Lecture 7. Model-View-Controller Idiom Support in Qt

Cross-Platform Application Development

November 3, 2017

Model-View-Controller

"Model-View-Controller" Pattern

Model: an object representing information on the document.

View: an object responsible for the screen representation of the model. Establishes collaboration with it via "subscription/notification" protocol.

Controller: an object implementing UI behavior in response to user actions.

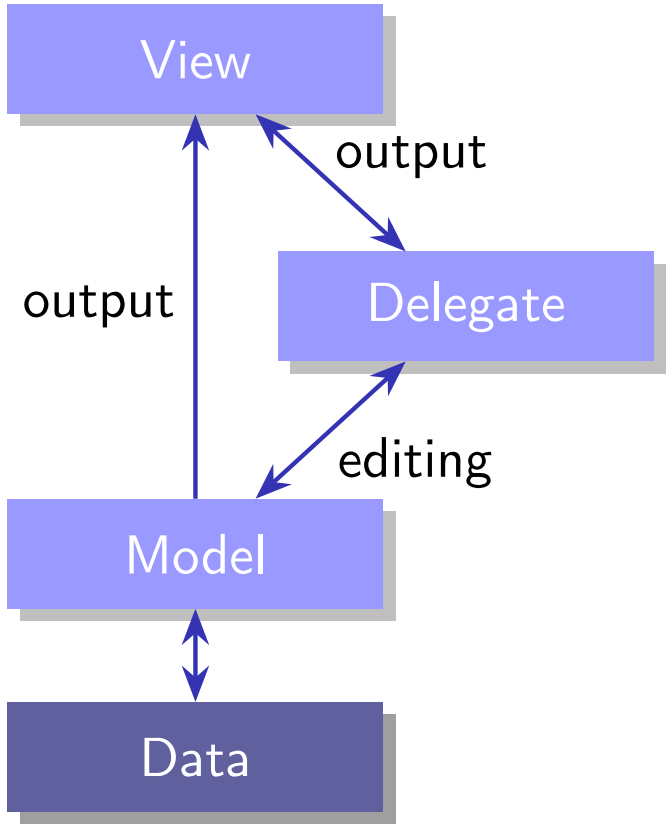

Figure 1: "Model-View-Controller" pattern

Signals from MVC Components

Source	Receivers	Information
Model	View	Data in the source change
View	UI	User interaction with data being displayed
Delegate	Model and view	Editor state

Table 1: information passed with signals from MVC objects

A Part of Qt Classes Hierarchy

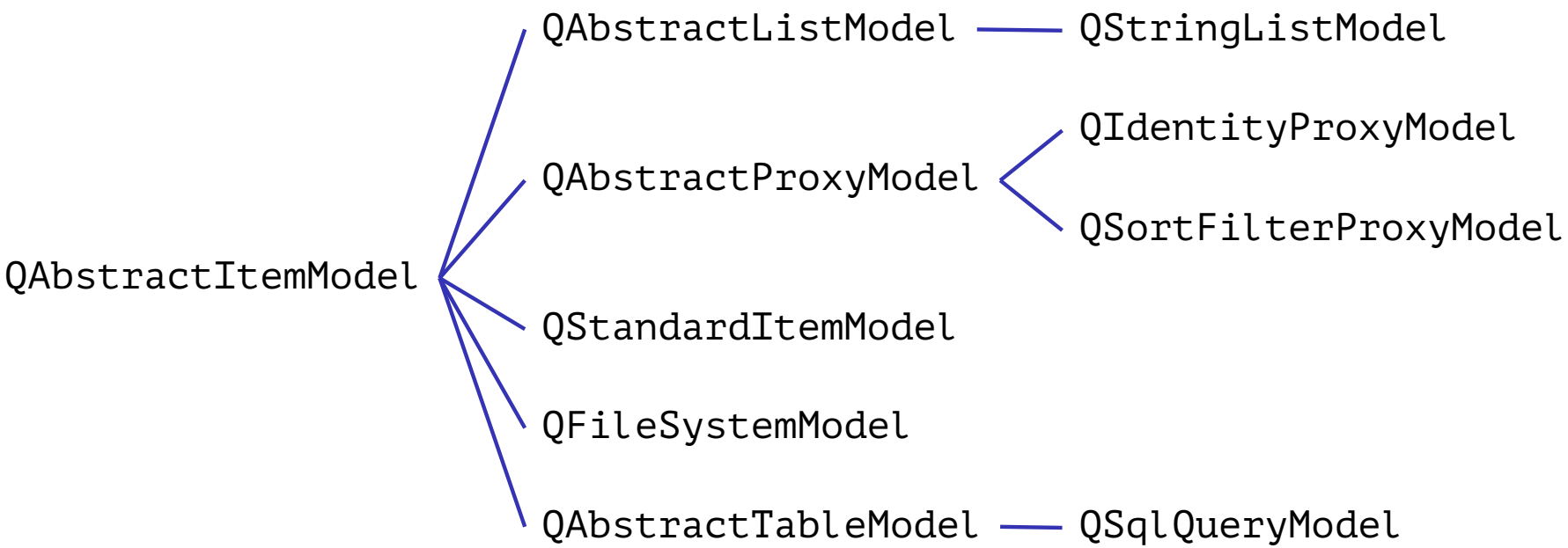

Figure 2: model classes hierarchy

A Part of Qt Classes Hierarchy (cont.)

