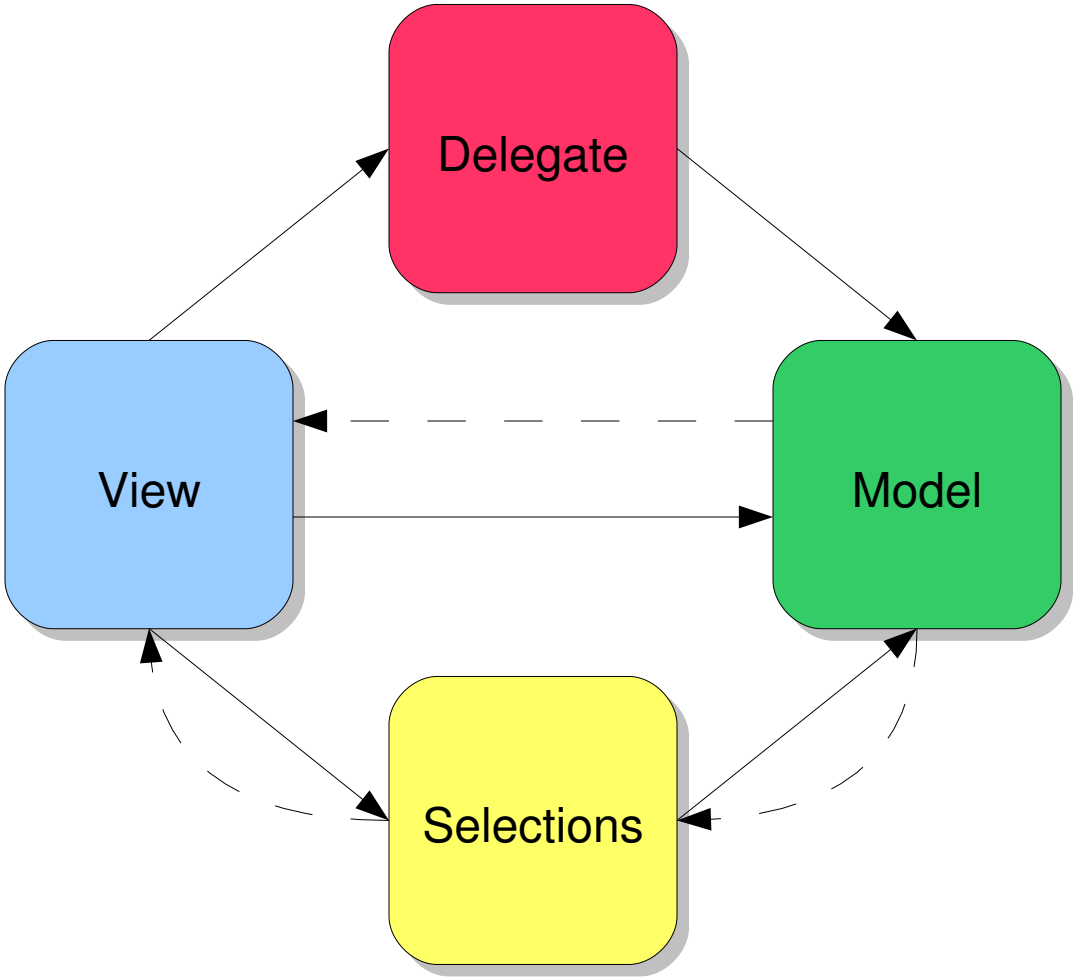
Introduction

- Model View Controller (MVC) Design Pattern
- Qt Model / View Design

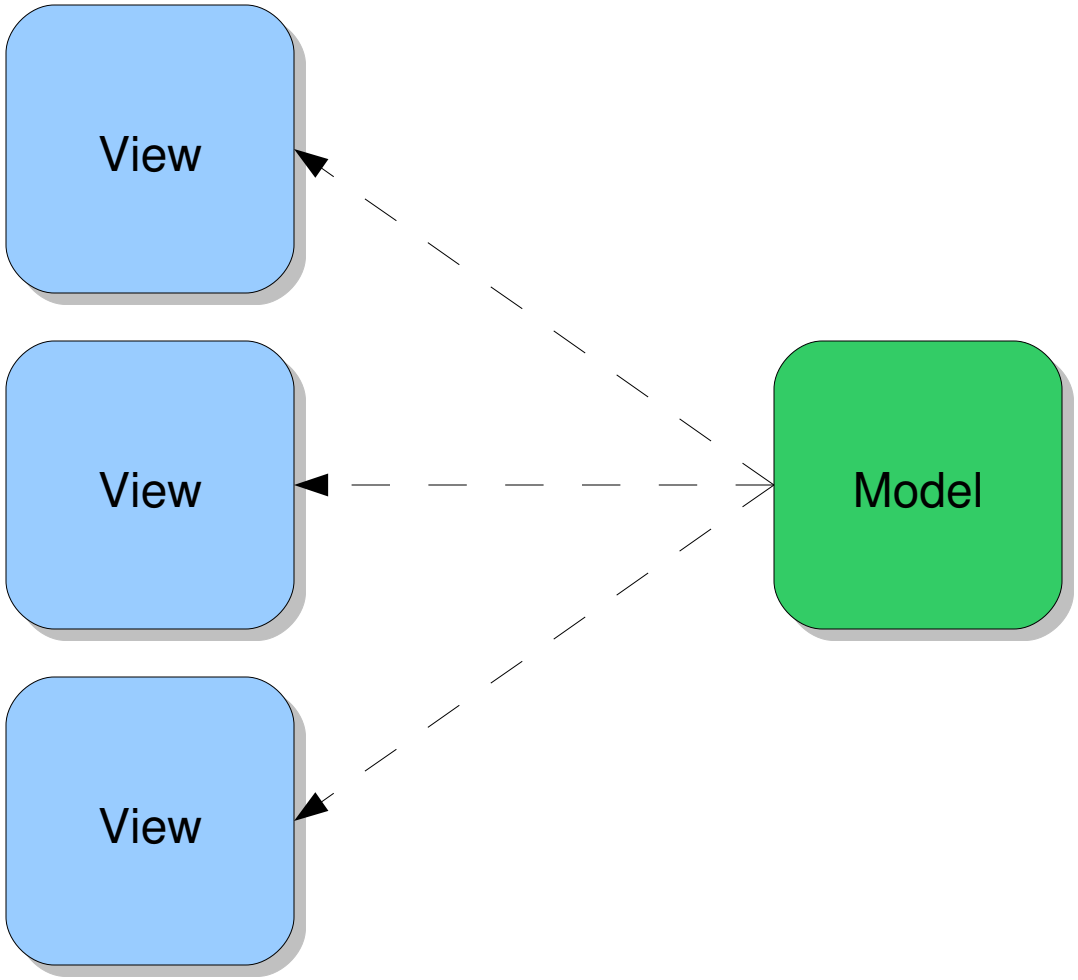
The MVC Design Pattern



The Qt Model/View Design

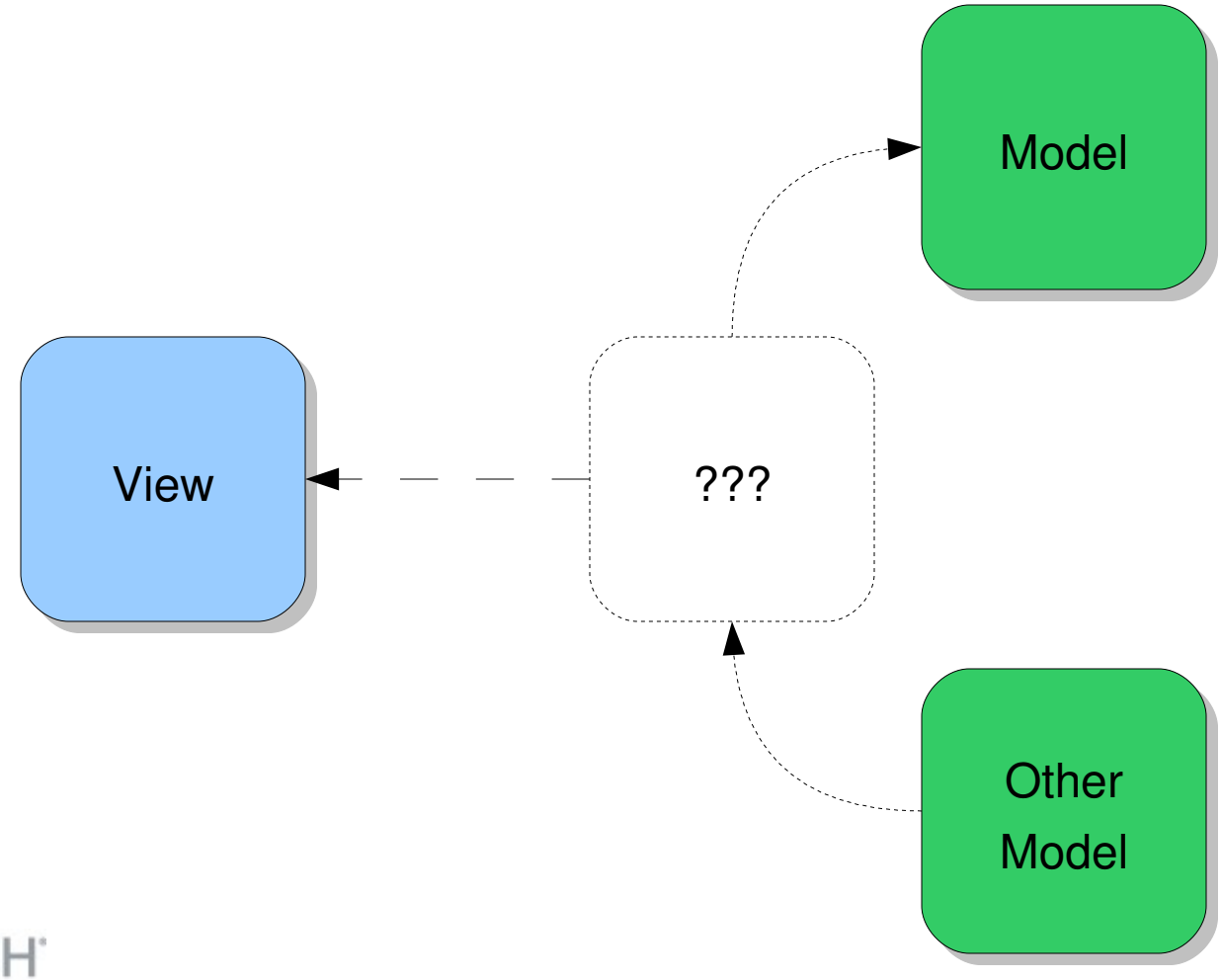


Multiple Views on a Model

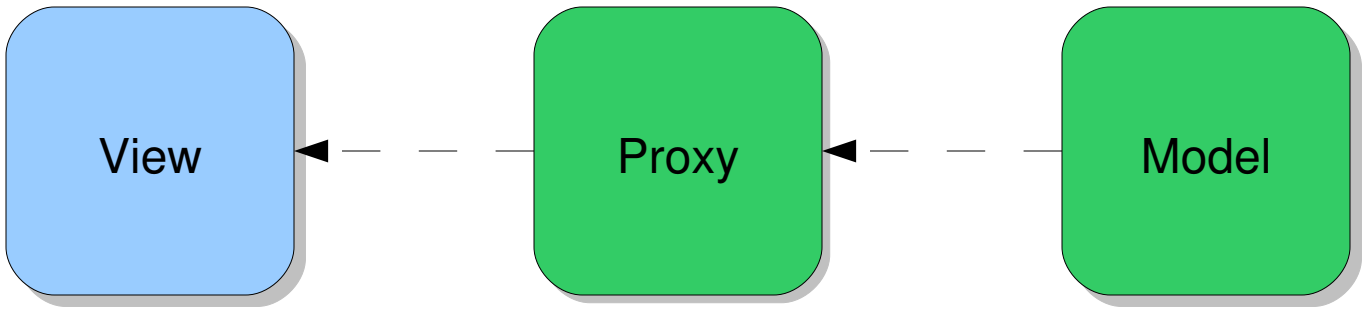


TROLLTECH DEV DAYS '07

Interchangeable Models



Chaining Models



Using Model / View

```
#include <QtGui>

int main(int argc, char *argv[])
{
    QApplication app(argc, argv);

    QStandardItemModel model(10, 10);
    QTableView view;

    view.setModel(&model);
    view.show();

    return app.exec();
}
```

TROLLTECH DEV DAYS '07

Model vs. Items

- Q3FileDialog
  - q3filedialog.cpp: 4512 loc
- QFileDialog
  - qfiledialog.cpp: 1639 loc
  - qfilesystemmodel.cpp: 1108 loc
    - total: 2747 loc

Qt 4 Item Views

- ~29000 lines of code
- 39 public classes
  - 9 view classes
  - 11 model classes

TROLLTECH DEV DAYS '07

- Introduction
- **Architecture**
- Models In Depth
- Selections In Depth
- I Just Want To ...

TROLLTECH DEV DAYS '07

Architecture

- Goals
- Separating Data And Presentation
- Designing An Interface
- Look Ma, No Items!

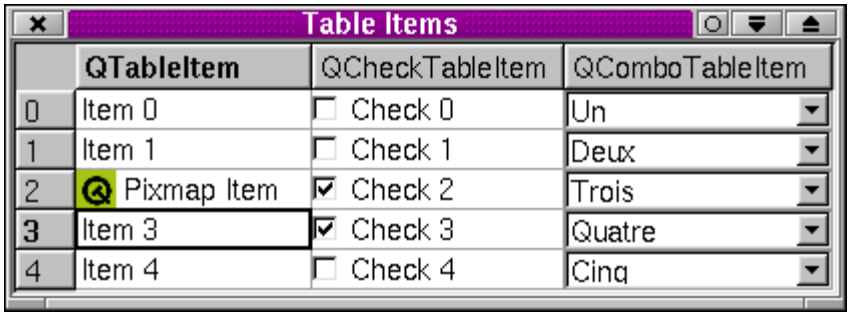
TROLLTECH DEV DAYS '07

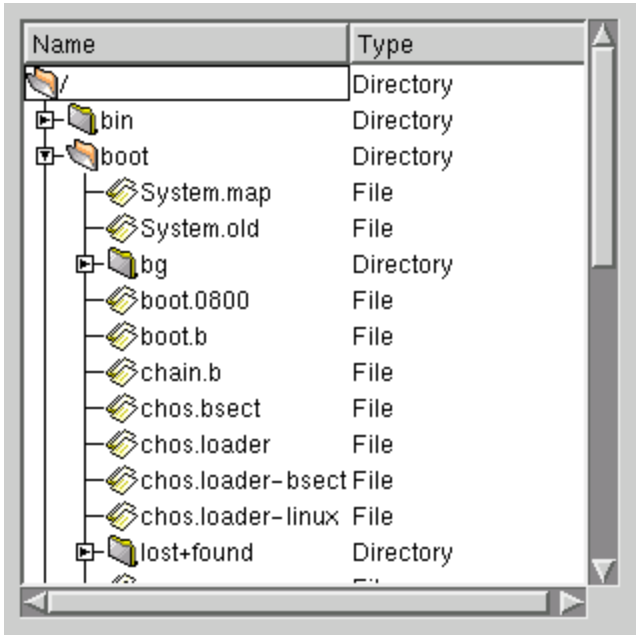
History

- Goals
  - Separate data from the view
  - Handle a large number of items
  - Only care about the data we show
  - Small memory footprint
  - ...
  - Profit!!!

