Lecture 10. Implementing User Interface with QML

Cross-Platform Application Development

December 1, 2017

Main Definitions

Definitions

QML:	(Qt Meta Language) — a declarative language for user interface description. Paradigms: • Reactive programming; • Embedded JavaScript.
Qt Quick:	a QML module providing basic user interface elements to QML applications.
JSON:	(JavaScript Object Notation) — a text format for structured objects consisting of attributes and associated values.

Basic QML Components

Graphical

- Rectangle;
- Canvas;
- Image;
- . . .

Behavioral

- State;
- Transition;
- Animation;
- . . .

Declarative Technologies

Year	Technologies	Developer
2003 (?)	XUL	Mozilla
2008	WPF, XAML	Microsoft
2008	JavaFX	Sun Microsystems
2009	QML, Qt Quick	Nokia

Table 1: overview of GUI declarative languages

Import Statement

Definition (`import`)

import ⟨*Module_ URI*⟩ ⟨*Major*⟩.⟨*Minor*⟩ [as ⟨*Local_ Identifier*⟩]
import "⟨*Directory_ Path*⟩" [as ⟨*Local_ Identifier*⟩]
import "⟨*JavaScript_ File*⟩" as ⟨*Local_ Identifier*⟩

Examples

```
import QtQuick 2.7
import "../common"
import "colorservice.js" as Service
```

Hello Example

Example (Hello.qml)

```
import QtQuick 2.5

Rectangle
{
  id: root
  width: 200
  height: 200
  //
```

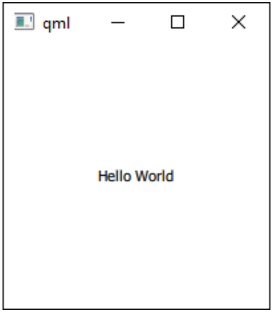
Example (Hello.qml, end)

```
  Text
  {
    anchors.centerIn: parent
    text: "Hello World"
  }
  //
  MouseArea
  {
    anchors.fill: parent
    onClicked: Qt.quit()
  }
}
```

Hello Example (end)

Example (launching)

```
> C:\Programs\Qt\Qt5.7.0_mingw\5.7\mingw53_32\bin\qml.exe Hello.qml
```



Hello World

Figure 1: main window of the QML application

QML Object Hierarchy

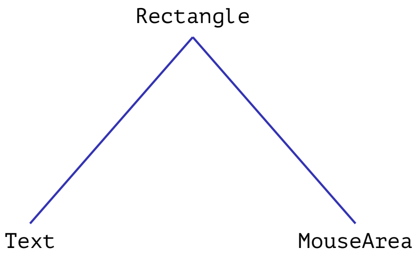


Figure 2: QML dynamic object tree

Quick View Example

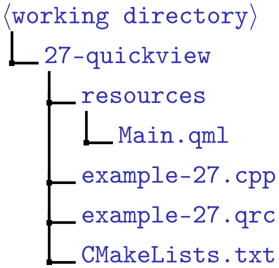


Figure 3: directory structure for the project with QML files

Quick View Example (cont.)

Example (CMakeLists.txt)

```
cmake_minimum_required(
  VERSION 2.8.11)

project(27-quickview)

find_package(
  Qt5 REQUIRED Gui Quick)

qt5_add_resources(
  QRC_WRAPPERS
  example-27.qrc)
```

Example (CMakeLists.txt, end)

```
add_executable(
  example-27 WIN32
  example-27.cpp
  ${QRC_WRAPPERS})

target_link_libraries(
  example-27 Qt5::Gui Qt5::Quick)
```

Quick View Example (cont.)

Example (resources/Main.qml)

```
import QtQuick 2.5

Item
{
  width: 400
  height: 400
  //
  Text
  {
    anchors.centerIn: parent
    text: "Hello World"
  }
}
```

Quick View Example (cont.)

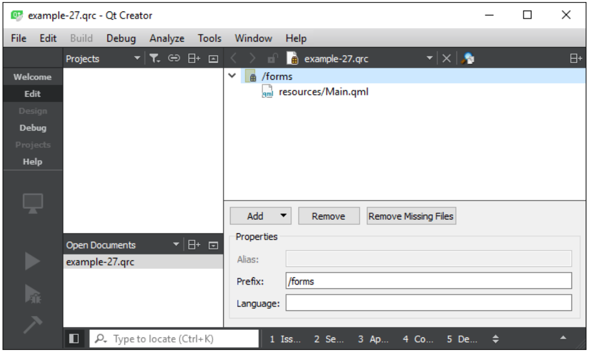


Figure 4: including QML source files to the Qt resources

Quick View Example (cont.)

Example (`example-27.qrc`)

```
<RCC>
    <qresource prefix="/forms">
        <file>resources/Main.qml</file>
    </qresource>
</RCC>
```

Quick View Example (cont.)