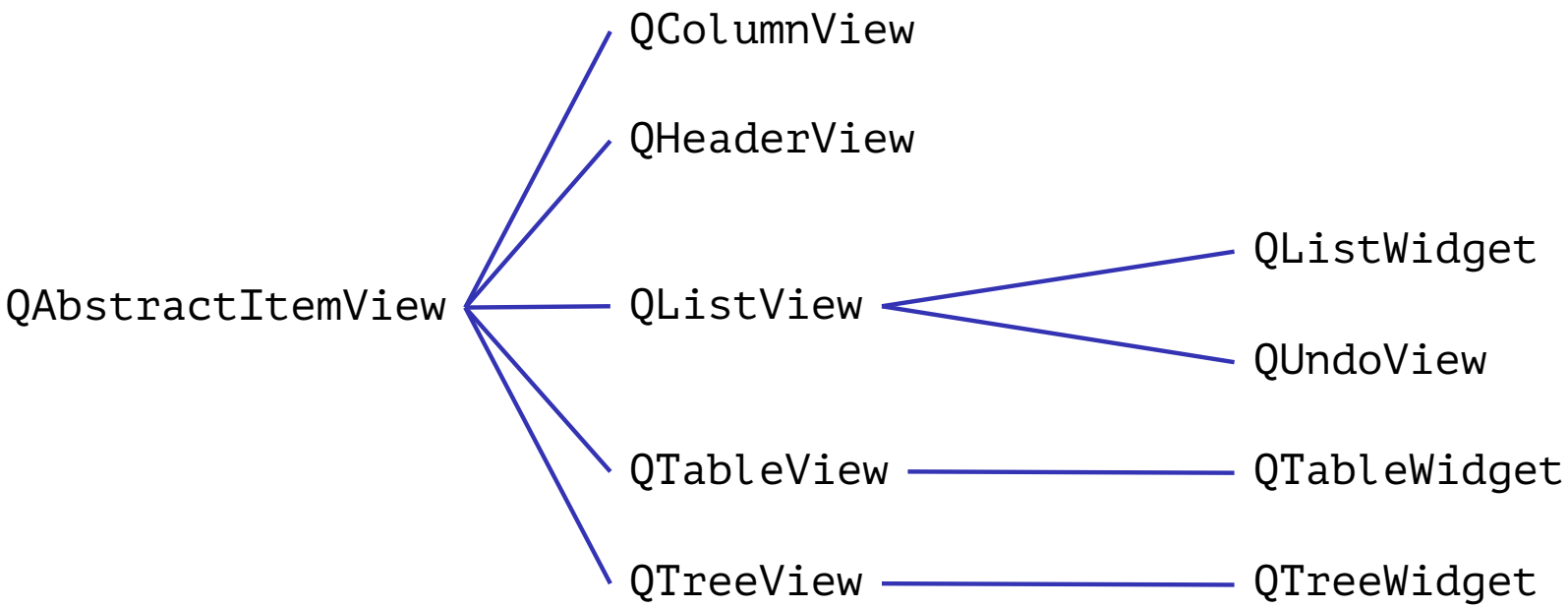

Figure 3: view classes hierarchy

A Part of Qt Classes Hierarchy (end)

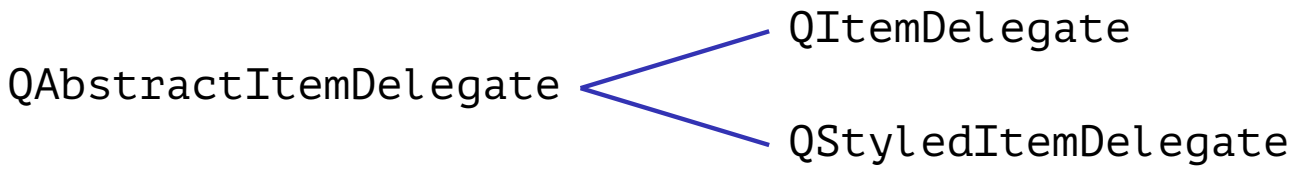

Figure 4: delegate classes hierarchy

Model Indexes

Example

```
QModelIndex index = pModel->index(
  nRow, nCol, parentIndex);
QPersistentModelIndex
  persistentIndex = index;
QAbstractItemModel *pAbstModel =
  index.model();
```