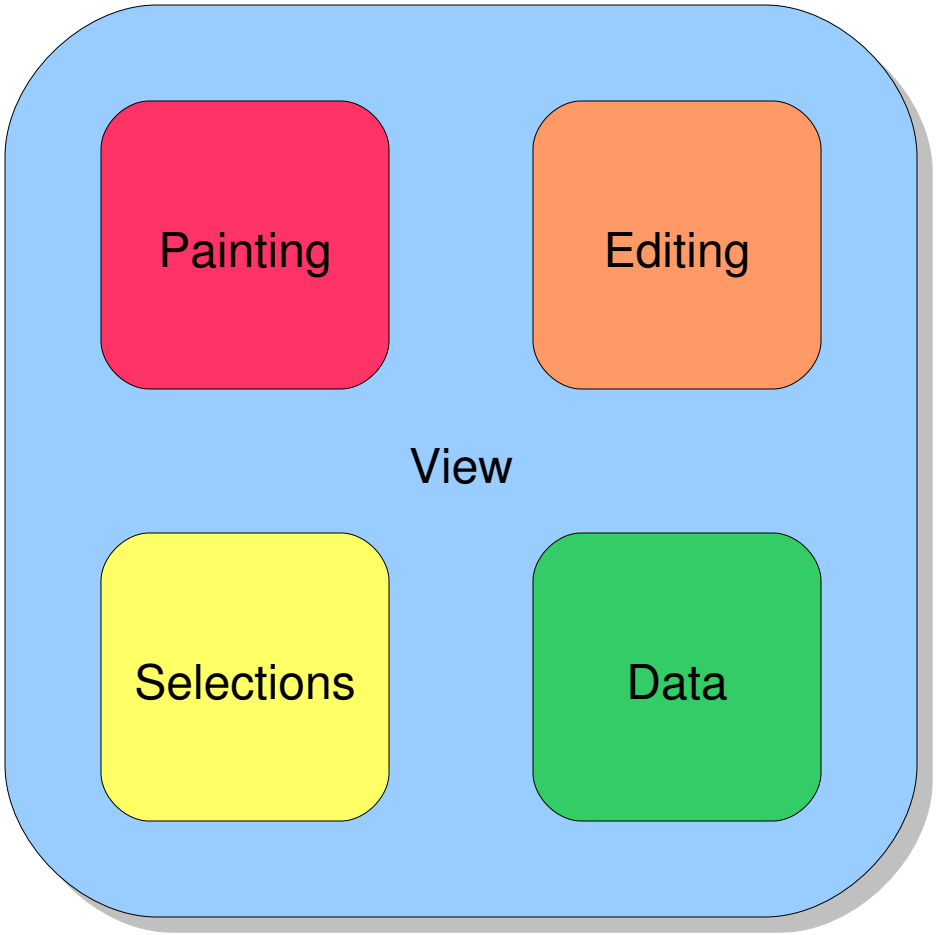
TROLLTECH DEV DAYS '07

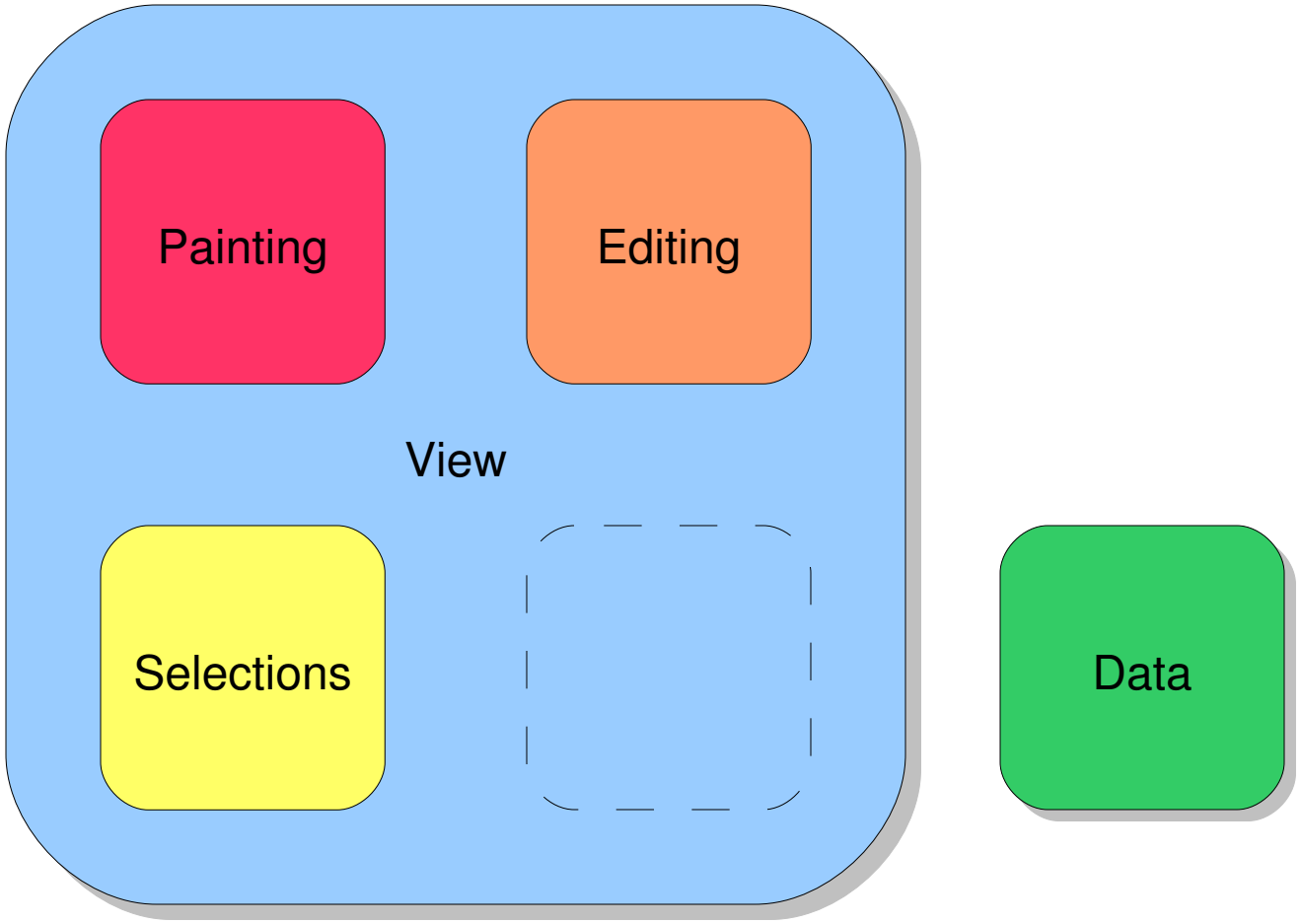
Qt 3 Item Views

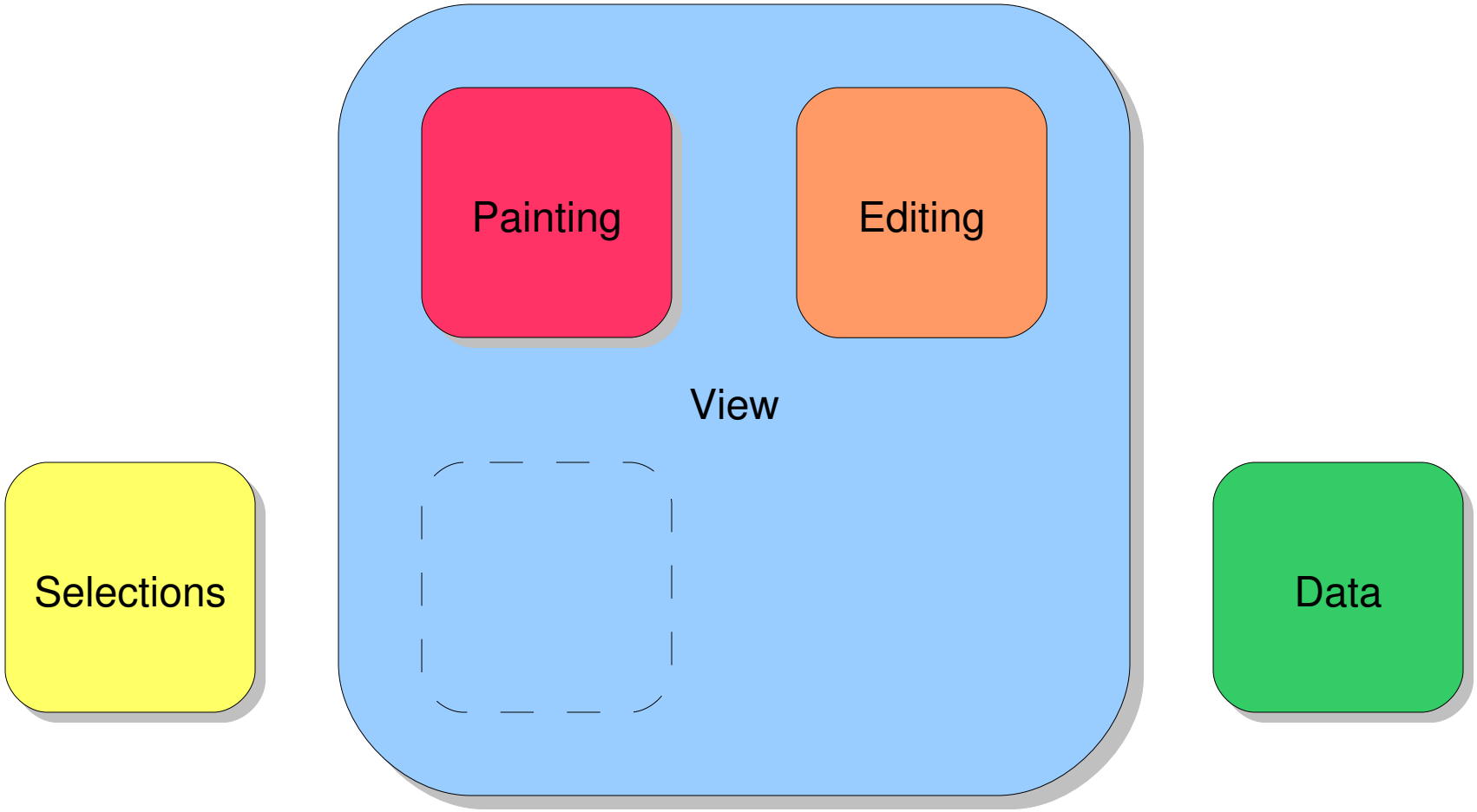


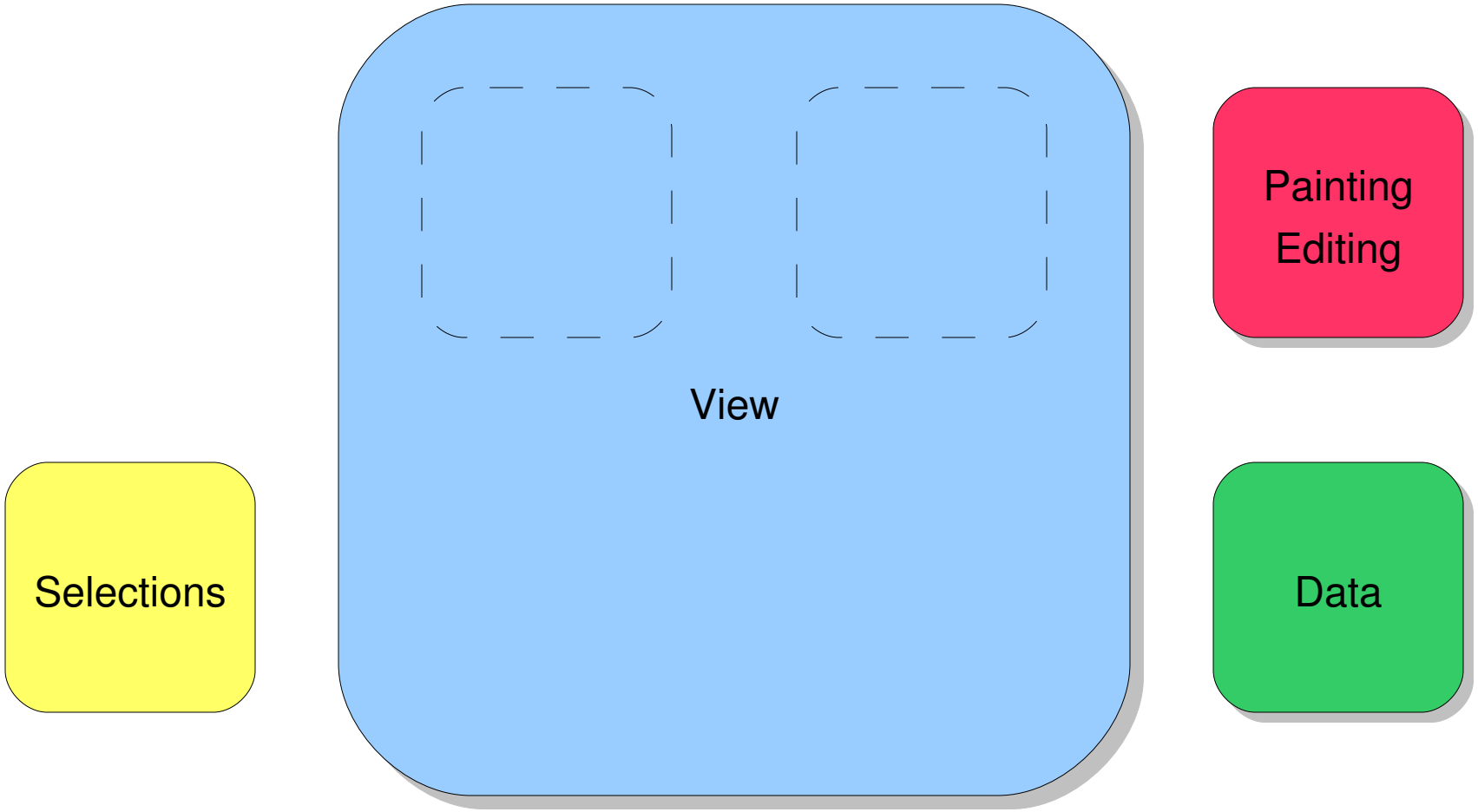


Qt 3 Item Views

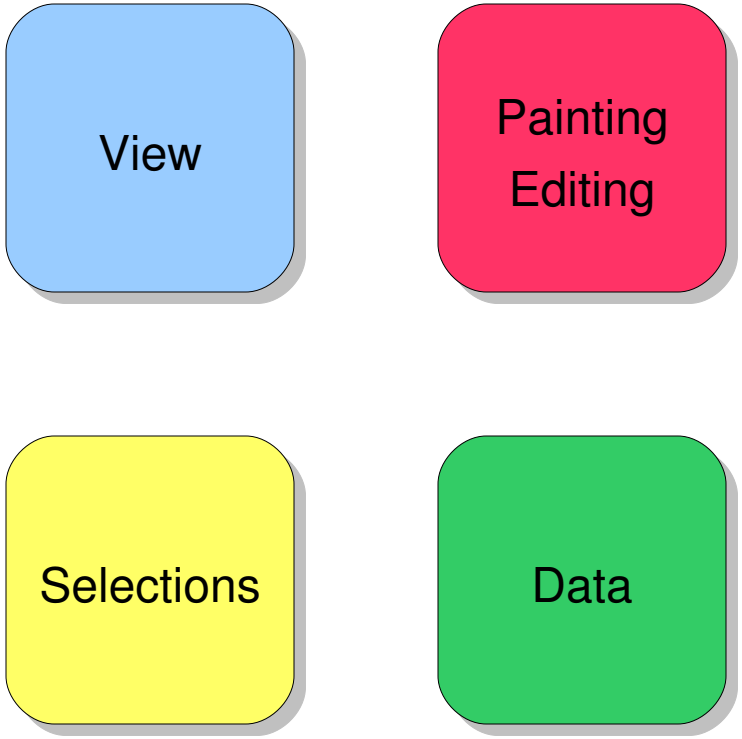



Painting
Editing
View
Selections
Data


Painting
Editing
View
Selections
Data

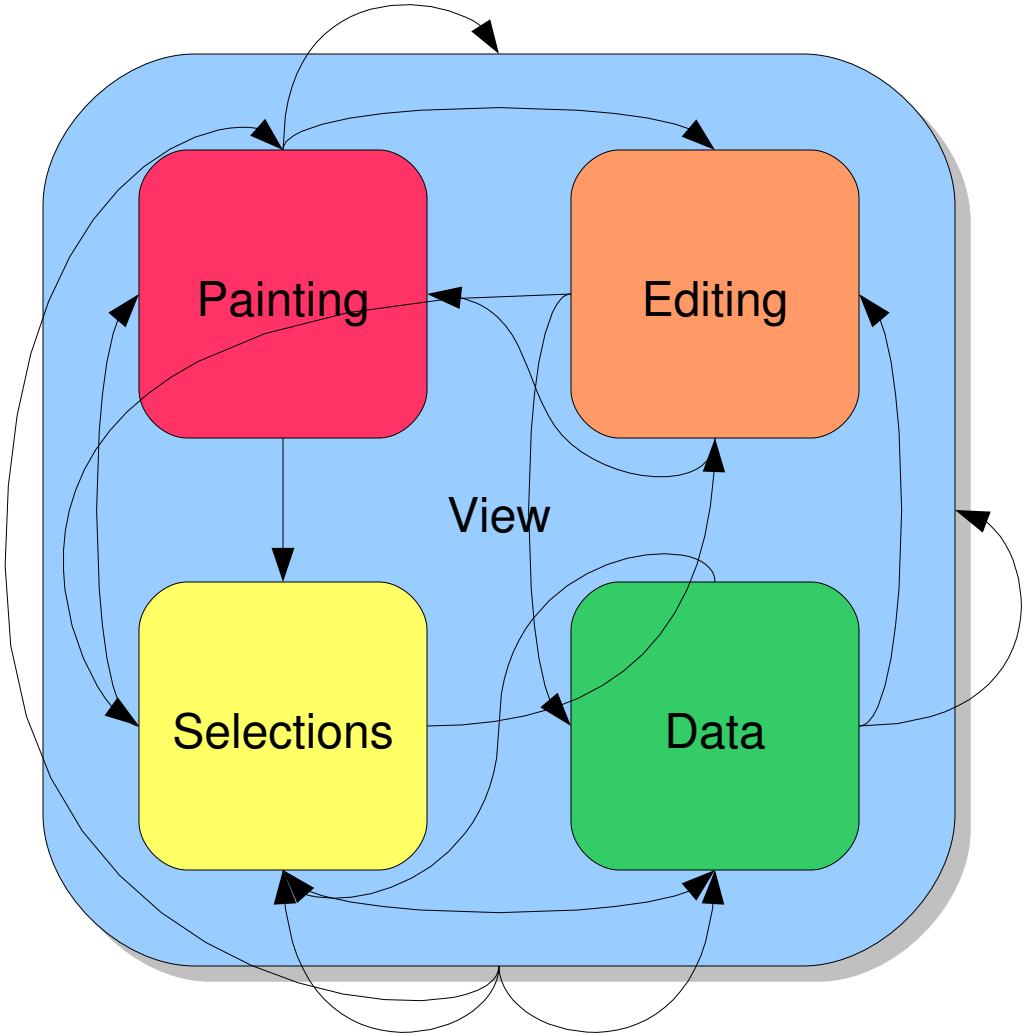

View
Painting
Editing