Example (example-27.cpp)

```
#include <QGuiApplication>
#include <QQuickView>
#include <QQmlError>
#include <QtDebug>

int main(int nArgC, char *apszArgV[])
{
  QGuiApplication app(nArgC, apszArgV);
  //
```

Quick View Example (end)

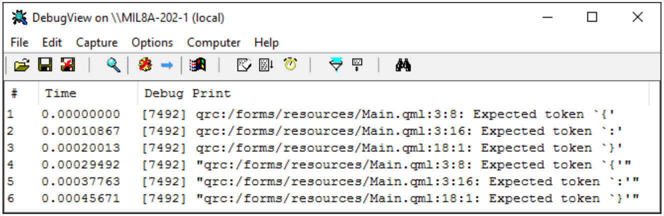
Example (example-27.cpp, end)

```
  QUrl url(QStringLiteral("qrc:/forms/resources/Main.qml"));
  QQuickView view(url);
  QList <QQmlError> errors = view.errors();
  foreach (const QQmlError &rcError, errors)
    qDebug() << rcError.toString();
  //
  view.show();
  //
  return app.exec();
}    // main()
```

Debug Output

Example (syntax error)

```
inport QtQuick 2.5
```


DebugView on \\MIL8A-202-1 (local)
File Edit Capture Options Computer Help

```
#  Time        Debug Print
1  0.00000000  [7492] qrc:/forms/resources/Main.qml:3:8: Expected token `{'
2  0.00010867  [7492] qrc:/forms/resources/Main.qml:3:16: Expected token `:'
3  0.00020013  [7492] qrc:/forms/resources/Main.qml:18:1: Expected token `}'
4  0.00029492  [7492] "qrc:/forms/resources/Main.qml:3:8: Expected token `{'"
5  0.00037763  [7492] "qrc:/forms/resources/Main.qml:3:16: Expected token `:'"
6  0.00045671  [7492] "qrc:/forms/resources/Main.qml:18:1: Expected token `}'"
```

Figure 5: an optput of DebugView utility (`http://sysinternals.com`)

QML Engine Example

Example (CMakeLists.txt)

```
cmake_minimum_required(VERSION 2.8.11)

project(28-appengine)

find_package(Qt5 REQUIRED Gui Qml)

# ...
```

QML Engine View Example (cont.)

Example (example-28.cpp)

```
#include <QGuiApplication>
#include <QQmlApplicationEngine>

int main(int nArgC, char *apszArgV[])
{
  QGuiApplication app(nArgC, apszArgV);
  //
  QQmlApplicationEngine engine(
    QUrl("qrc:/forms/resources/Main.qml"));
  //
  return app.exec();
}
```

QML Engine Example (end)

Example (resources/Main.qml)

```
import QtQuick 2.5
import QtQuick.Window 2.0

Window
{
  visible: true
  width: 200
  height: 200
```

Example (resources/Main.qml, end)

```
  //
  MouseArea
  {
    anchors.fill: parent
    onClicked: Qt.quit()
  }
}
```

Value Assignment/Initialization vs. Binding

Example

```
Rectangle
{
  width: 300      // static value, initialization
  height: width   // property binding, initialization
  color: "blue"
  MouseArea
  {
    anchors.fill: parent
    onClicked: parent.color = "green"    // imperative value assignment
  }
}
```

Object Attributes

Types of Attributes

- The `id` attribute;
- `property` attribute;
  - property alias
- `signal` attribute;
- Signal handler attribute;
- Method attribute;
- Attached signal handler attribute.
- Attached property attribute.

Example

```
Rectangle
{
  id: root
  //  ...
  Text
  {
    id: text
    //  ^ lower case
    // ...
  }
}
```

Object Attributes

Types of Attributes

- The id attribute;
- property attribute;
  - property alias
- signal attribute;
- Signal handler attribute;
- Method attribute;
- Attached signal handler attribute.
- Attached property attribute.

Example

```
Rectangle
{
  width: 300
  height: 300
  property int number
  default property var data: "AB"
  onNumberChanged:
    console.log(
      "New: " + number.toString())
  Component.onCompleted:
    number = 100 * Math.random()
}
```

Object Attributes

Types of Attributes

- The id attribute;
- property attribute;
  - property alias
- signal attribute;
- Signal handler attribute;
- Method attribute;
- Attached signal handler attribute.
- Attached property attribute.

Example

```
Rectangle
{
  width: 300
  height: 300
  property alias text: text1.text
  Component.onCompleted:
    text = "Hello World"
  Text
  {
    id: text1
    anchors.centerIn: parent
  }
}
```

Object Attributes

Types of Attributes

- The `id` attribute;
- `property` attribute;
  - property alias
- `signal` attribute;
- Signal handler attribute;
- Method attribute;
- Attached signal handler attribute.
- Attached property attribute.

Example

```
Rectangle
{
  width: 300
  height: 300
  signal newPoint(point pt)
  onNewPoint:
    console.log(pt.toString())
  Component.onCompleted:
    newPoint(
      Qt.point(
        width * Math.random(),
        height * Math.random()))
}
```

Object Attributes

Types of Attributes

- The id attribute;
- property attribute;
    - property alias
- signal attribute;
- Signal handler attribute;
- Method attribute;
- Attached signal handler attribute.
- Attached property attribute.