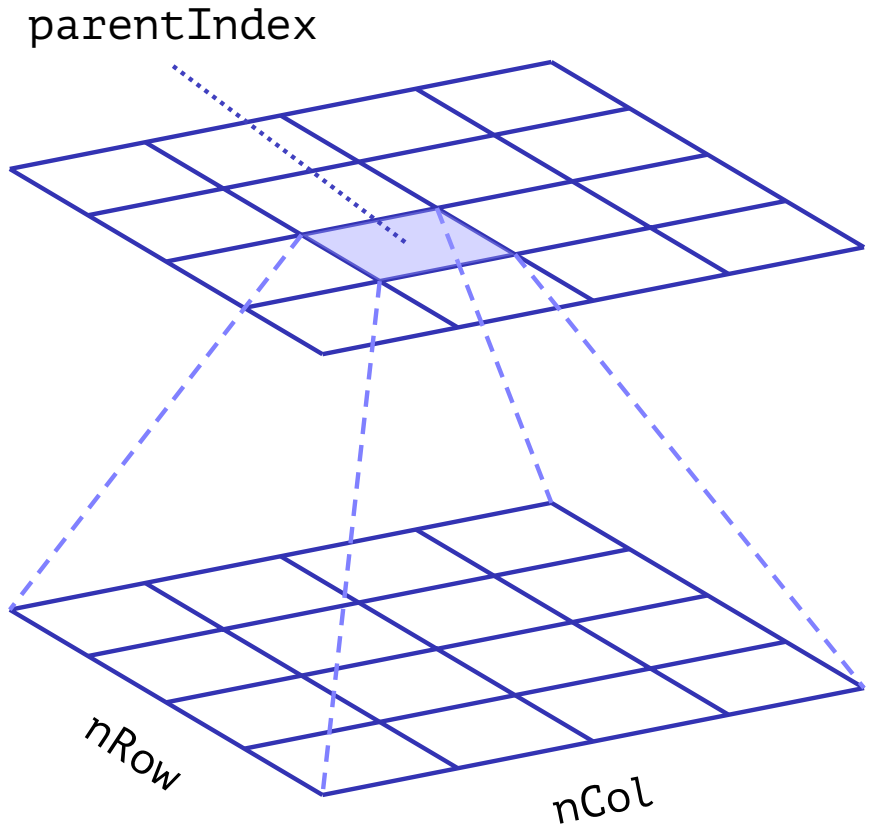

Figure 5: generalized indexes for a model

Main Model Types

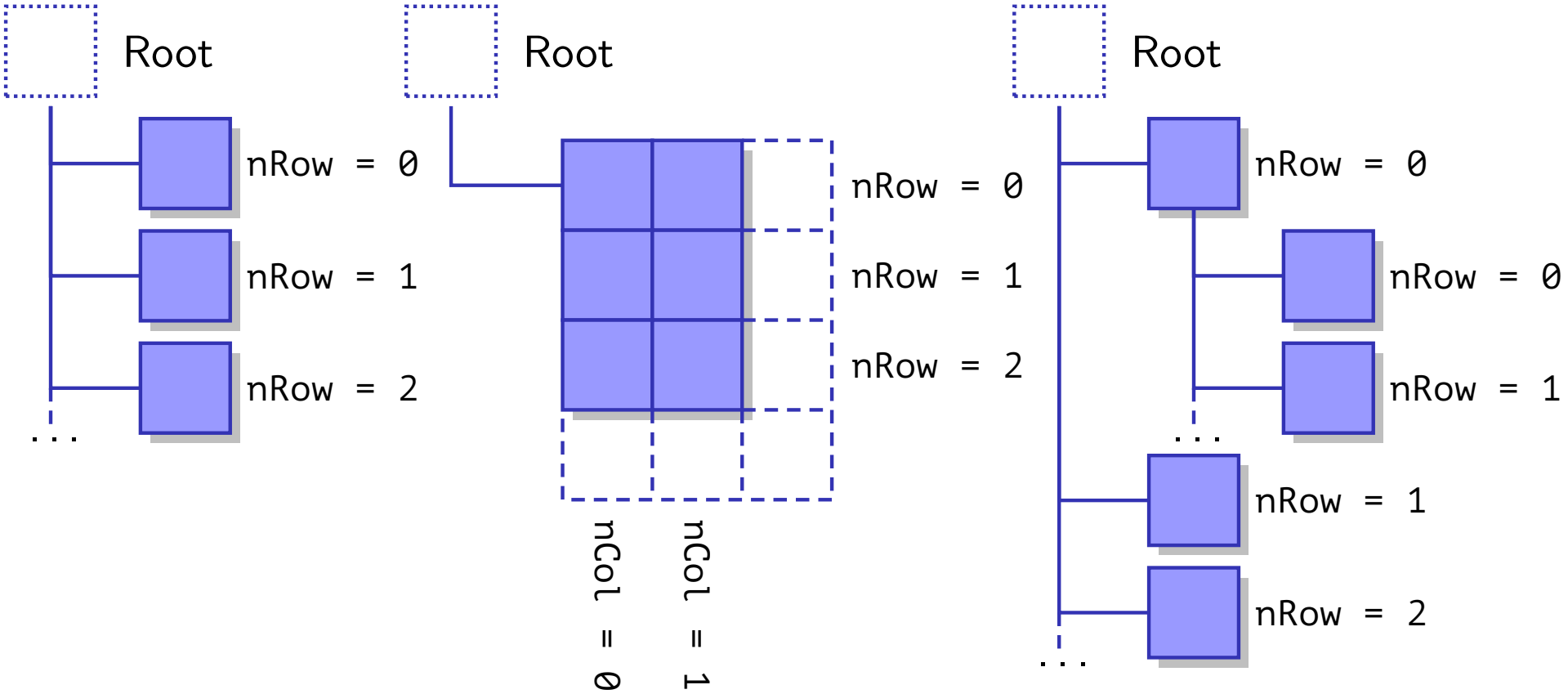

Figure 6: particular instances of model indexes

Using Indexes to Access a Tabular Model

Example

```
QModelIndex indexA = pModel->index(
  0, 0, QModelIndex());
QModelIndex indexB = pModel->index(
  1, 1, QModelIndex());
QModelIndex indexC = pModel->index(
  2, 1, QModelIndex());
```

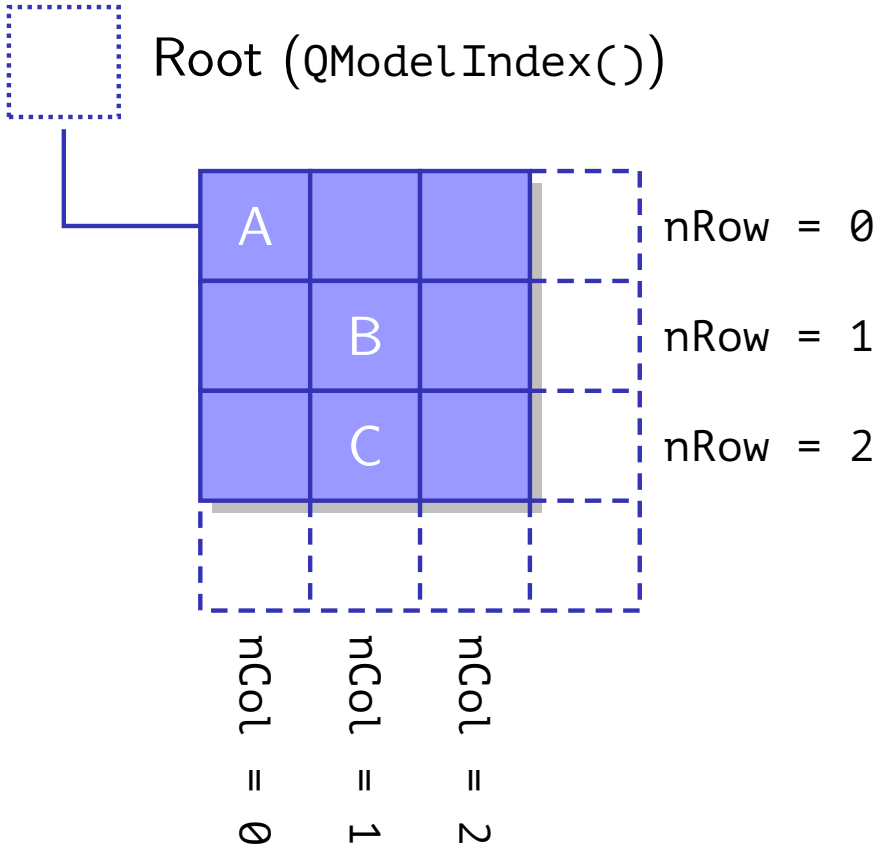

Figure 7: indexes for a tabular model

Using Indexes to Access a Tree Model

Example

```
QModelIndex indexA = pModel->index(
  0, 0, QModelIndex());
QModelIndex indexC = pModel->index(
  2, 1, QModelIndex());
QModelIndex indexB = pModel->index(
  1, 0, indexA);
```

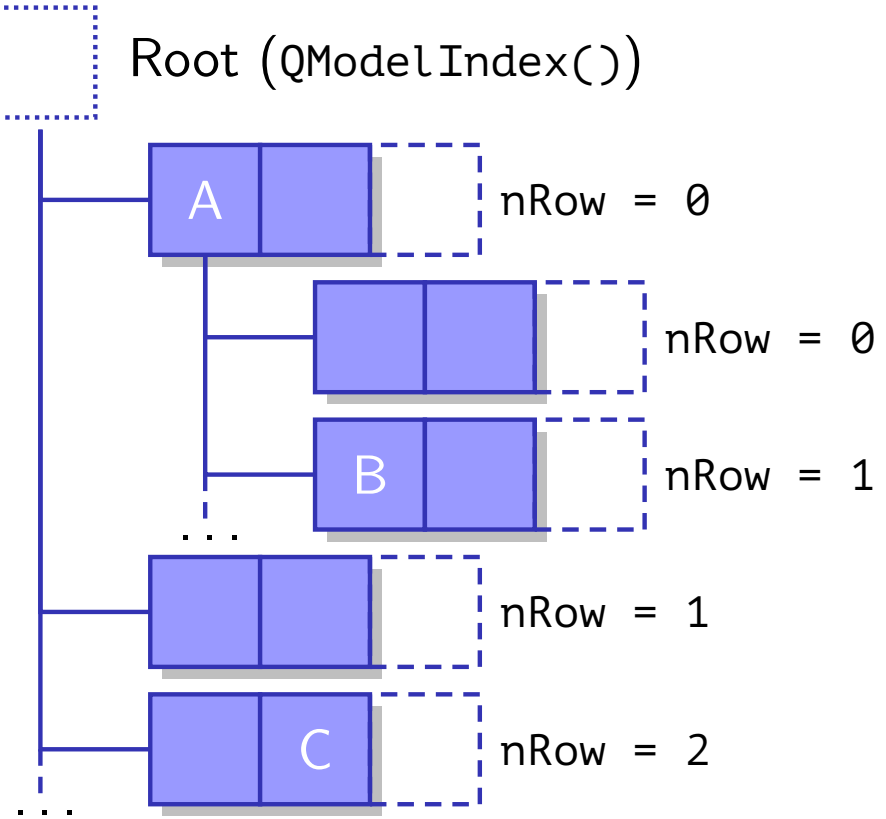

Figure 8: indexes for a tree model

Model Roles

Example (`enum Qt::ItemDataRole` Type)

```
QVariant value = pModel->data(index, Qt::DisplayRole);
```