Selections
Data

Qt 4 Item Views

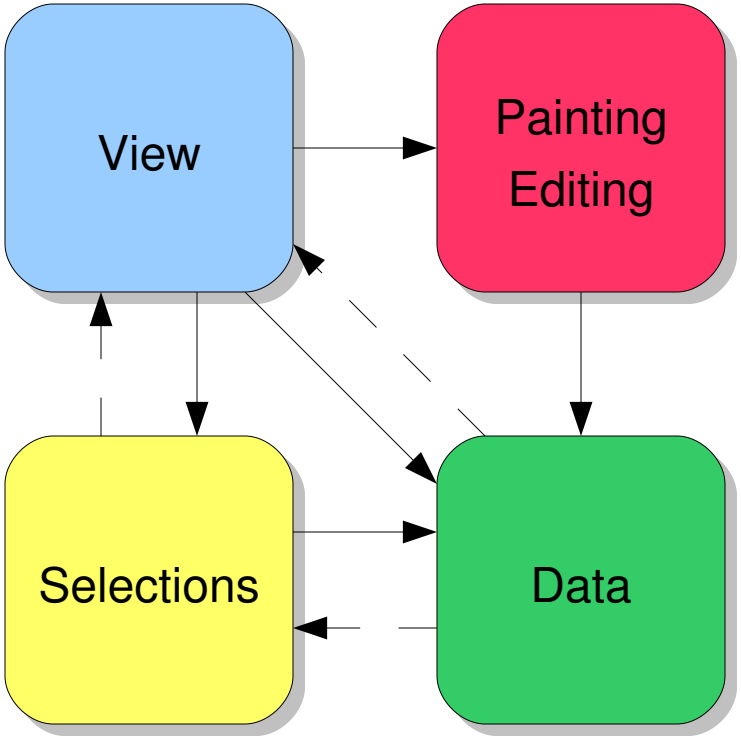


TROLLTECH DEV DAYS '07

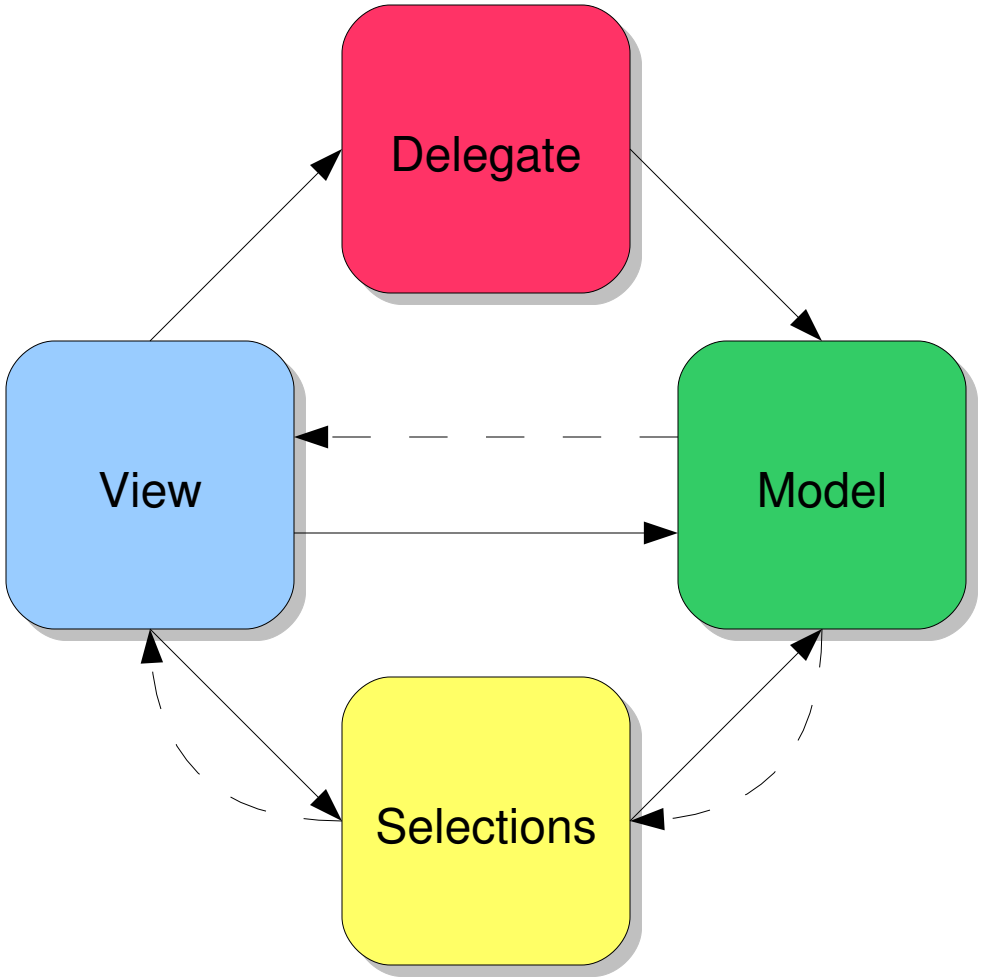
Qt 3 Item Views



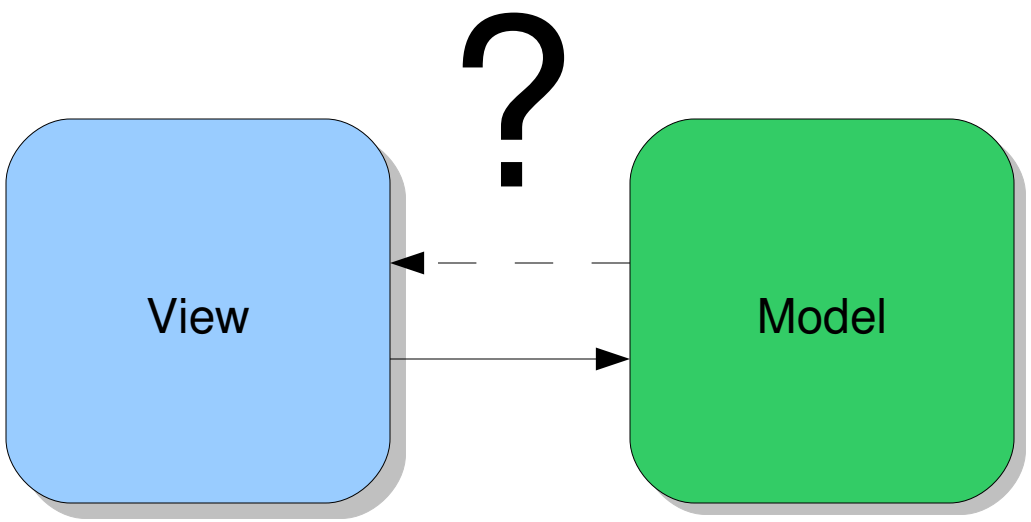
Qt 4 Item Views



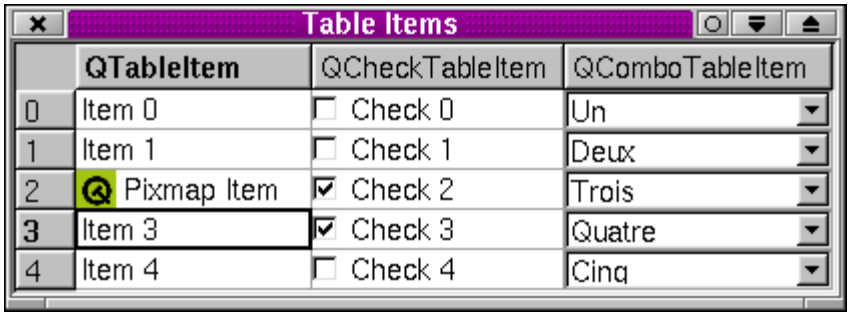
Qt 4 Item Views

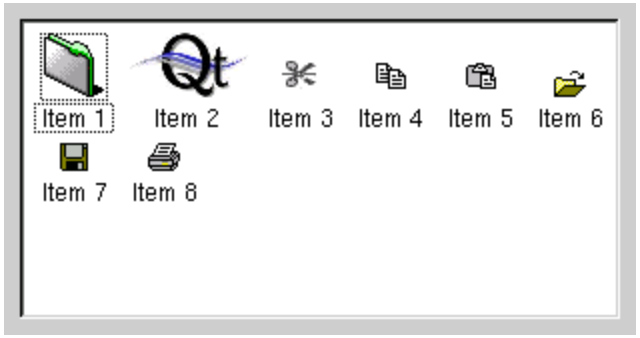


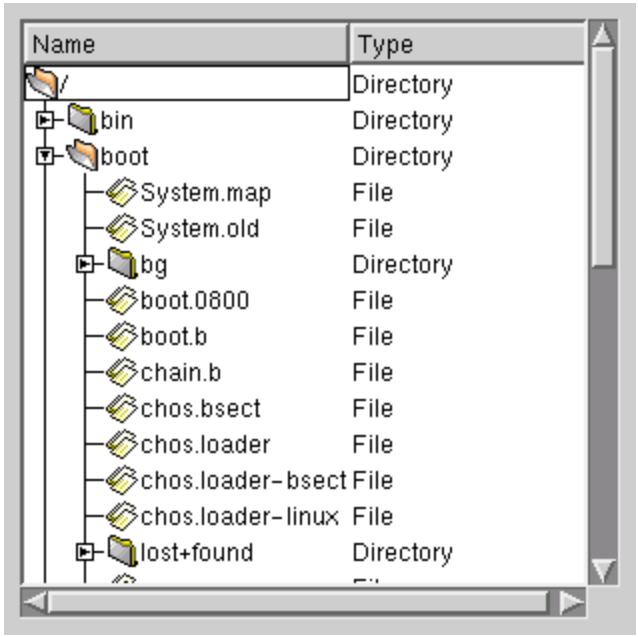
Designing An Interface



What Do We Want?







