THE UNIVERSITY OF ALABAMA IN HUNTSVILLE

Lecture Qt013
Model View

Instructor: David J. Coe

CPE 353 – Software Design and Engineering

Department of Electrical and Computer Engineering

Outline

- Model-View Framework
- Directory View Example
- File Dialog Example
- Key Points

Model-View-Controller

THE UNIVERSITY OF ALABAMA IN HUNTSVILLE

- Model-View-Controller (MVC)
  - Classic design pattern
  - Model responsible for retrieving data and saving any data modifications
  - View responsible for rendering data for display
  - Controller handles editing of data

Model-View Framework

- Model-View Framework
  - Qt approach derived from MVC
  - **Model** responsible for retrieving data and saving any data modifications
  - **View** responsible for rendering data for display
  - **Delegate** assists with rendering and editing of data

- Key concept
  - Separation between data and the data display

Model-View Framework

- Model represents the set of data
  - Inherit from abstract class **QAbstractItemModel**
- Qt provides a number of predefined models such as
  - **QStringListModel**
  - **QSqlQueryModel**
  - **QSqlTableModel**
  - **QDirModel**

Model-View Framework

- One model can support multiple views
  - Inherit from abstract class **QAbstractItemView**
- Qt provides default views including
  - **QListView**
  - **QTableView**
  - **QTreeView**
- Multiple views are kept synchronized automatically
- Default delegate is provided for each view

Example: Directory View

```
// dirviews/main.cpp
// Molkentin, Book of Qt4
#include <QtGui>
int main(int argc, char* argv[])
{
          QApplication app(argc, argv);

          QDirModel dirModel;
          QWidget w;
          w.setWindowTitle(QObject::tr("Four directory views using one model"));
          QGridLayout *lay = new QGridLayout(&w);

          QListView *lv = new QListView;
          lay->addWidget(lv, 0, 0);
          lv->setModel(&dirModel);

          QListView *lvi = new QListView;
          lay->addWidget(lvi, 0, 1);
          lvi->setViewMode(QListView::IconMode);
          lvi->setModel(&dirModel);

          QTreeView *trv = new QTreeView;
          lay->addWidget(trv, 1, 0);
          trv->setModel(&dirModel);

          QTableView *tav = new QTableView;
          tav->setModel(&dirModel);
          lay->addWidget(tav, 1, 1);

          QModelIndex cwdIndex = dirModel.index(QDir::currentPath());
          lv->setRootIndex(cwdIndex);
          lvi->setRootIndex(cwdIndex);
          trv->setRootIndex(cwdIndex);
          tav->setRootIndex(cwdIndex);

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

Data model for file system

Multiple views of data within model

Example: File Dialog

```
// main.cpp

#include <QApplication>
#include <QtDebug>
#include "filedialog.h”
int main(int argc, char* argv[])
{
  QApplication app(argc, argv);
  FileDialog dialog;

  if ( dialog.exec() == QDialog::Accepted )
    qDebug() << dialog.selectedFiles();

  return 0;
}
```

**Example is from Daniel Molkentin,
The Book of Qt4: The Art of Building Qt Applications

Example: File Dialog

```
// filedialog.h

#ifndef FILEDIALOG_H
#define FILEDIALOG_H

#include "ui_filedialog.h”

class QModelIndex;
class QDirModel;
class QItemSelectionModel;

class FileDialog: public QDialog, private Ui::FileDialog
{
  Q_OBJECT

public:
    FileDialog(QWidget *parent = 0);
    QStringList selectedFiles();

protected slots:
    void switchToDir(const QModelIndex& index);   // Update all views as user navigates system
    void syncActive(const QModelIndex& index);    // Synchronize active item across all views
    void switchView();                            // Cycle through all three views

private:
    QItemSelectionModel *selModel;                // Will track items selected within view
    QDirModel *dirModel;                          // Data model
};

#endif // FILEDIALOG_H
```

Multiple Inheritance

**Example is from Daniel Molkentin,
The Book of Qt4: The Art of Building Qt Applications

Example: File Dialog

THE UNIVERSITY OF
ALABAMA IN HUNTSVILLE