DisplayRole	ToolTipRole	SizeHintRole
DecorationRole	StatusTipRole	
EditRole	WhatsThisRole	

Table 2: general purpose roles

FontRole	ForegroundRole	CheckStateRole
TextAlignmentRole	BackgroundRole	InitialSortOrderRole

Table 3: roles describing appearance and metadata

Example

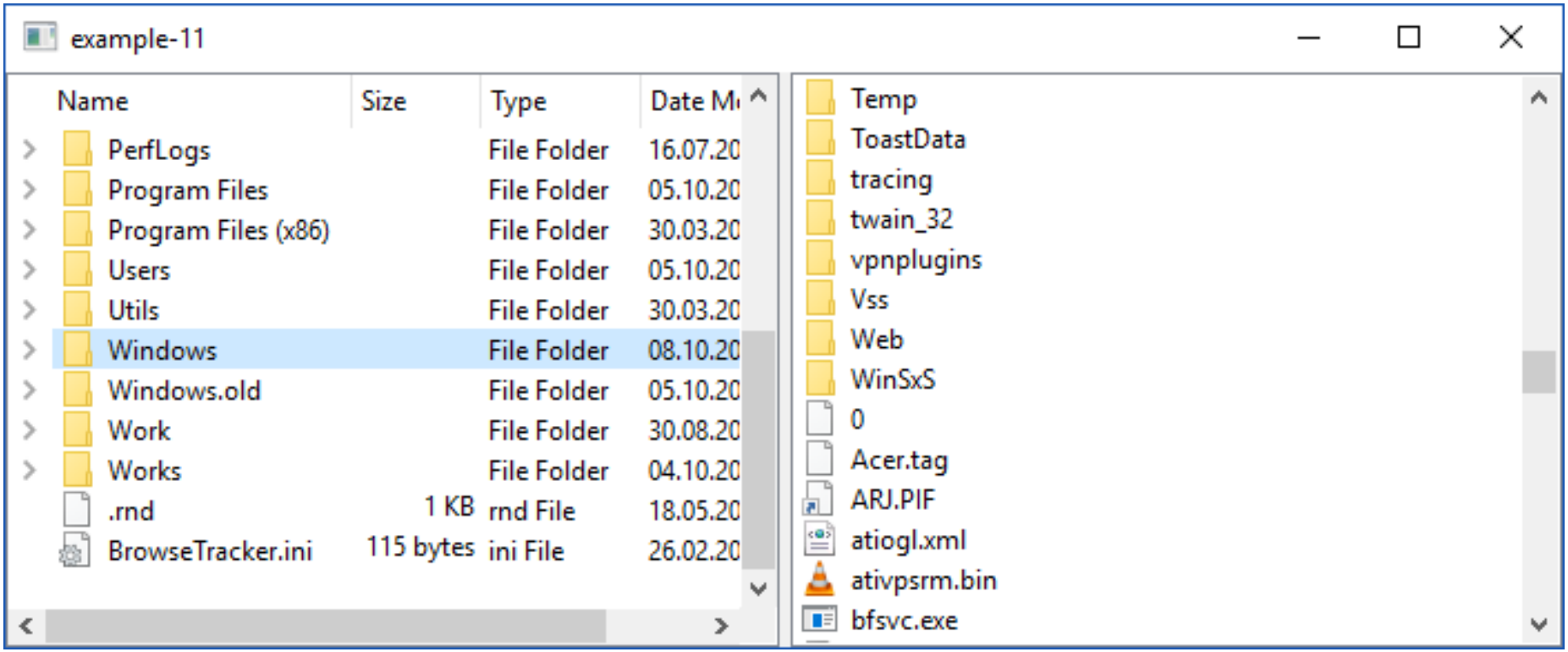

Figure 9: two views of one model

Example

Example (example-11.cpp)

```
int main(int nArgC, char *apszArgV[])
{
  QApplication app(nArgC, apszArgV);
  //
  QSplitter *pSplitter = new QSplitter;
  //
  QFileSystemModel *pModel = new QFileSystemModel;
  QString strPath = QDir::currentPath();
  pModel->setRootPath(strPath);
  //
```

Example (cont.)

Example (example-11.cpp, cont.)

```
QTreeView *pTreeView = new QTreeView(pSplitter);
pTreeView->setModel(pModel);
pTreeView->setRootIndex(pModel->index(strPath));
//
QListView *pListView = new QListView(pSplitter);
pListView->setModel(pModel);
pListView->setRootIndex(pModel->index(strPath));
//
```

Example (end)

Example (example-11.cpp, end)

```
  QObject::connect(
    pTreeView, SIGNAL(activated(const QModelIndex &)),
    pListView, SLOT(setRootIndex(const QModelIndex &)));
  //
  pSplitter->show();
  //
  return app.exec();
}    // main()
```