Data Structure Categories

- Lists
  - One dimensional arrays of data
- Tables
  - Two dimensional arrays of data
- Trees
  - Hierarchies of lists
- Graphs

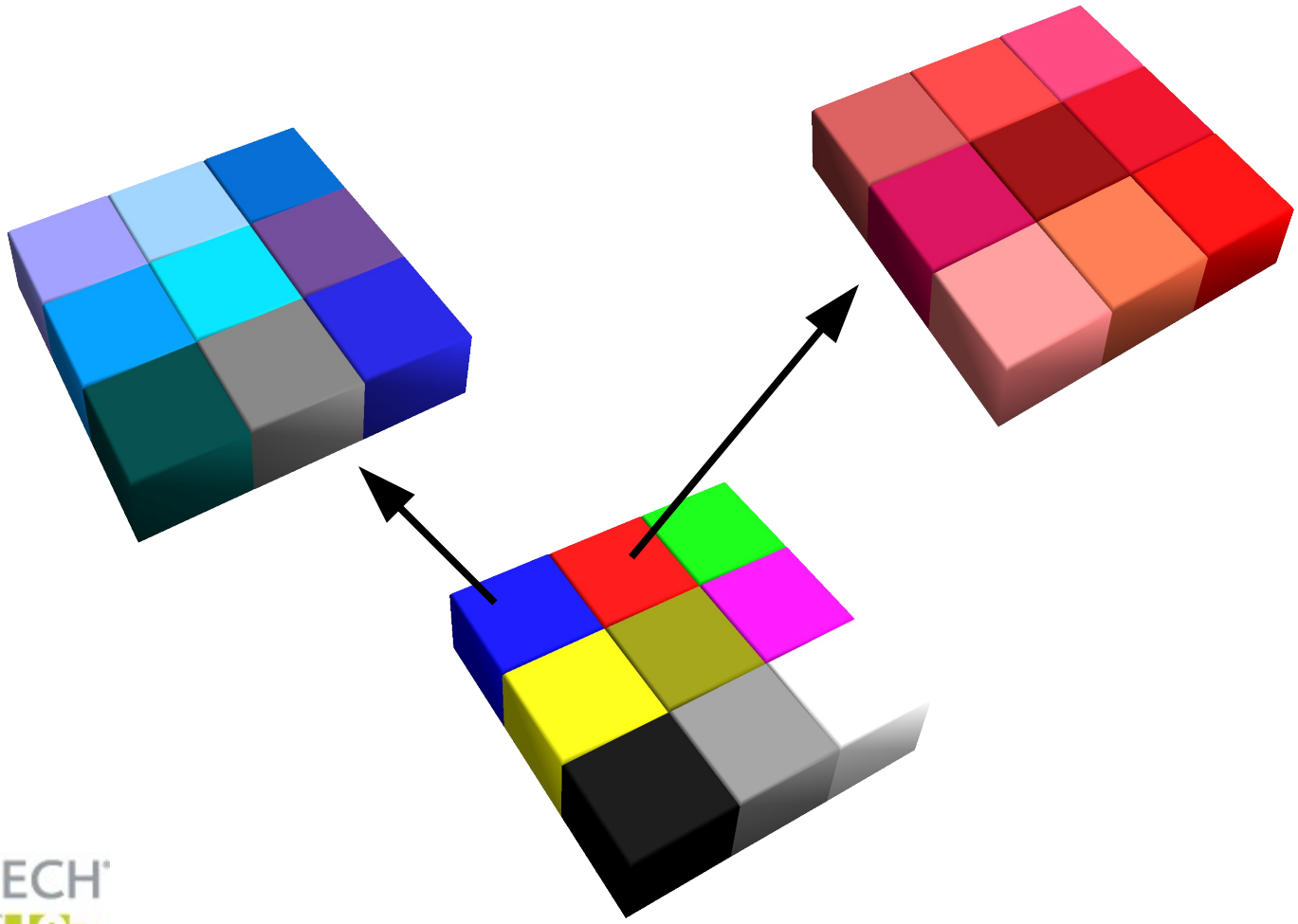
Data Structure Representation

- Lists
  - One dimensional arrays of data
- Tables
  - Two dimensional arrays of data
- Trees
  - Hierarchies of lists
- Graphs

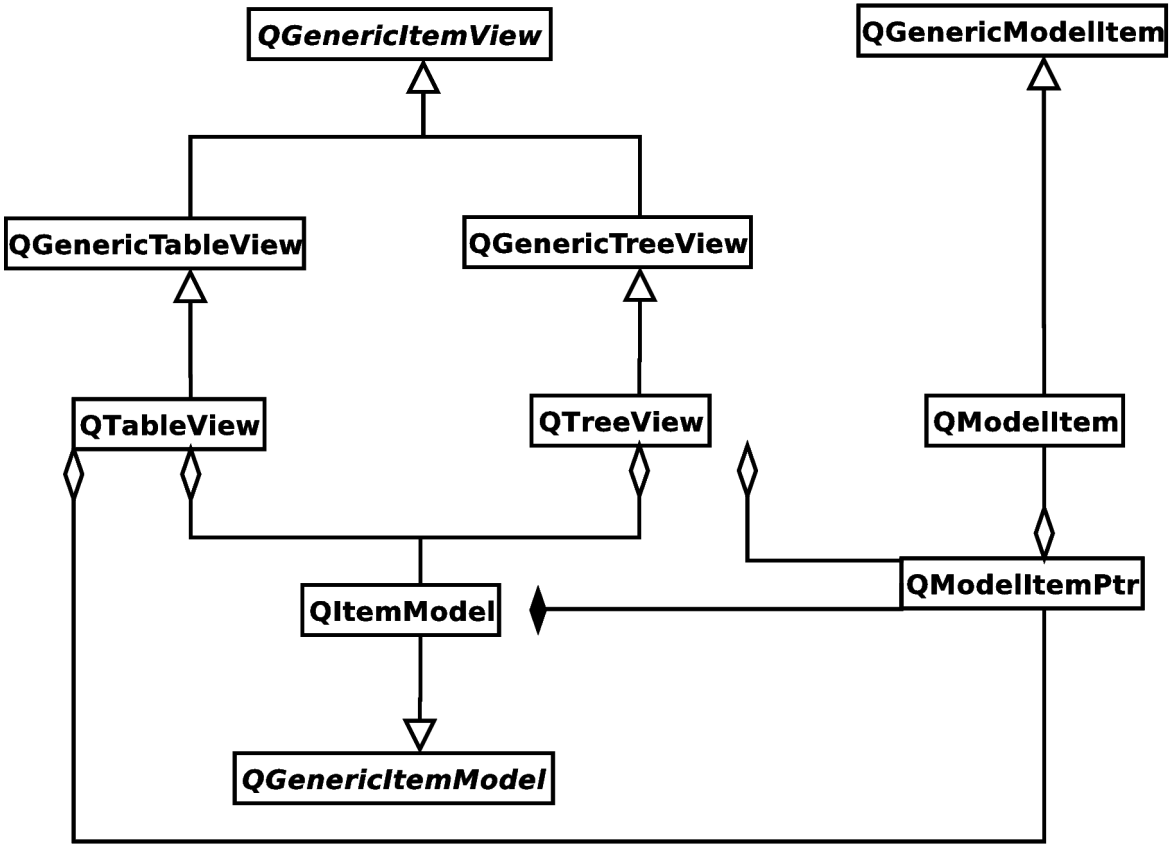
???

TROLLTECH DEV DAYS '07

A Hierarchy Of Tables



The First Prototypes (January 2003)



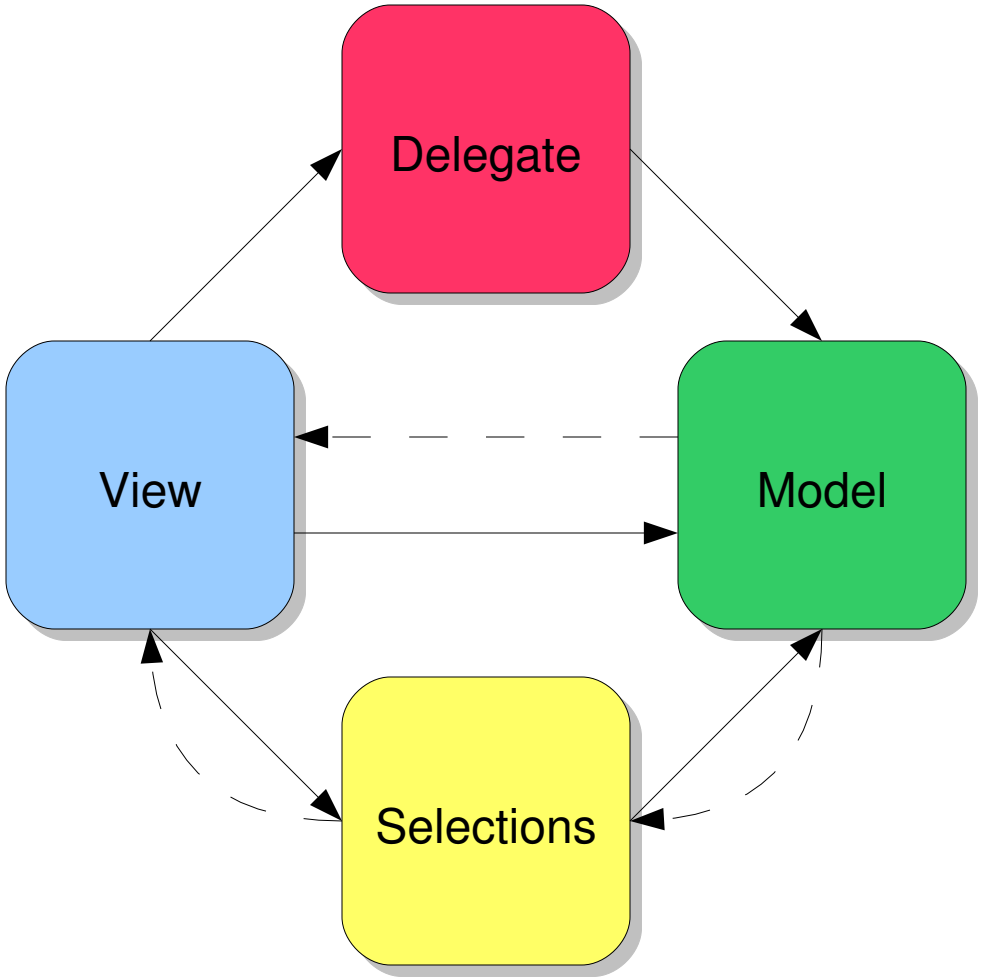
Look Ma, No Items! (Oct. 2003)

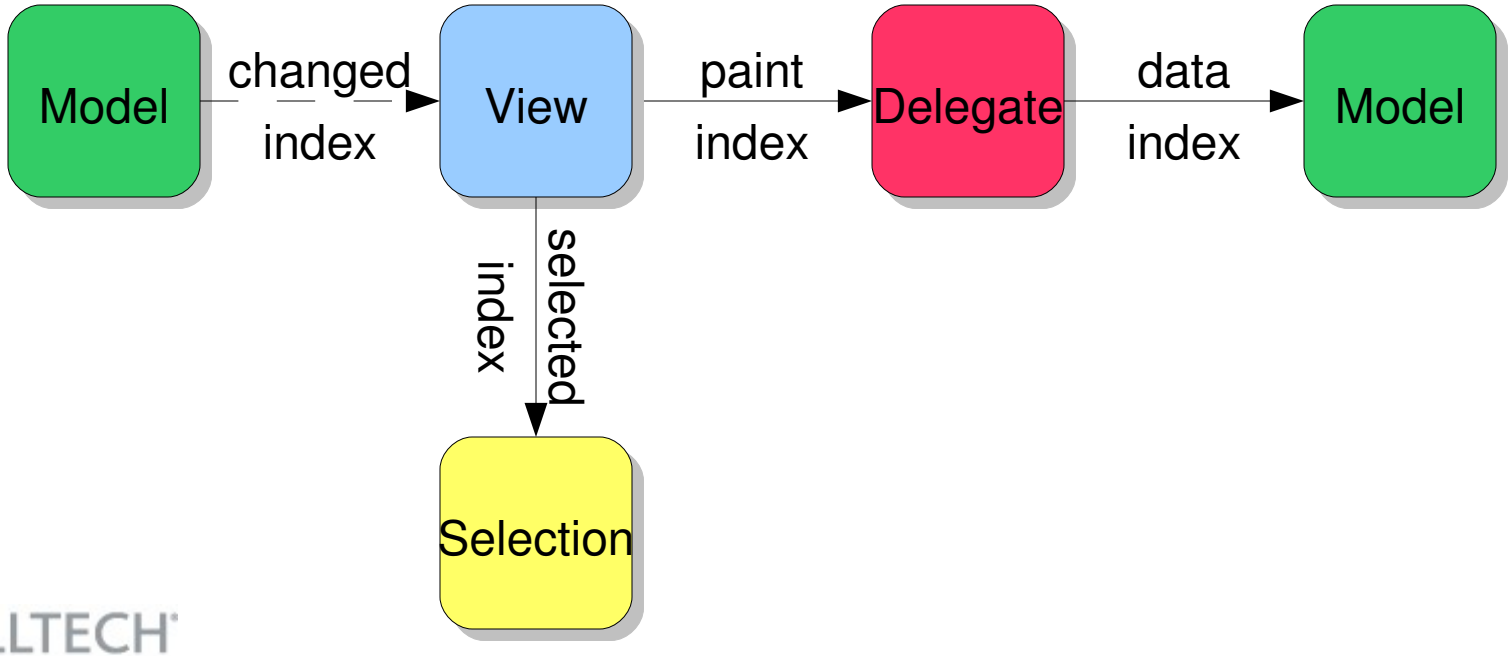
```
Change from QGenericModelItem to QModelIndex
---------------------------------------------
Why:

- allocating QGenericModelItems on the heap is too slow and we
do that a lot

- using a pool/repository is a workaround and not a proper
solution

- the QModelIndex is lightweight, have no virtual functions
and fast inline constructor and destructor

- we move all functionality to the model itself, which in
effect was what happened anyway
```