```
// filedialog.cpp
#include <QDirModel>
#include <QItemSelectionModel>
#include "filedialog.h"

FileDialog::FileDialog(QWidget *parent)  : QDialog(parent)
{
  setupUi(this);

  dirModel = new QDirModel;
  selModel = new QItemSelectionModel(dirModel);

  listView->setModel(dirModel);
  treeView->setModel(dirModel);
  iconView->setModel(dirModel);

  listView->setSelectionModel(selModel);
  treeView->setSelectionModel(selModel);
  iconView->setSelectionModel(selModel);

  QModelIndex cwdIndex = dirModel->index(QDir::rootPath());    // Start at / for Linux, C:\ for Windows
  listView->setRootIndex(cwdIndex);
  treeView->setRootIndex(cwdIndex);
  iconView->setRootIndex(cwdIndex);

  for (int r = 0; r < dirModel->rowCount(QModelIndex()); ++r)
  {
    QModelIndex index = dirModel->index(r, 0, QModelIndex());
    if (index.isValid())
      comboBox->addItem(dirModel->fileIcon(index), dirModel->filePath(index));
  }
```

Sets model for view to present

Establish common selection model so that selection of an item within one view results in same item being selected in other views

Models have rows and columns. Each row represents a data item, and each column represents a property. So, each data item has an index with a row, column, and pointer.

**Example is from Daniel Molkentin,
The Book of Qt4: The Art of Building Qt Applications

Example: File Dialog

```
// filedialog.cpp - continued

  connect(listView, SIGNAL(activated(const QModelIndex&)), SLOT(switchToDir(const QModelIndex&)));
  connect(treeView, SIGNAL(activated(const QModelIndex&)), SLOT(switchToDir(const QModelIndex&)));
  connect(iconView, SIGNAL(activated(const QModelIndex&)), SLOT(switchToDir(const QModelIndex&)));

  connect(listView, SIGNAL(clicked(const QModelIndex&)), SLOT(syncActive(const QModelIndex&)));
  connect(treeView, SIGNAL(clicked(const QModelIndex&)), SLOT(syncActive(const QModelIndex&)));
  connect(iconView, SIGNAL(clicked(const QModelIndex&)), SLOT(syncActive(const QModelIndex&)));

  connect(switchButton, SIGNAL(clicked()), SLOT(switchView()));  // Responds to Toggle View

}

QStringList FileDialog::selectedFiles()
{
  QStringList fileNames;
  QModelIndexList indexes = selModel->selectedIndexes();

  foreach( QModelIndex index, indexes )
    fileNames.append( dirModel->filePath(index) );

  return fileNames;
}

void FileDialog::switchToDir(const QModelIndex& index)
{
  if (dirModel->isDir(index))
  {
    listView->setRootIndex(index);
    iconView->setRootIndex(index);
    treeView->setExpanded(index, true);
  }
}
```

Would be used in conjunction with the Open button

**Example is from Daniel Molkentin,
The Book of Qt4: The Art of Building Qt Applications

Example: File Dialog

```
// filedialog.cpp – continued

void FileDialog::syncActive(const QModelIndex& index)
{
  listView->setCurrentIndex(index);
  treeView->setCurrentIndex(index);
  iconView->setCurrentIndex(index);
}

void FileDialog::switchView()
{
  stackedWidget->setCurrentIndex( (stackedWidget->currentIndex()+1) % stackedWidget->count() );
}
```

Widget stack – stack of widgets where only one is visible at a time (established in Qt Designer)

**Example is from Daniel Molkentin, *The Book of Qt4: The Art of Building Qt Applications*

Key Points

- Model-view framework provides a way to separate the data from the display of the data
  - Allows for multiple ways of viewing the same data
- In most cases, the default delegates adequately display the data.
- See Qt Assistant and Qt Essentials – Widget Edition slides for details regarding the creation of custom delegates