Example

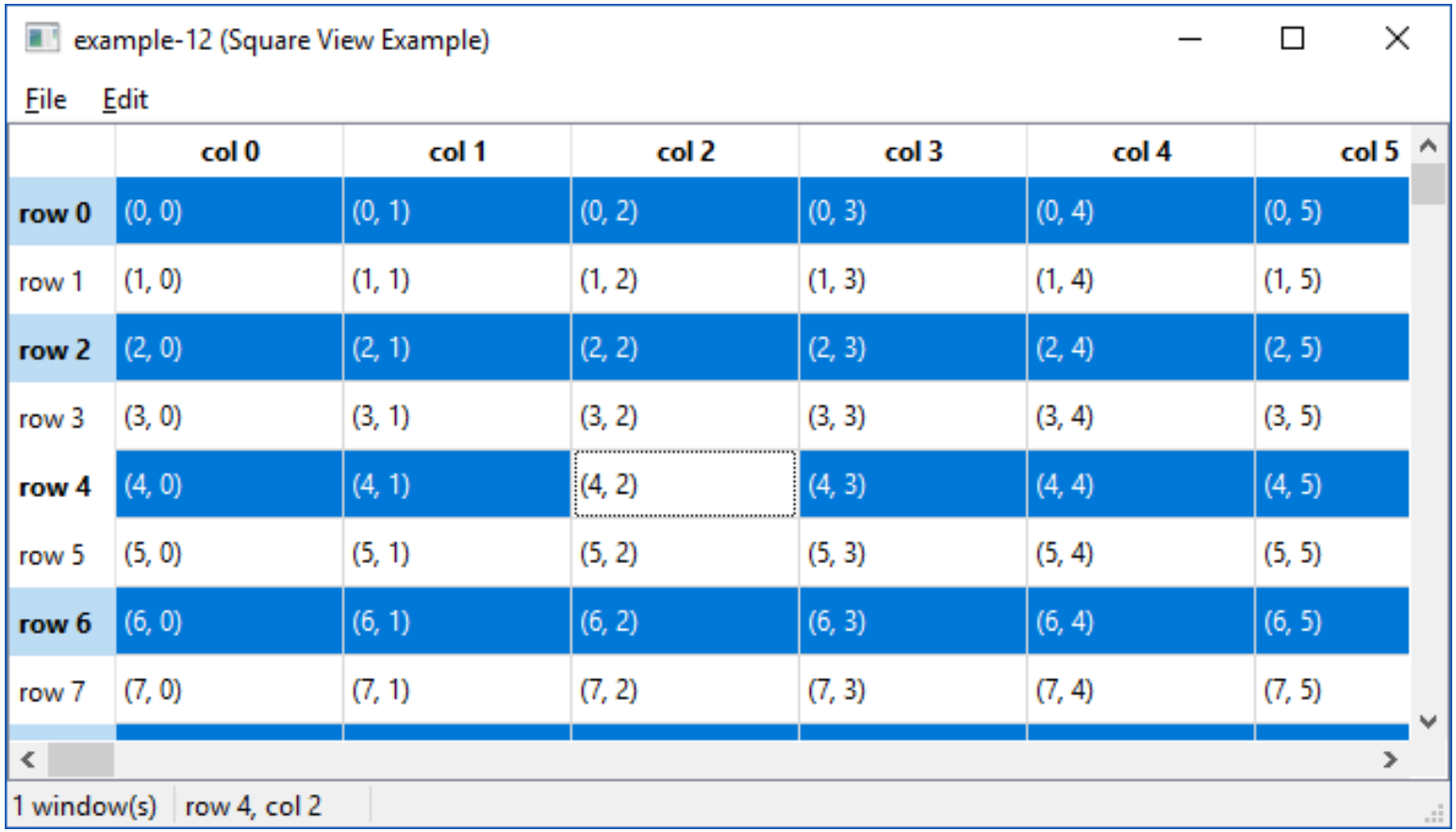

Figure 10: diaplaying a tabular model

Example

Example (square-model.h)

```
#include <QAbstractTableModel>

class SquareModel : public QAbstractTableModel
{
public:
  //
  virtual int rowCount(
    const QModelIndex &rcParent) const;
  virtual int columnCount(
    const QModelIndex &rcParent) const;
```

Example (cont.)

Example (square-model.h, end)

```
  virtual QVariant data(
    const QModelIndex &rcIndex, int nRole) const;
  virtual QVariant headerData(
    int nSection, Qt::Orientation nOrientation, int nRole) const;
};    // class SquareModel
```

Example (cont.)

Example (square-model.cpp)

```
#include "square-model.h"

int SquareModel::rowCount(
  const QModelIndex &rcParent) const
{
  return 100;
}

int SquareModel::columnCount(
  const QModelIndex &rcParent) const
{
  return 100;
}
```

Example (cont.)

Example (square-model.cpp, cont.)

```
QVariant SquareModel::data(
  const QModelIndex &rcIndex, int nRole) const
{
  if (rcIndex.isValid() && nRole == Qt::DisplayRole)
    return QString("(%1, %2)")
      .arg(rcIndex.row())
      .arg(rcIndex.column());
  else
    return QVariant();
}
```

Example (cont.)

Example (square-model.cpp, end)

```
QVariant SquareModel::headerData(
  int nSection, Qt::Orientation nOrientation, int nRole) const
{
  if (nRole != Qt::DisplayRole)
    return QVariant();
  //
  if (nOrientation == Qt::Horizontal)
    return QString("col %1")
      .arg(nSection);
  else
    return QString("row %1")
      .arg(nSection);
}
```

Example (cont.)

Example (main-window.cpp)

```
#include "main-window.h"

#include <QtWidgets>

MainWindow::MainWindow(
  QAbstractItemModel *pItemModel,
  QItemSelectionModel *pSelectionModel)
  : m_nWindows(1)
{
  setupUi(this);
  //
```

Example (cont.)

Example (main-window.cpp, cont.)

```
connect(
  actionClose, SIGNAL(triggered()),
  this, SLOT(close()));
connect(
  actionExit, SIGNAL(triggered()),
  qApp, SLOT(closeAllWindows()));
//
m_pTableView = new QTableView;
m_pTableView->setModel(pItemModel);
if (pSelectionModel)
  m_pTableView->setSelectionModel(pSelectionModel);
//
```

Example (cont.)

Example (main-window.cpp, cont.)

```
setCentralWidget(m_pTableView);
//
// ... Filling out the status bar
//
if (pSelectionModel == 0)    // if the first window
  updateStatusBar();
//
connect(
  m_pTableView->selectionModel(),
  &QItemSelectionModel::currentChanged,
  this,
  &MainWindow::onCurrentChanged);
```

Example (cont.)

Example (main-window.cpp, cont.)

```
  //
  QModelIndex indexCurrent =
    m_pTableView->selectionModel()->currentIndex();
  onCurrentChanged(indexCurrent, indexCurrent);
  //
  setAttribute(Qt::WA_DeleteOnClose);
}    // MainWindow::MainWindow()
```

Example (cont.)

Example (main-window.cpp, cont.)

```
void MainWindow::closeEvent(QCloseEvent *pEvent)
{
  pEvent->accept();
  //
  -- m_nWindows;
  updateAllNumWindows();
}
```

Example (cont.)

Example (main-window.cpp, cont.)

```
void MainWindow::on_actionNew_triggered()
{
  MainWindow *pMainWindow = new MainWindow(
    m_pTableView->model(),
    m_pTableView->selectionModel());
  //
  ++ m_nWindows;
  updateAllNumWindows();
  //
  pMainWindow->show();
}
```

Example (cont.)

Example (main-window.cpp, cont.)

```
void MainWindow::on_actionSelectAll_triggered()
{
  QAbstractItemModel *pModel = m_pTableView->model();
  QModelIndex indexFirst = pModel->index(0, 0);
  QModelIndex indexLast = pModel->index(
    pModel->rowCount() - 1,
    pModel->columnCount() - 1);
  QItemSelection selection(indexFirst, indexLast);
  //
  m_pTableView->selectionModel()->select(
    selection, QItemSelectionModel::Select);
}
```

Example (cont.)

Example (main-window.cpp, cont.)

```
void MainWindow::on_actionSelectNone_triggered()
{
  m_pTableView->selectionModel()->select(
    QModelIndex(), QItemSelectionModel::Clear);
}
```

Example (cont.)

Example (main-window.cpp, cont.)

```
void MainWindow::on_actionToggleRows_triggered()
{
  QAbstractItemModel *pModel = m_pTableView->model();
  QItemSelection selection;
  for (int i = 0, n = pModel->rowCount(); i < n; i += 2)
  {
    QModelIndex indexFirst = pModel->index(i, 0);
    QItemSelection selectionTemp(indexFirst, indexFirst);
    selection.merge(selectionTemp, QItemSelectionModel::Select);
  }
```

Example (cont.)