The Model Index

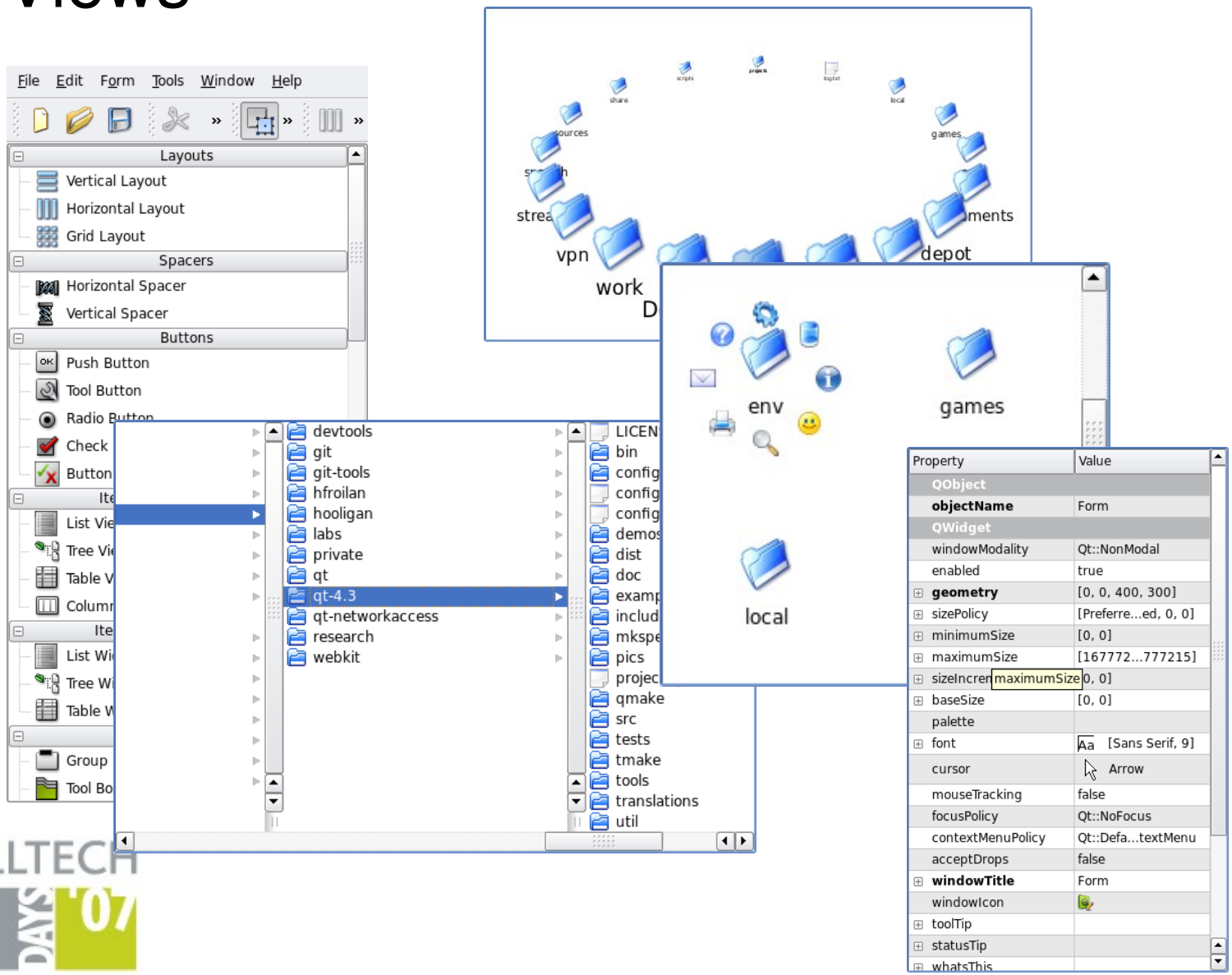
The Model Index

- Not an item
- Represents a position in a specific table
- Lightweight
- Shortlived


Delegate
View
Model
Selections


Model
changed
index
View
paint
index
Delegate
data
index
Model
index
selected
Selection

Views


File Edit Form Tools Window Help
Layouts
Vertical Layout
Horizontal Layout
Grid Layout
Spacers
Horizontal Spacer
Vertical Spacer
Buttons
Push Button
Tool Button
Radio Button
Check
Button
List Vie
Tree Vi
Table V
Column
List Wi
Tree Wi
Table W
Group
Tool Bo
vpn
work
depot
games
env
local
devtools
git
git-tools
hfroilan
hooligan
labs
private
qt
qt-4.3
qt-networkaccess
research
webkit
bin
config
config
config
demos
dist
doc
pics
qmake
src
tests
tmake
tools
translations
util
Property
Value
QObject
objectName
Form
QWidget
windowModality
Qt::NonModal
enabled
true
geometry
[0, 0, 400, 300]
sizePolicy
[Preferre...ed, 0, 0]
minimumSize
[0, 0]
maximumSize
[167772...777215]
sizeIncrem
maximumSize
0, 0]
baseSize
[0, 0]
palette
font
[Sans Serif, 9]
cursor
Arrow
mouseTracking
false
focusPolicy
Qt::NoFocus
contextMenuPolicy
Qt::Defa...textMenu
acceptDrops
false
windowTitle
Form
windowIcon
toolTip
statusTip
whatsThis

- Introduction
- Architecture
- **Models In Depth**
- Selections In Depth
- I Just Want To ...

TROLLTECH DEV DAYS '07

Models In Depth

- Main Model Classes
- List
- Tree
- Editable Items
- Inserting And Removing Items
- Lazy Initialization
- Drag And Drop
- Testing

TROLLTECH DEV DAYS '07

Main Model Classes

- QAbstractItemModel
  - **QAbstractListModel**
    - QStringListModel
  - **QAbstractTableModel**
    - QSqlQueryModel
  - **QStandardItemModel**
  - QAbstractProxyModel
    - QSortFilterProxyModel

QAbstractItemModel

- Base class for all models
- Use when
  - no existing model fits your use case
  - you need to wrap an existing tree/hierarchical structure
- Ignore when
  - only want to show 10 items in a list
  - you can use an existing model or widget

QAbstractItemModel

- 5 functions have to be implemented
  - `QModelIndex index(int row, int column,`
  - `                  const QModelIndex &parent) const`
  - `QModelIndex parent(const QModelIndex &index) const`
  - `int rowCount(const QModelIndex &parent) const`
  - `int columnCount(const QModelIndex &parent) const`
  - `QVariant data(const QModelIndex &index, int role) const`

TROLLTECH DEV DAYS '07

QAbstractTableModel

- 3 functions have to be implemented
  - `int **rowCount**(const QModelIndex &parent) const`
  - `int **columnCount**(const QModelIndex &parent) const`
  - `QVariant **data**(const QModelIndex &index, int role) const`

TROLLTECH DEV DAYS '07

QAbstractListModel

- 2 functions have to be implemented
  - `int rowCount(const QModelIndex &parent) const`
  - `QVariant data(const QModelIndex &index, int role) const`

TROLLTECH DEV DAYS '07

List Model

List Model

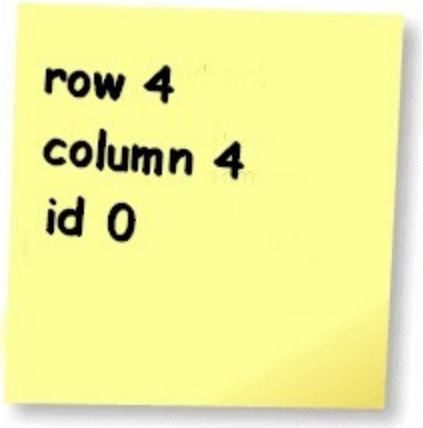
```
class MyModel : public QAbstractListModel
{
public:
    MyModel(QObject *parent) : QAbstractListModel(parent) {}

    int rowCount(const QModelIndex&) const {
        return list.count();
    }

    QVariant data(const QModelIndex &index, int role) const {
        if (index.isValid() && role == Qt::DisplayRole)
            return list.at(index.row());
        return QVariant();
    }
private:
    QList<QString> list;
};
```

The Model Index

The Model Index

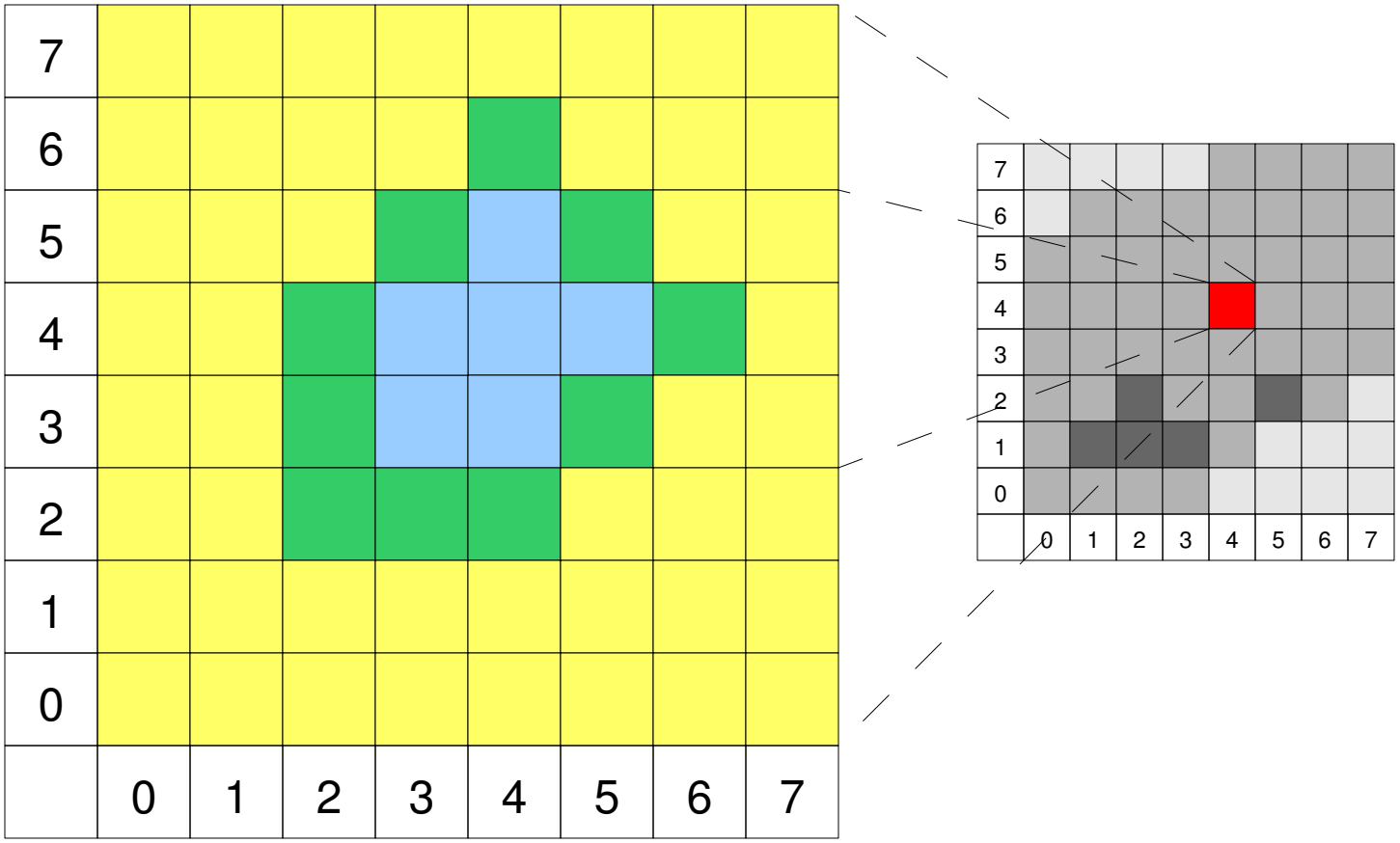
```
QVariant MyModel::data(const QModelIndex &index,
                       int role) const
{
    int row = index.row();
    int column = index.column();
    void *ptr = index.internalPointer();
    qint64 id = index.internalId(); // same as ptr

    MyInternalItem *item = lookup(row, column, id);
    return item->data(role);
}
```

The Model Index



7								
6								
5								
4								
3								
2								
1								
0								
	0	1	2	3	4	5	6	7

The Model Index



Tree Model

Models And Hierarchy

```
struct SimpleNode
{
    SimpleNode(SimpleNode *parent)
        : text(“text”), parentNode(parent) {
        if (parentNode)
            parentNode->children.append(this);
    }
    ~SimpleNode() {
        foreach(SimpleNode *child, children)
            delete child;
    }
    QVariant data() const {
        return text;
    }
    QString text;
    SimpleNode *parentNode;
    QList<SimpleNode*> children;
};
```

TROLLTECH DEV DAYS '07

Tree Model

```
class SimpleModel : public QAbstractItemModel
{
public:
    SimpleModel(QObject *parent = 0);
    ~SimpleModel();
    QModelIndex index(int row, int column,
                      const QModelIndex &parent) const;
    QModelIndex parent(const QModelIndex &child) const;
    int rowCount(const QModelIndex &parent) const;
    int columnCount(const QModelIndex &parent) const;
    QVariant data(const QModelIndex &index, int role) const;
protected:
    SimpleNode *nodeForIndex(const QModelIndex &index) const;
    int rowForNode(SimpleNode *node) const;
private:
    SimpleNode *root;
};
```

Tree Model

```
QModelIndex SimpleModel::index(int row, int column,
                               const QModelIndex &parent) const
{
    if (hasIndex(row, column, parent)) {
        SimpleNode *parentNode = nodeForIndex(parent);
        SimpleNode *childNode = parentNode->children.at(row);
        return createIndex(row, column, childNode);
    }
    return QModelIndex();
}
```

Tree Model

```
QModelIndex SimpleModel::parent(const QModelIndex &child) const
{
    SimpleNode *childNode = nodeForIndex(child);
    SimpleNode *parentNode = childNode->parentNode;

    if (parentNode == root)
        return QModelIndex();

    int row = rowForNode(parentNode);
    int column = 0;
    return createIndex(row, column, parentNode);
}
```

Tree Model

```
int SimpleModel::rowCount(const QModelIndex &parent) const
{
    SimpleNode *parentNode = nodeForIndex(parent);
    return parentNode->children.count();
}

int SimpleModel::columnCount(const QModelIndex &parent) const
{
    Q_UNUSED(parent);
    return 1;
}
```

Tree Model

```
QVariant SimpleModel::data(const QModelIndex &index,
                           int role) const
{
    if (index.isValid() && role == Qt::DisplayRole) {
        SimpleNode *node = nodeForIndex(index);
        return node->data();
    }
    return QVariant();
}
```

TROLLTECH DEV DAYS '07

Tree Model

```
SimpleNode *SimpleModel::nodeForIndex(const QModelIndex &index)
const
{
    if (index.isValid())
        return static_cast<SimpleNode*>(
            index.internalPointer());
    return root;
}

int SimpleModel::rowForNode(SimpleNode *node) const
{
    return node->parentNode->children.indexOf(node);
}
```

TROLLTECH DEV DAYS '07

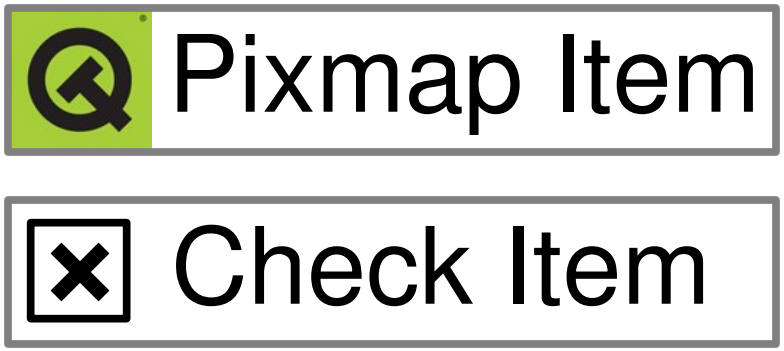
Editable Items

TROLLTECH DEV DAYS '07

Editable Items

```
class EditableModel : public SimpleModel
{
    Q_OBJECT
public:
    EditableModel(QObject *parent = 0);
    ~EditableModel();

    bool setData(const QModelIndex &index,
                 const QVariant &value, int role);

    Qt::ItemFlags flags(const QModelIndex &index) const;
};
```

Editable Items



Qt::ItemDataRole

- **DisplayRole**
  - usually text
- **DecorationRole**
  - usually an icon
- **EditRole**
  - key data to be edited
- **CheckStateRole**
  - item check box state

Qt::ItemDataRole

- **AccessibleTextRole**
- **AccessibleDescriptionRole**
- SizeHintRole
- TextAlignmentRole
- BackgroundRole
- ForegroundRole
- TextColorRole

Editable Items

```
bool EditableModel::setData(const QModelIndex &index,
                            const QVariant &value, int role)
{
    if (index.isValid() && role == Qt::EditRole) {
        SimpleNode *node = nodeForIndex(index);
        node->text = value.toString();
        QModelIndex topLeft = index;
        QModelIndex bottomRight = index;
        emit dataChanged(topLeft, bottomRight);
        return true;
    }
    return false;
}
```

TROLLTECH DEV DAYS '07

Editable Items

```
Qt::ItemFlags EditableModel::flags(const QModelIndex &index)
const
{
    Qt::ItemFlags defaultFlags = SimpleModel::flags(index);
    if (index.isValid())
        return Qt::ItemIsEditable | defaultFlags;
    return defaultFlags;
}
```

TROLLTECH DEV DAYS '07

Qt::ItemFlags

- **ItemIsSelectable**
- ItemIsEditable
- ItemIsDragEnabled
- ItemIsDropEnabled
- ItemIsUserCheckable
- **ItemIsEnabled**
- ItemIsTristate

Inserting And Removing Items

Inserting And Removing Items

```
class InsertRemoveModel : public EditableModel
{
public:
    InsertRemoveModel(QObject *parent = 0);
    ~InsertRemoveModel();

    // specialized insert and remove functions
    void insertNode(int i,
                    SimpleNode *parentNode,
                    SimpleNode *node);
    void removeNode(SimpleNode *node);
    void removeAllNodes();
};
```

Inserting And Removing Items

```
void InsertRemoveModel::insertNode(int i,
                                   SimpleNode *parentNode,
                                   SimpleNode *node)
{
    const QModelIndex parent = indexForNode(parentNode);
    int firstRow = i;
    int lastRow = i;

    beginInsertRows(parent, firstRow, lastRow);
    parentNode->children.insert(i, node);
    endInsertRows();
}
```

Inserting And Removing Items

```
void InsertRemoveModel::removeNode(SimpleNode *node)
{
    SimpleNode *parentNode = node->parent;
    const QModelIndex parent = indexForNode(parentNode);
    int i = rowForNode(node);
    int firstRow = i;
    int lastRow = i;

    beginRemoveRows(parent, firstRow, lastRow);
    parentNode->children.remove(i);
    endRemoveRows();
}
```

TROLLTECH DEV DAYS '07

Inserting And Removing Items

```
void InsertRemoveModel::removeAllNodes()
{
    root->children.clear();
    reset();
}
```

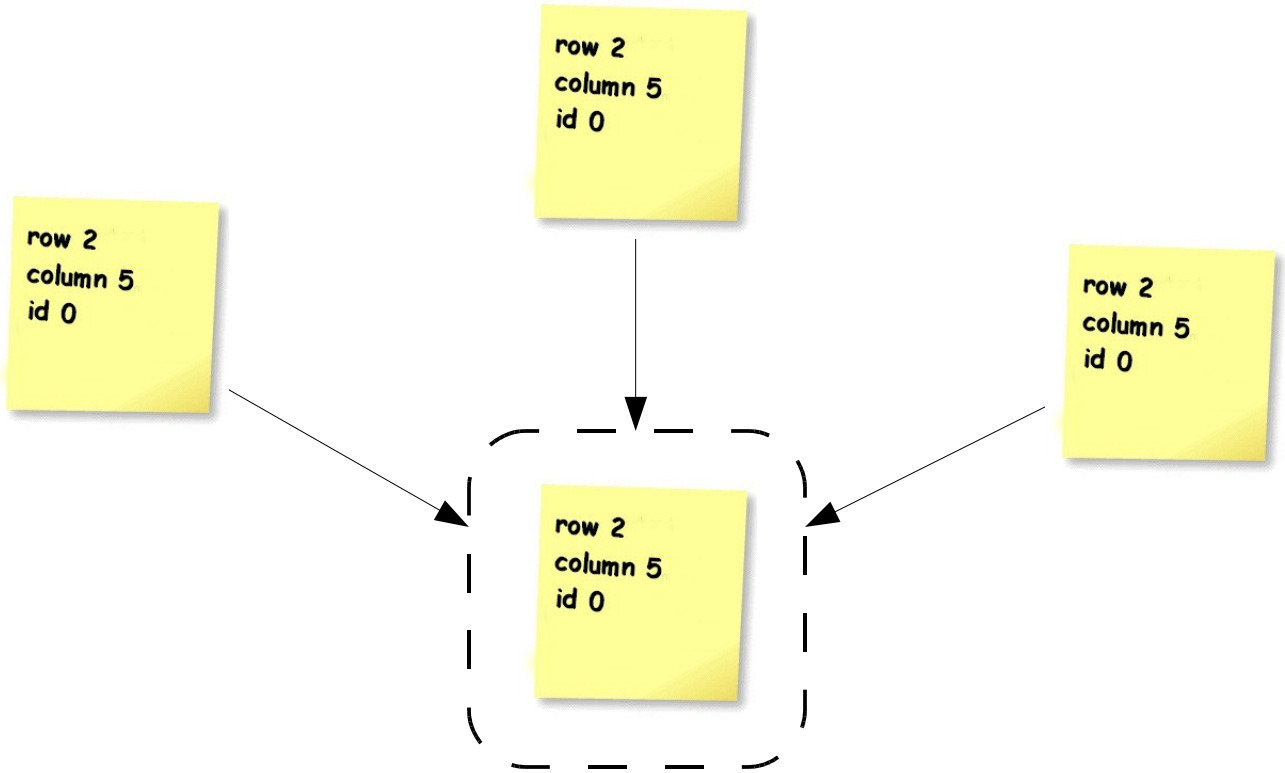
The Persistent Model Index

TROLLTECH DEV DAYS '07

The Persistent Model Index

- A Model Index
- Persistent (i.e. not short lived)
- Updated By The Model

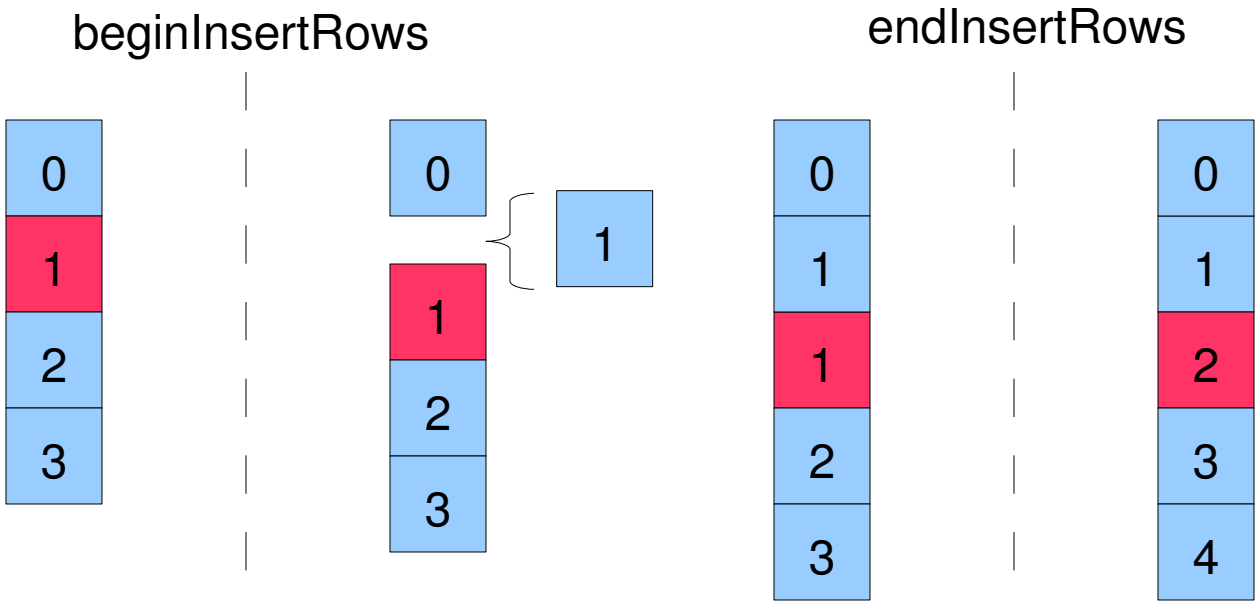
The Persistent Model Index



Persistent Model Index

```
class Q_CORE_EXPORT QPersistentModelIndexData
{
public:
    QPersistentModelIndexData();
    QPersistentModelIndexData(const QModelIndex &idx);
    QModelIndex index;
    QAtomicInt ref;
    const QAbstractItemModel *model;
    ...
};
```

The Persistent Model Index



Persistent Model Index

```
void changePersistentIndex(const QModelIndex &from,
                           const QModelIndex &to);

void changePersistentIndexList(const QModelIndexList &from,
                               const QModelIndexList &to);

QModelIndexList persistentIndexList() const;
```

Lazy Initialization

Lazy Initialization

```
class LazyModel : public SimpleModel
{
public:
    LazyModel(QObject *parent = 0);
    ~LazyModel();

    bool hasChildren(const QModelIndex &parent) const;
    bool canFetchMore(const QModelIndex &parent) const;
    void fetchMore(const QModelIndex &parent)
};
```

Lazy Initialization

```
bool LazyModel::hasChildren(const QModelIndex &parent) const
{
    int depth = 0;
    SimpleNode *parentNode = nodeForIndex(parent);
    while (parentNode) {
        parentNode = parentNode->parentNode;
        ++depth;
    }
    return (depth <= 3);
}
```

Lazy Initialization

```
bool LazyModel::canFetchMore(const QModelIndex &parent) const
{
    static const int maxChildren = 100;
    SimpleNode *parentNode = nodeForIndex(parent);
    return parentNode->children.count() < maxChildren;
}
```

Lazy Initialization

```
void LazyModel::fetchMore(const QModelIndex &parent)
{
    static const int maxChildren = 100;
    SimpleNode *parentNode = nodeForIndex(parent);
    int count = parentNode->children.count();
    if (count < maxChildren) {
        beginInsertRows(parent, count, maxChildren - count);
        while (parentNode->children.count() < maxChildren)
            new SimpleNode(parentNode);
        endInsertRows();
    }
}
```

TROLLTECH DEV DAYS '07

Drag And Drop

TROLLTECH DEV DAYS '07

Drag And Drop

```
class DndModel : public InsertRemoveModel
{
    ...
    Qt::ItemFlags flags(const QModelIndex &index) const;
    Qt::DropActions supportedDragActions() const;
    Qt::DropActions supportedDropActions() const;
    QStringList mimeTypes() const;
    QMimeData *mimeData(const QModelIndexList &indexes) const;
    ...
};
```

Drag And Drop

```
Qt::ItemFlags DndModel::flags(const QModelIndex &index) const
{
    if (index.isValid())
        return Qt::ItemIsDragEnabled
             | Qt::ItemIsDropEnabled
             | InsertRemoveModel::flags(index);

    // we allow dropping in the empty area (invalid index)
    return Qt::ItemIsDropEnabled
         | InsertRemoveModel::flags(index);
}
```

Drag And Drop

```
Qt::DropActions DndModel::supportedDragActions() const
{
    return Qt::CopyAction | Qt::MoveAction;
}

Qt::DropActions DndModel::supportedDropActions() const
{
    return Qt::CopyAction | Qt::MoveAction;
}
```

Drag And Drop

```
QStringList DndModel::mimeTypes() const
{
    QStringList types;
    types << "application/vnd.text.list";
    return types;
}
```

Drag And Drop

```
QMimeData *DndModel::mimeData(const QModelIndexList &indexes) const
{
    QByteArray encodedData;
    QDataStream stream(&encodedData, QIODevice::WriteOnly);
    foreach (QModelIndex index, indexes) {
        if (index.isValid()) {
            QString text = data(index, Qt::DisplayRole).toString();
            stream << text;
        }
    }
    QMimeData *mimeData = new QMimeData();
    mimeData->setData("application/vnd.text.list", encodedData);
    return mimeData;
}
```

Drag And Drop

```
bool insertRows(int row, int count, const QModelIndex &parent);
bool insertColumns(int column, int count,
                   const QModelIndex &parent);

bool removeRows(int row, int count, const QModelIndex &parent);
bool removeColumns(int column, int count,
                   const QModelIndex &parent);
```

Drag And Drop

```
class DndModel : public InsertRemoveModel
{
    ...
    bool dropMimeData(const QMimeData *data,
                      Qt::DropAction action,
                      int row, int column,
                      const QModelIndex &parent);
    bool removeRows(int row, int count,
                    const QModelIndex &parent);
};
```

TROLLTECH DEV DAYS '07

Drag And Drop

```
bool DndModel::dropMimeData(const QMimeData *data,
                            Qt::DropAction action,
                            int row, int column,
                            const QModelIndex &parent)
{
    if (action == Qt::IgnoreAction)
        return true;
    if (!data->asFormat("application/vnd.text.list"))
        return false;
    if (column >= columnCount(parent))
        return false;
```

TROLLTECH DEV DAYS '07

Drag And Drop

```
    // find where to insert the new items
    SimpleNode *parentNode = nodeForIndex(parent);
    // get the data
    QByteArray encodedData = data->data("application/vnd.text.list");
    QDataStream stream(&encodedData, QIODevice::ReadOnly);
    while (!stream.atEnd()) {
        QString text;
        stream >> text;
        SimpleNode *node = new SimpleNode(0);
        node->text = text;
        insertNode(row, parentNode, node);
    }
    return true;
}
```

TROLLTECH DEV DAYS '07

Drag And Drop

```
bool DndModel::removeRows(int row, int count,
                          const QModelIndex &parent)
{
    beginRemoveRows(row, row + count - 1, parent);
    SimpleNode *parentNode = nodeForIndex(parent);
    for (int i = 0; i < count; ++i) {
        if (row < parentNode->children.count()) {
            SimpleNode *node = parentNode->children.at(row);
            parentNode->children.remove(row);
            node->parentNode = 0;
        }
    }
    endRemoveRows();
}
```

TROLLTECH DEV DAYS '07

Testing

- QtTestLibrary
- **ModelTest**
  - http://labs.trolltech.com/page/Projects/Itemview/Modeltest

Testing

- **ModelTest**
  - Signal emission
  - Hierarchy
  - Row and column count
  - Parent / Child relationships
  - Much more...