Example (main-window.cpp, cont.)

```
  //
  m_pTableView->selectionModel()->select(
    selection,
    QItemSelectionModel::Toggle |
    QItemSelectionModel::Rows);
}    //  MainWindow::on_actionToggleRows_triggered()
```

Example (cont.)

Example (main-window.cpp, cont.)

```
void MainWindow::onCurrentChanged(
  const QModelIndex &rcCurrent,
  const QModelIndex &rcPrevious)
{
  QString strLabel;
  if (rcCurrent.isValid())
    strLabel = QString("row %1, col %2")
      .arg(rcCurrent.row())
      .arg(rcCurrent.column());
  //
  m_pLabelCursorPos->setText(strLabel);
}
```

Example (cont.)

Example (main-window.cpp, cont.)

```
void MainWindow::updateStatusBar()
{
  QString strLabel;
  strLabel = QString("%1 window(s)")
    .arg(m_nWindows);
  //
  m_pLabelNumWindows->setText(strLabel);
}
```

Example (end)

Example (main-window.cpp, end)

```
void MainWindow::setNumWindows(int nWindows)
{
  m_nWindows = nWindows;
  updateStatusBar();
}

void MainWindow::updateAllNumWindows()
{
  foreach (QWidget *pWidget, QApplication::topLevelWidgets())
    if (MainWindow *pMainWindow =
      qobject_cast <MainWindow *> (pWidget))
      pMainWindow->setNumWindows(m_nWindows);
}
```

Editable Models

Turning On Editing (QAbstractItemModel)

```
virtual Qt::ItemFlags flags(
  const QModelIndex &rcIndex) const;

virtual bool setData(
  const QModelIndex &rcIndex, const QVariant &rcValue, int nRole);
```

NoItemFlags	ItemIsDragEnabled	ItemIsEnabled
ItemIsSelectable	ItemIsDropEnabled	ItemIsTristate
ItemIsEditable	ItemIsUserCheckable	ItemNeverHasChildren

Table 4: flags of a model element's state

Example

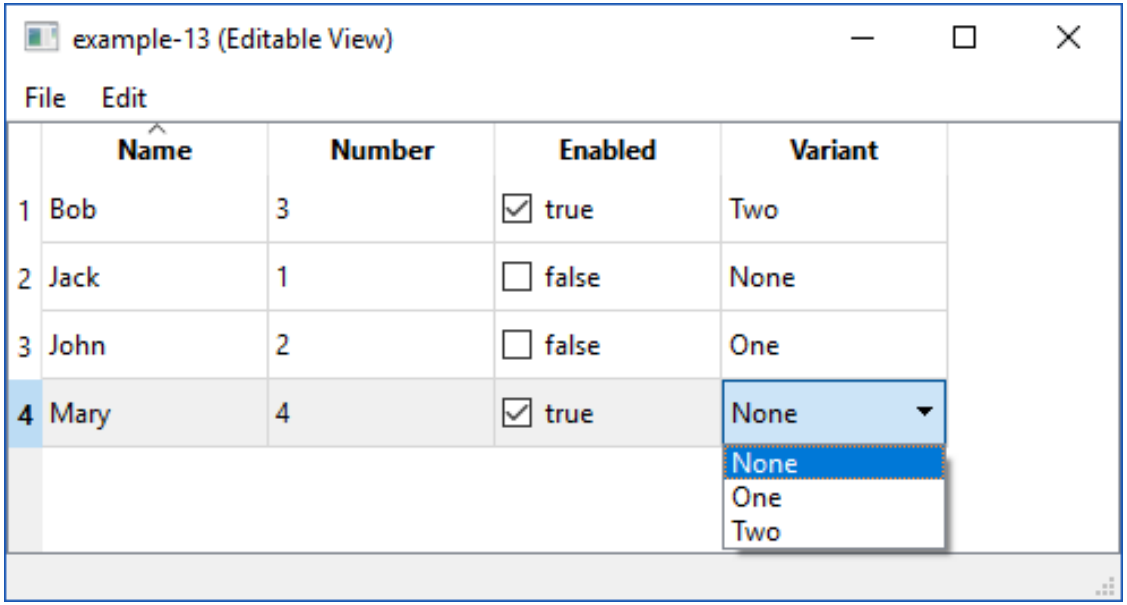

Figure 11: editing a model

dropMimeData() Method

QAbstractItemModel Method

```
virtual bool dropMimeData(
  const QMimeData *pData,
  Qt::DropAction nAction,
  int nRow,
  int nColumn,
  const QModelIndex &pParent);
```

IgnoreAction
CopyAction
MoveAction
LinkAction
...

Table 5: Qt::DropAction constants

Example

Example (combo-delegate.h)

```
#include <QStyledItemDelegate>

class ComboDelegate : public QStyledItemDelegate
{
public:
  //
  ComboDelegate(QObject *pParent = 0);
  //
```

Example (cont.)

Example (combo-delegate.h, cont.)

```
public:
  //
  virtual QWidget *createEditor(
    QWidget *pParent,
    const QStyleOptionViewItem &rcOption,
    const QModelIndex &rcIndex) const;
  virtual void setEditorData(
    QWidget *pEditor,
    const QModelIndex &rcIndex) const;
```

Example (cont.)

Example (combo-delegate.h, end)

```
  virtual void setModelData(
    QWidget *pEditor,
    QAbstractItemModel *pModel,
    const QModelIndex &rcIndex) const;
  virtual void updateEditorGeometry(
    QWidget *pEditor,
    const QStyleOptionViewItem &rcOption,
    const QModelIndex &rcIndex) const;
};    // class ComboDelegate
```

Example (cont.)

Example (combo-delegate.cpp)

```
#include "combo-delegate.h"

#include "table-model.h"

#include <QtWidgets>

ComboDelegate::ComboDelegate(QObject *pParent)
  : QStyledItemDelegate(pParent)
{
  //
}
```

Example (cont.)

Example (combo-delegate.cpp, cont.)

```
QWidget *ComboDelegate::createEditor(
  QWidget *pParent,
  const QStyleOptionViewItem &rcOption,
  const QModelIndex &rcIndex) const
{
  QComboBox *pComboBox = new QComboBox(pParent);
  const TableModel *pcModel =
    qobject_cast <const TableModel *> (rcIndex.model());
  if (pcModel)
    pComboBox->addItems(pcModel->getItemNames());
  //
  return pComboBox;
}
```

Example (cont.)

Example (combo-delegate.cpp, cont.)

```
void ComboDelegate::setEditorData(
  QWidget *pEditor,
  const QModelIndex &rcIndex) const
{
  const int cnValue = rcIndex.model()->data(
    rcIndex, Qt::EditRole).toInt();
  QComboBox *pComboBox =
    qobject_cast <QComboBox *> (pEditor);
  if (pComboBox)
    pComboBox->setCurrentIndex(cnValue);
}
```

Example (cont.)