Testing

```
int main(int argc, char *argv[])
{
    SimpleModel model;
    ModelTest test(&model);
    return 0;
}
```

- Add to the project (.pro) file:
  - **include(../path/to/dir/modeltest.pri)**

TROLLTECH DEV DAYS '07

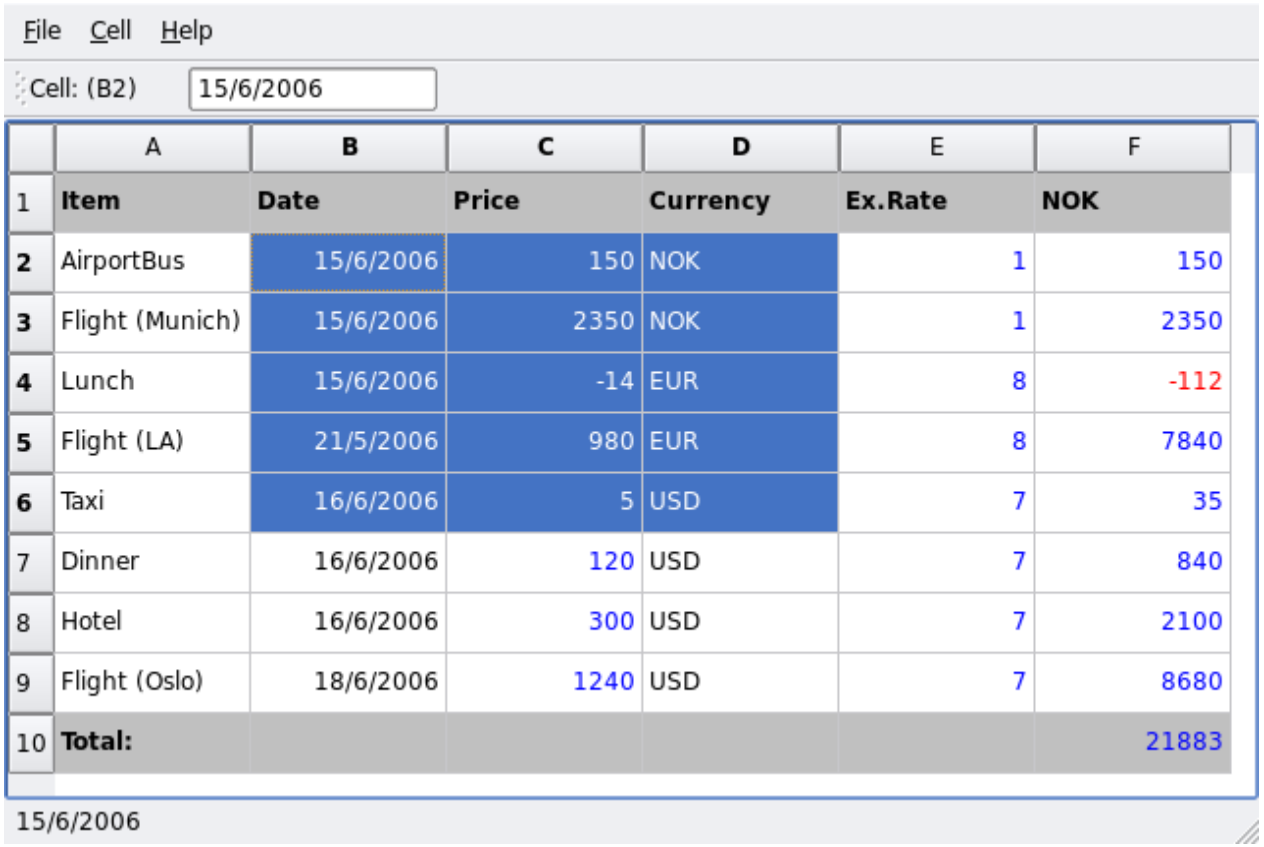
- Introduction
- Architecture
- Models In Depth
- **Selections In Depth**
- I Just Want To ...

TROLLTECH DEV DAYS '07

Selections In Depth

- Selection Ranges
- Making Selections
- Selection Properties
- Tips When Selecting

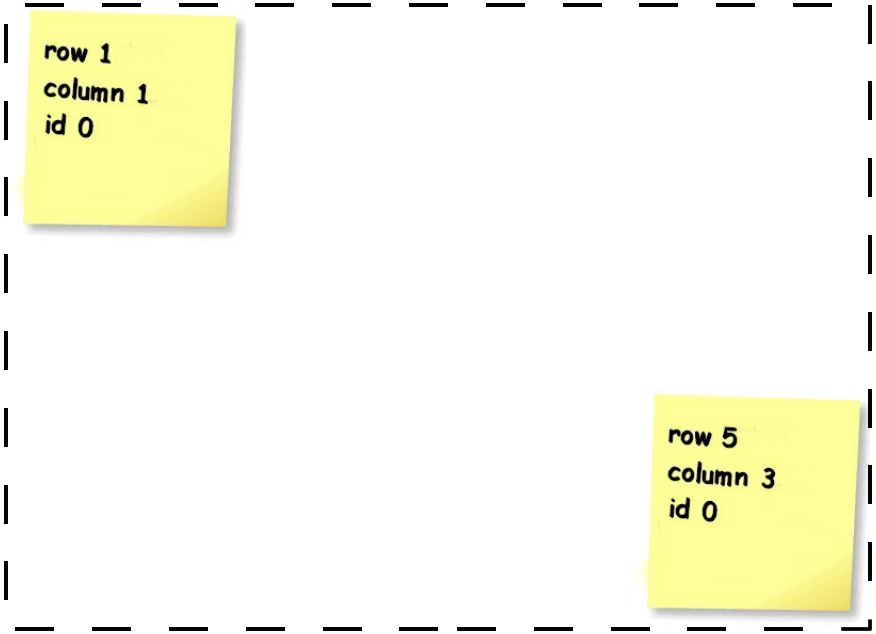
Selections In Depth



	A	B	C	D	E	F
1	**Item**	**Date**	**Price**	**Currency**	**Ex.Rate**	**NOK**
2	AirportBus	15/6/2006	150	NOK	1	150
3	Flight (Munich)	15/6/2006	2350	NOK	1	2350
4	Lunch	15/6/2006	-14	EUR	8	-112
5	Flight (LA)	21/5/2006	980	EUR	8	7840
6	Taxi	16/6/2006	5	USD	7	35
7	Dinner	16/6/2006	120	USD	7	840
8	Hotel	16/6/2006	300	USD	7	2100
9	Flight (Oslo)	18/6/2006	1240	USD	7	8680
10	**Total:**					21883

TROLLTECH DEV DAYS '07

Selection Range



Selection Ranges

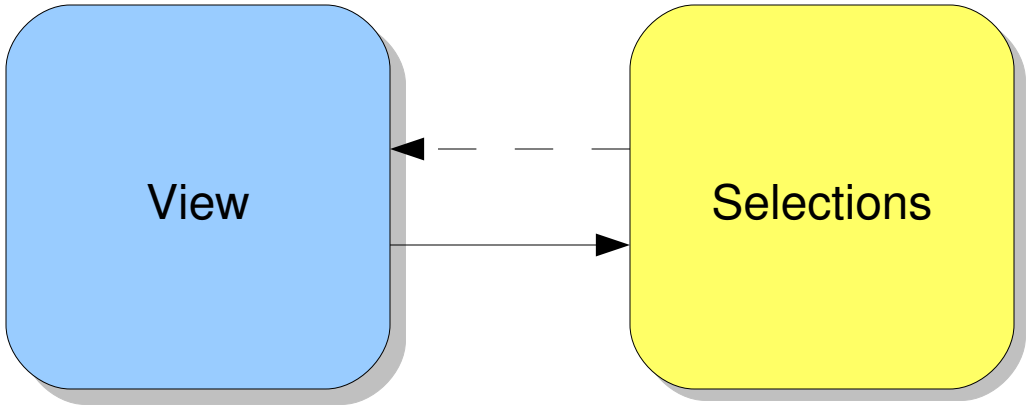
- Benefits
  - Selecting consecutive items is fast
  - Memory efficient (with few or no selections)
  - Fast (with few or no selections)
- Problems
  - Selecting many individual items is slow
  - Slow (with many selection ranges)

Selections

- Selection Range
  - Top Left Index
  - Bottom Right Index
- Selection
  - List of Ranges
- Selection Model
  - Lost of Selections
  - Current Selection
  - Current Index

TROLLTECH DEV DAYS '07

Views And Selections



Making Selections

```
void select(const QItemSelection &selection,
            QItemSelectionModel::SelectionFlags command)
```

Making Selections

- SelectionFlags
  - **NoUpdate -** don't change the selections
  - **Clear** - clear existing selections
  - **Select** – select the given ranges
  - **Deselect** – unselect the given ranges
  - **Toggle** – reverse the selection state of the given ranges
  - **Current** – set the current selection
  - **Rows** – expand the given ranges to rows
  - **Columns** – expand the given ranges to to columns

View Selection Properties

- **SelectionMode**
- **SelectionBehavior**

TROLLTECH DEV DAYS '07

View Selection Properties

- **SelectionBehavior**
  - SelectItems
  - SelectRows
  - SelectColumns

View Selection Properties

- **SelectionMode**
  - NoSelection
  - SingleSelection
  - MultiSelection
  - ContigousSelection
  - ExtendedSelection

View Selection Properties

- SelectionMode
  - **NoSelection**
    - No selection ranges
  - **SingleSelection**
    - Single item (may be expanded by SelectionBehavior)
  - **MultiSelection**
    - Multiple selection ranges
    - Toggles existing selected items
  - ContiguousSelection
  - ExtendedSelection

View Selection Properties

- SelectionMode
  - NoSelection
  - SingleSelection
  - MultiSelection
  - **ContigousSelection**
    - Clears existing selections before selecting
    - Shift + selecting will update the current selection only
  - ExtendedSelection

View Selection Properties

- SelectionMode
  - NoSelection
  - SingleSelection
  - MultiSelection
  - ContiguousSelection
  - **ExtendedSelection**
    - Clears existing selections before selecting
    - Shift + selecting will select the current selection only
    - Ctrl + selecting will toggle the current selection range

Tips When Selecting

- Select Using Ranges
- Compile Ranges in Selection Objects
- Clear When Possible
  - avoid merging selection ranges
- Avoid Toggle
  - avoids splitting selection ranges

- Introduction
- Architecture
- Models In Depth
- Selections In Depth
- **I Just Want To ...**

TROLLTECH DEV DAYS '07

I Just Want To...

- Show Some Data
- Insert Lots Of Items Fast
- Do Sorting And Filtering
- Have Checkable Items
- Do Custom Item Painting
- Have A Custom Item Editor
- Animate Items

TROLLTECH DEV DAYS '07

Showing Data

- Widgets
  - QListWidget
  - QTableWidget
  - QTreeWidget
- Models
  - QStringListModel
  - QStandardItemModel

Showing Data

- The General Rules (tm)
  - Models
    - Faster
    - Less Overhead
    - Harder To Implement
  - Widgets
    - Slower
    - More Overhead
    - Easier To Use

TROLLTECH DEV DAYS '07

Showing Data Using A Widget

```
int main(int argc, char *argv[])
{
    QApplication app(argc, argv);
    QStringList entryList = QDir::current().entryList();

    QListWidget widget;
    foreach (QString entry, entryList) {
        QListWidgetItem *item = new QListWidgetItem(entry);
        widget.addItem(item);
    }

    widget.show();
    return app.exec();
}
```

Showing Data Using A Widget

```
int main(int argc, char *argv[])
{
    QApplication app(argc, argv);
    QStringList entryList = QDir::current().entryList();

    QListWidget widget;
    widget.addItems(entryList);

    widget.show();
    return app.exec();
}
```

TROLLTECH DEV DAYS '07

Showing Data Using A Model

```
int main(int argc, char *argv[])
{
    QApplication app(argc, argv);
    QStringList entryList = QDir::current().entryList();

    QStringListModel model(entryList);
    QListView view;
    view.setModel(&model);

    view.show();
    return app.exec();
}
```

Insert Lots Of Items Fast

- Bottlenecks
  - Converting Items To Model Indexes
  - Laying Out Items In The Views
- Workaround
  - Insert Items Before Setting The Model On The View
  - Insert Before Showing The View
  - Insert Lists Of Items
  - Q*Widgets – set the item data before you set the view

Inserting Lots Of Items Fast

```
int main(int argc, char *argv[])
{
    QApplication app(argc, argv);
    QTreeView view;
    QStandardItemModel model;
    QStandardItem *parent = model.invisibleRootItem();
    for (int i = 0; i < 100000; ++i) {
        QStandardItem *item = new QStandartModelItem();
        item->setText(“test”);
        parent->appendRow(item);
    }
    view.setModel(&model);
    view.show();
    app.exec();
}
```