Example (combo-delegate.cpp, cont.)

```
void ComboDelegate::setModelData(
  QWidget *pEditor,
  QAbstractItemModel *pModel,
  const QModelIndex &rcIndex) const
{
  QComboBox *pComboBox =
    qobject_cast <QComboBox *> (pEditor);
  if (pComboBox)
  {
    const int cnValue = pComboBox->currentIndex();
    pModel->setData(rcIndex, cnValue, Qt::EditRole);
  }
}
```

Example (cont.)

Example (combo-delegate.cpp, end)

```
void ComboDelegate::updateEditorGeometry(
  QWidget *pEditor,
  const QStyleOptionViewItem &rcOption,
  const QModelIndex &rcIndex) const
{
  pEditor->setGeometry(rcOption.rect);
}
```

Example (cont.)

Example (table-model.h)

```
#include <QAbstractTableModel>
#include <QList>
#include <QStringList>

class TableModel : public QAbstractTableModel
{
  Q_OBJECT
  //
public:
  //
  TableModel();
  //
```

Example (cont.)

Example (table-model.h, cont.)

```
public:
  //
  const QStringList &getItemNames() const;
  //
public:
  //
  virtual int rowCount(
    const QModelIndex &rcParent = QModelIndex()) const;
  virtual int columnCount(
    const QModelIndex &rcParent = QModelIndex()) const;
```

Example (cont.)

Example (table-model.h, cont.)

```
virtual QVariant data(
  const QModelIndex &rcIndex, int nRole) const;
virtual bool setData(
  const QModelIndex &rcIndex, const QVariant &rcValue, int nRole);
virtual QVariant headerData(
  int nSection, Qt::Orientation nOrientation, int nRole) const;
virtual Qt::DropActions supportedDropActions() const;
virtual bool insertRows(
  int nRow, int nCount, const QModelIndex &rcParent = QModelIndex());
virtual bool removeRows(
  int nRow, int nCount, const QModelIndex &rcParent = QModelIndex());
```

Example (cont.)

Example (table-model.h, cont.)

```
virtual bool moveRows(
  const QModelIndex &rcParentSource, int nRowSource, int nCount,
  const QModelIndex &rcParentDest, int nChildDest);
virtual Qt::ItemFlags flags(
  const QModelIndex &rcIndex) const;
virtual void sort(
  int nColumn, Qt::SortOrder nOrder = Qt::AscendingOrder);
//
```

Example (cont.)

Example (table-model.h, cont.)

```
private:
  //
  struct Data
  {
    QString m_strName;
    int m_nValue;
    bool m_bEnabled;
    int m_nVariant;
    //
```

Example (cont.)

Example (table-model.h, end)

```
    Data(
      const QString &rcName = QString(),
      int nValue = 0,
      bool bEnabled = false,
      int nVariant = 0);
  };    // struct Data
  //
  QList <Data> m_Data;
  QStringList m_Items;
};    // class TableModel
```

Example (cont.)

Example (table-model.cpp)

```
TableModel::Data::Data(
  const QString &rcName,
  int nValue,
  bool bEnabled,
  int nVariant)
  : m_strName(rcName),
    // ... initialize fields
    m_nVariant(nVariant)
{
  //
}
```

Example (cont.)

```
TableModel::TableModel()
{
  m_Items <<
    QString("None") <<
    QString("One") <<
    QString("Two");
  //
  m_Data <<
    // ... initialize data
    Data("Mary", 4, true);
}
```

Example (cont.)

Example (table-model.cpp, cont.)

```
const QStringList &TableModel::getItemNames() const
{
  return m_Items;
}
```

Example (cont.)

Example (table-model.cpp, cont.)

```
int TableModel::rowCount(
  const QModelIndex &rcParent) const
{
  return m_Data.size();
}

int TableModel::columnCount(
  const QModelIndex &rcParent) const
{
  return 4;
}
```

Example (cont.)

Example (table-model.cpp, cont.)

```
QVariant TableModel::data(
  const QModelIndex &rcIndex, int nRole) const
{
  if (rcIndex.isValid() &&
    rcIndex.row() < rowCount() &&
    rcIndex.column() < columnCount())
  {
    const Data &rcData = m_Data[rcIndex.row()];
```

Example (cont.)

Example (table-model.cpp, cont.)

```
switch (nRole)
{
  case Qt::DisplayRole:
  case Qt::EditRole:
    switch (rcIndex.column())
    {
      case 0:
        return rcData.m_strName;
      case 1:
        return rcData.m_nValue;
      case 2:
        return rcData.m_bEnabled;
```

Example (cont.)

Example (table-model.cpp, cont.)

```
    case 3:
      if (nRole == Qt::DisplayRole)
        return m_Items[rcData.m_nVariant];
      else
        return rcData.m_nVariant;
  }    // switch (rcIndex.column())
```

Example (cont.)

Example (table-model.cpp, cont.)

```
      case Qt::CheckStateRole:
        switch (rcIndex.column())
        {
          case 2:
            return
              rcData.m_bEnabled ?
              Qt::Checked : Qt::Unchecked;
        }
    }    // switch (nRole)
  }    // if (rcIndex.isValid() && ...)
  //
  return QString();
}    // TableModel::data()
```

Example (cont.)

Example (table-model.cpp, cont.)

```
bool TableModel::setData(
  const QModelIndex &rcIndex, const QVariant &rcValue, int nRole)
{
  if (rcIndex.isValid() &&
    rcIndex.row() < rowCount() &&
    rcIndex.column() < columnCount() &&
    (nRole == Qt::DisplayRole ||
     nRole == Qt::EditRole ||
     nRole == Qt::CheckStateRole))
  {
    Data &rData = m_Data[rcIndex.row()];
```

Example (cont.)

Example (table-model.cpp, cont.)

```
switch (nRole)
{
  case Qt::DisplayRole:
  case Qt::EditRole:
    switch (rcIndex.column())
    {
      case 0:
        rData.m_strName = rcValue.toString();
        break;
      case 1:
        rData.m_nValue = rcValue.toInt();
        break;
```

Example (cont.)

Example (table-model.cpp, cont.)

```
    case 3:
      rData.m_nVariant = rcValue.toInt();
      break;
    default:
      return false;
  }    // switch (rcIndex.column())
  break;
```

Example (cont.)

Example (table-model.cpp, cont.)

```
    case Qt::CheckStateRole:
      switch (rcIndex.column())
      {
        case 2:
          rData.m_bEnabled =
            (rcValue.toInt() == Qt::Checked);
          break;
        default:
          return false;
      }
  }    // switch (nRole)
```

Example (cont.)

Example (table-model.cpp, cont.)

```
    //
    emit dataChanged(rcIndex, rcIndex);
    //
    return true;
  }
  //
  return false;
}    // TableModel::setData()
```

Example (cont.)

Example (table-model.cpp, cont.)

```
QVariant TableModel::headerData(
  int nSection, Qt::Orientation nOrientation, int nRole) const
{
  if (nRole == Qt::DisplayRole &&
    nOrientation == Qt::Horizontal)
    switch (nSection)
    {
      case 0:
        return QString("Name");
      case 1:
        return QString("Number");
```

Example (cont.)