TROLLTECH DEV DAYS '07

Inserting Lots Of Items Fast

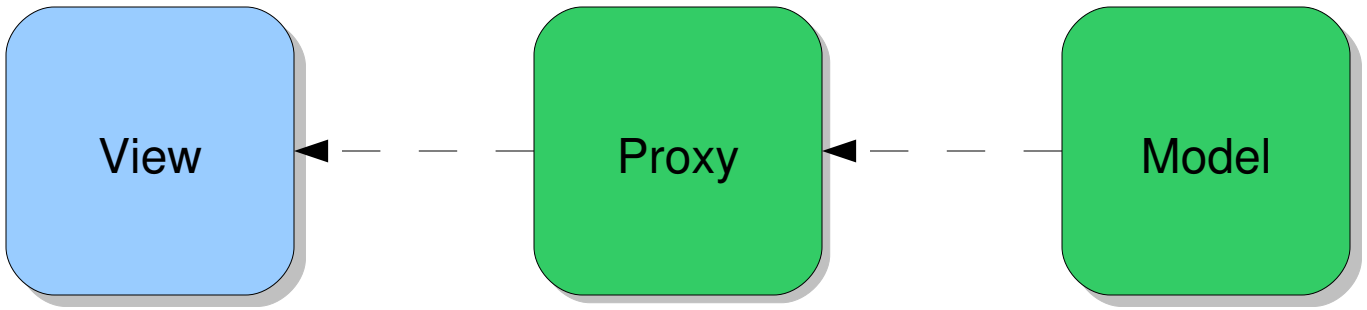
```
int main(int argc, char *argv[])
{
    QApplication app(argc, argv);
    QTreeWidget widget;
    QList<QTreeWidgetItem*> items;
    for (int i = 0; i < 100000; ++i) {
        QTreeWidgetItem *item = new QTreeWidgetItem();
        item->setText(0, “test”);
        items.append(item);
    }
    widget.invisibleRootItem()->addChildren(items);
    widget.show();
    app.exec();
}
```

TROLLTECH DEV DAYS '07

Sorting And Filtering

- Sorting
  - QListWidget
  - QTableWidget
  - QTreeWidget
  - **QSortFilterProxyModel**
- Filtering
  - **QSortFilterProxyModel**

QSortFilterProxyModel



Sorting And Filtering

```
int main(int argc, char *argv[])
{
    QApplication app(argc, argv);

    QStringListModel model;
    model.setStringList(QDir::current().entryList());

    QSortFilterProxyModel filter;
    filter.setSourceModel(&model);

    QListView view;
    view.setModel(&filter);
    view.show();

    return app.exec();
}
```

TROLLTECH DEV DAYS '07

Checkable Items

- **Qt::ItemDataRoles**
  - ItemCheckStateRole
- **Qt::ItemFlags**
  - ItemIsUserCheckable
  - ItemIsTristate
- **Qt::CheckedState**
  - Unchecked
  - PartiallyChecked
  - Checked

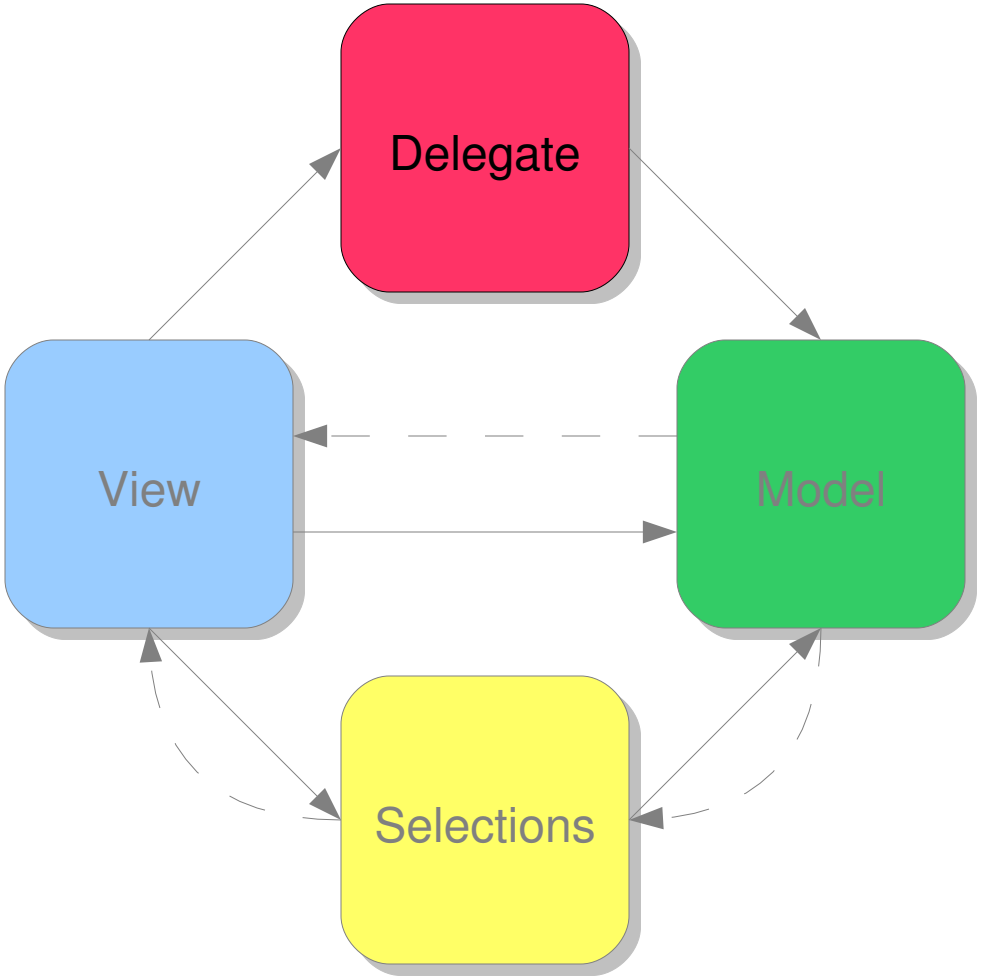
TROLLTECH DEV DAYS '07

Checkable Items

```
int main(int argc, char *argv[])
{
    QApplication app(argc, argv);
    QTreeWidget widget;

    for (int i = 0; i < 100; ++i) {
        QTreeWidgetItem *item = new QTreeWidgetItem;
        item->setText(0, “hello”);
        item->setCheckState(0, Qt::Unchecked);
        widget.invisibleRootItem()->addChild(item);
    }

    widget.show;
    return app.exec();
}
```

TROLLTECH DEV DAYS '07

Custom Item Painting



Custom Item Painting

- QAbstractItemDelegate
  - **QItemDelegate**
    - QSqlRelationalDelegate

Custom Item Painting

- Item Delegate
  - Painting
  - Clipping
  - Size Hint
  - Creating Editor Widget
  - Managing Editor Widget Geometry

Custom Item Painting

```
class CustomDelegate : public QItemDelegate
{
public:
    CustomDelegate(QObject *parent = 0)
        : QItemDelegate(parent) {}
    ~CustomDelegate() {}

    void paint(QPainter *painter,
               const QStyleOptionViewItem &option,
               const QModelIndex &index) const
    {
        painter->fillRect(option.rect, Qt::green);
        painter->drawText(option.rect,
                          index.data().toString());
    }
};
```

TROLLTECH DEV DAYS '07

Custom Item Painting

```
int main(int argc, char *argv[])
{
    QApplication app(argc, argv);
    QListWidget widget;
    widget.addItems(QDir::current().entryList());

    CustomDelegate *delegate = new CustomDelegate(&widget);
    widget.setItemDelegate(delegate);

    widget.show();
    return app.exec();
}
```

TROLLTECH DEV DAYS '07

Custom Item Painting

```
class CustomDelegate : public QItemDelegate
{
public:
    CustomDelegate(QObject *parent = 0)
        : QItemDelegate(parent) {}
    ~CustomDelegate() {}

    void paint(QPainter *painter,
               const QStyleOptionViewItem &option,
               const QModelIndex &index) const
    {
        painter->fillRect(option.rect, Qt::green);
        painter->drawText(option.rect,
                          index.data().toString());
    }
};
```

TROLLTECH DEV DAYS '07

Custom Item Editor

- QAbstractItemDelegate::createEditor(...)
- **QItemEditorFactory**
  - **QItemEditorCreatorBase**

Custom Item Editor

```
class Button : public QPushButton
{
    Q_OBJECT
public:
    Button(QWidget *parent) : QPushButton(parent) {
        setCheckable(true);
        connect(this, SIGNAL(toggled(bool)),
                this, SLOT(showState(bool)));
    }
protected slots:
    void showState(bool checked) {
        setText(checked ? "true" : "false");
    }
};
```

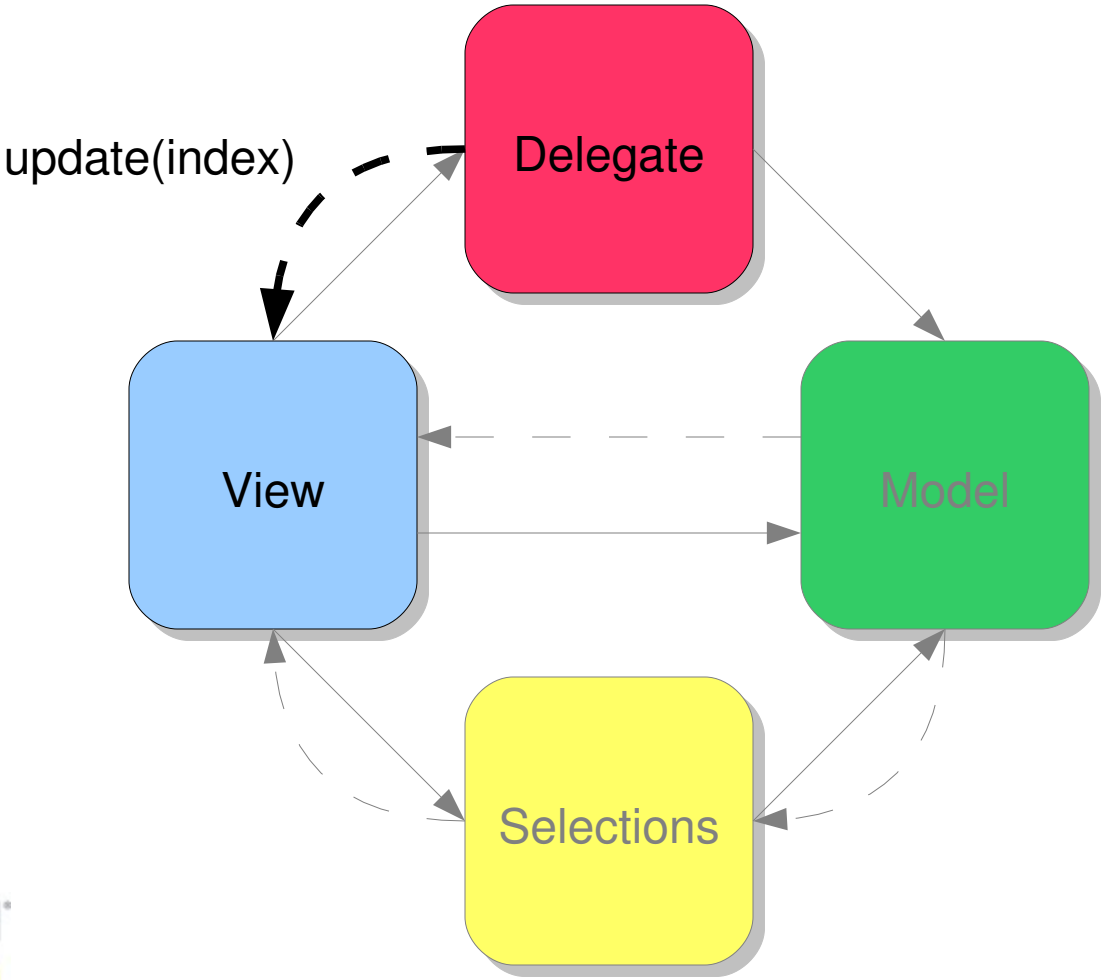
TROLLTECH DEV DAYS '07

Custom Item Editor

```
int main(int argc, char *argv[])
{
    ...
    QItemDelegate *delegate =
        qobject_cast<QItemDelegate*>(view.itemDelegate())

    QItemEditorCreatorBase *creator =
        new QItemEditorCreator<Button>("checked");

    QItemEditorFactory factory;
    factory.registerEditor(QVariant::Bool, creator);
    delegate->setItemEditorFactory(&factory);

    view.show();
    return app.exec();
}
```

TROLLTECH DEV DAYS '07

Animated Items



Animated Items

```
class Delegate : public QItemDelegate
{
    Q_OBJECT
public:
    Delegate(QObject *parent = 0);
    ~Delegate();

    void paint(QPainter *painter,
               const QStyleOptionViewItem &option,
               const QModelIndex &index) const;

    QSize sizeHint(const QStyleOptionViewItem &option,
                   const QModelIndex &index) const;

signals:
    void update(const QModelIndex &index);
    ...
```

Animated Items

```
    ...
private:
    struct Animation {
        enum Type { Incrementing, Decrementing };
        inline Animation(const QModelIndex &idx = QModelIndex())
            : frame(0), type(Incrementing) { index = idx; }
        int frame;
        Type type;
        QPersistentModelIndex index;
    };

    int frameCount;
    int increment;
    int timerId;
    mutable QList<Animation> animations;
};
```

Animated Items

```
void Delegate::paint(QPainter *painter,
                     const QStyleOptionViewItem &option,
                     const QModelIndex &index) const
{
    bool mouseOver = option.state & QStyle::State_MouseOver;
    int f = animationFrame(index, mouseOver);
    QColor color(f, f, f);
    painter->save();
    painter->setBrush(color);
    painter->drawRoundRect(option.rect);
    painter->restore();
}
```

Animated Items

```
void Delegate::timerEvent(QTimerEvent *event)
{
    if (event->timerId() == timerId && !animations.isEmpty()) {
        for (int i = animations.count() - 1; i >= 0; --i) {
            emit update(animations.at(i).index);
            if (animations.at(i).type ==
                Animation::Incrementing) { // increment frame
                if (animations.at(i).frame < frameCount)
                    animations[i].frame += increment;
            } else {                       // decrement frame
                if (animations.at(i).frame <= 0)
                    animations.removeAt(i);
                else
                    animations[i].frame -= increment;
            }
        }
    }
}
```

Animated Items

```
int Delegate::animationFrame(const QModelIndex &index,
                             bool mouseOver) const
{
    int i = animationIndex(index);
    if (i != -1) { // we have a running animation
        if (!mouseOver // but the mouse is no longer over
            && animations.at(i).type == Animation::Incrementing)
            // reverse the animation
            animations[i].type = Animation::Decrementing;

        return qBound(0, animations.at(i).frame, frameCount);
    }

    // no running animation
    if (mouseOver)
        animations.append(Animation(index));
    return 0;
}
```

Resources

http://en.wikipedia.org/wiki/Model-view-controller

http://doc.trolltech.com/4.3/model-view.html

http://doc.trolltech.com/4.3/model-view-programming.html

http://labs.trolltech.com/page/Projects/Itemview/Modeltest

That's all folks!

Questions ?

TROLLTECH DEV DAYS '07