Example (table-model.cpp, cont.)

```
      case 2:
        return QString("Enabled");
      case 3:
        return QString("Variant");
    }    // switch (nSection)
  //
  return QAbstractTableModel::headerData(
    nSection, nOrientation, nRole);
}    // TableModel::headerData()
```

Example (cont.)

Example (table-model.cpp, cont.)

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

Example (cont.)

Example (table-model.cpp, cont.)

```
bool TableModel::insertRows(
  int nRow, int nCount, const QModelIndex &rcParent)
{
  if (nRow < 0)
    nRow = rowCount();
  //
  beginInsertRows(QModelIndex(), nRow, nRow + nCount - 1);
  //
  for (int i = 0; i < nCount; ++ i)
    m_Data.insert(nRow, Data());
  //
  endInsertRows();
```

Example (cont.)

Example (table-model.cpp, cont.)

```
    //
    return true;
}    // TableModel::insertRows()
```

Example (cont.)

Example (table-model.cpp, cont.)

```
bool TableModel::removeRows(
  int nRow, int nCount, const QModelIndex &rcParent)
{
  beginRemoveRows(QModelIndex(), nRow, nRow + nCount - 1);
  //
  for (int i = 0; i < nCount; ++ i)
    m_Data.removeAt(nRow);
  //
  endRemoveRows();
  //
  return true;
}
```

Example (cont.)

Example (table-model.cpp, cont.)

```
bool TableModel::moveRows(
  const QModelIndex &rcParentSource, int nRowSource, int nCount,
  const QModelIndex &rcParentDest, int nChildDest)
{
  beginMoveRows(
    QModelIndex(), nRowSource, nRowSource + nCount - 1,
    QModelIndex(), nChildDest + nCount - 1);
  //
  for (int i = 0; i < nCount; ++ i)
    m_Data.move(nRowSource, nChildDest + i);
  //
  endMoveRows();
```

Example (cont.)

Example (table-model.cpp, cont.)

```
    //
    return true;
}    // TableModel::moveRows()
```

Example (cont.)

Example (table-model.cpp, cont.)

```
Qt::ItemFlags TableModel::flags(
  const QModelIndex &rcIndex) const
{
  Qt::ItemFlags nReturn =
    QAbstractTableModel::flags(rcIndex);
  //
  nReturn |= Qt::ItemIsDropEnabled;
```

Example (cont.)

Example (table-model.cpp, cont.)

```
  if (rcIndex.isValid())
  {
    nReturn |= Qt::ItemIsDragEnabled;
    if (rcIndex.column() == 2)
      nReturn |= Qt::ItemIsUserCheckable;
    else
      nReturn |= Qt::ItemIsEditable;
  }
  //
  return nReturn;
}    // TableModel::flags()
```

Example (cont.)

Example (table-model.cpp, cont.)

```
void TableModel::sort(int nColumn, Qt::SortOrder nOrder)
{
  emit layoutAboutToBeChanged();
  //
  std::sort(
    m_Data.begin(), m_Data.end(),
    [=] (const Data &rcData1, const Data &rcData2)
    {
      switch (nColumn)
      {
```

Example (cont.)

Example (table-model.cpp, cont.)

```
switch (nColumn)
{
  case 0:
    switch (nOrder)
    {
      case Qt::AscendingOrder:
        return (rcData1.m_strName < rcData2.m_strName);
      case Qt::DescendingOrder:
        return (rcData1.m_strName > rcData2.m_strName);
    }
  case 1:
    // ...
```

Example (cont.)

Example (table-model.cpp, end)

```
        }    // switch (nColumn)
    });    // std::sort()
  //
  emit layoutChanged();
  //
}    // TableModel::sort()
```

Example (cont.)

Example (main-window.h)

```
class TableModel;
class QTableView;

class MainWindow :
  public QMainWindow, public Ui::MainWindow
{
  Q_OBJECT
  //
public:
  //
  MainWindow();
  //
```

Example (cont.)

Example (main-window.h, cont.)

```
private slots:
  //
  void on_actionAddRow_triggered();
  void on_actionDeleteSelectedRows_triggered();
  void on_actionMoveSelectedRowsFront_triggered();
  //
```

Example (cont.)

Example (main-window.h, end)

```
private:
  //
  TableModel *m_pTableModel;
  QTableView *m_pTableView;
  //
  typedef std::set <int> IntSet;
  //
  void getSelectedRows(IntSet &rRows);
};    // class MainWindow
```

Example (cont.)

Example (main-window.cpp)

```
MainWindow::MainWindow()
{
  setupUi(this);
  //
  m_pTableModel = new TableModel;
  m_pTableView = new QTableView;
  m_pTableView->setModel(m_pTableModel);
  ComboDelegate *pDelegate = new ComboDelegate(m_pTableModel);
  m_pTableView->setItemDelegateForColumn(3, pDelegate);
```

Example (cont.)

Example (main-window.cpp, cont.)

```
m_pTableView->setSelectionBehavior(QAbstractItemView::SelectRows);
m_pTableView->setSelectionMode(QAbstractItemView::ExtendedSelection);
m_pTableView->setDragEnabled(true);
m_pTableView->setAcceptDrops(true);
m_pTableView->setDropIndicatorShown(true);
m_pTableView->setSortingEnabled(true);
m_pTableView->sortByColumn(0, Qt::AscendingOrder);
```

Example (cont.)

Example (main-window.cpp, cont.)

```
  //
  setCentralWidget(m_pTableView);
  //
  connect(
    m_pActionExit, SIGNAL(triggered()),
    this, SLOT(close()));
}    // MainWindow::MainWindow()
```

Example (cont.)

Example (main-window.cpp, cont.)

```
void MainWindow::on_actionAddRow_triggered()
{
  const int cnRow = m_pTableModel->rowCount();
  m_pTableModel->insertRow(cnRow);
  QModelIndex indexNew = m_pTableModel->index(cnRow, 0);
  m_pTableView->setCurrentIndex(indexNew);
}
```

Example (cont.)

Example (main-window.cpp, cont.)

```
void MainWindow::on_actionDeleteSelectedRows_triggered()
{
  IntSet rows;
  getSelectedRows(rows);
  //
  IntSet::reverse_iterator
    i = rows.rbegin(),
    e = rows.rend();
  for (; i != e; ++ i)
    m_pTableModel->removeRow(*i);
}
```

Example (cont.)

Example (main-window.cpp, cont.)

```
void MainWindow::on_actionMoveSelectedRowsFront_triggered()
{
  IntSet rows;
  getSelectedRows(rows);
  //
  IntSet::iterator
    i = rows.begin(),
    e = rows.end();
  for (int n = 0; i != e; ++ i, ++ n)
    m_pTableModel->moveRow(
      QModelIndex(), *i, QModelIndex(), n);
}
```

Example (end)

Example (main-window.cpp, end)

```
void MainWindow::getSelectedRows(IntSet &rRows)
{
  QModelIndexList indexes =
    m_pTableView->selectionModel()->selectedIndexes();
  foreach (QModelIndex index, indexes)
    rRows.insert(index.row());
}
```