Example

```
Rectangle
{
  width: 300
  height: 300
  signal newPoint(point pt)
  onNewPoint:
    console.log(pt.toString())
  Component.onCompleted:
    newPoint(
      Qt.point(
        width * Math.random(),
        height * Math.random()))
}
```

Object Attributes

Types of Attributes

- The `id` attribute;
- `property` attribute;
  - property alias
- `signal` attribute;
- Signal handler attribute;
- Method attribute;
- Attached signal handler attribute.
- Attached property attribute.

Example

```
Rectangle
{
  width: 300
  height: 300
  function process(n)
  {
    console.log(n.toString())
  }
  Component.onCompleted:
    process(100 * Math.random())
}
```

Object Attributes

Types of Attributes

- The id attribute;
- property attribute;
  - property alias
- signal attribute;
- Signal handler attribute;
- Method attribute;
- Attached signal handler attribute.
- Attached property attribute.

Example

```
Rectangle
{
  width: 300
  height: 300
  function process(n)
  {
    console.log(n.toString())
  }
  Component.onCompleted:
    process(100 * Math.random())
}
```

Object Attributes

Types of Attributes

- The id attribute;
- property attribute;
  - property alias
- signal attribute;
- Signal handler attribute;
- Method attribute;
- Attached signal handler attribute.
- Attached property attribute.

Example

```
ListView
{
  width: 300; height: 300
  model: 40
  delegate: Rectangle
  {
    width: ListView.view.width
    height: 20
    border.color:
      ListView.isCurrentItem ?
      "blue" : "white"
  }
}
```

QML Types

bool	int	real	double	string
url	list <Object>	enumeration	var	

Table 2: basic types

point	size	rect	color	date	font
vector2d	vector3d	vector4d	quaternion	matrix4x4	

Table 3: types provided by QtQuick module

Part of QML Object Types Hierarchy

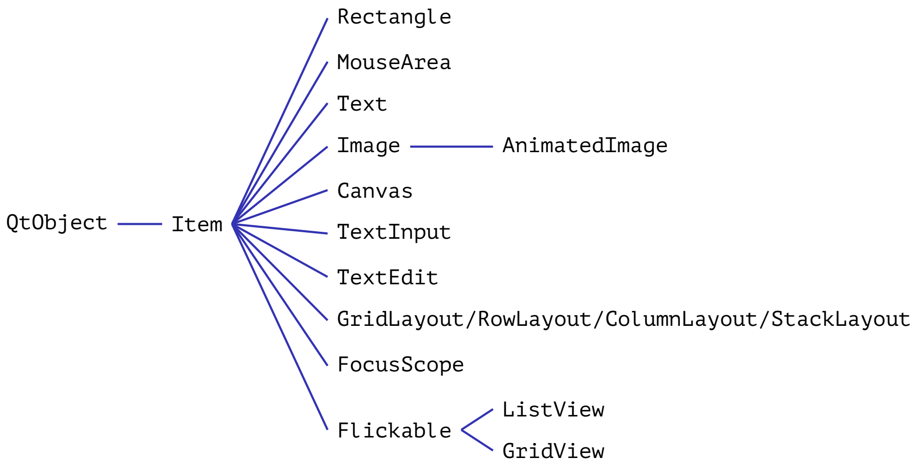


Figure 6: part of QML visual components hierarchy

Item Properties

x	y	z	width	height
rotation	scale	opacity		

Table 4: `real` properties

visible	enabled	focus	activeFocusOnTab
activeFocus	clip	smooth	antialiasing

Table 5: `bool` properties

children: list <Item>	resources: list <Object>
parent: Item	data: list <Object>
state: string	states: list <State>

Table 6: object properties

Item.anchors Group Properties

horizontalCenterOffset	verticalCenterOffset	baselineOffset
margins	topMargin	bottomMargin
leftMargin	rightMargin	

Table 7: real properties

top	bottom	left	right
horizontalCenter	verticalCenter	baseline	

Table 8: AnchorLine properties

fill	centerIn

Table 9: Item properties

Canvas Example

Example (Main.qml)

```
import QtQuick 2.7

Canvas
{
  width: 640
  height: 480
  //
  function rnd(a, b)
  {
    return a + (b - a) * Math.random()
  }
  //
```

Canvas Example (cont.)

Example (`Main.qml`, cont.)

```
onPaint:
{
  var ctx = getContext("2d")
  ctx.fillStyle = 'rgb(0.1, 0.1, 0.1)'
  ctx.fillRect(0, 0, width, height)
  for (var i = 0; i < 100; i ++)
  {
    var x = rnd(0, width)
    var y = rnd(0, height)
    var d = rnd(1, 5)
```

Canvas Example (cont.)

Example (`Main.qml`, end)

```
      var r = rnd(.5, 1)
      var g = rnd(.5, 1)
      var b = rnd(.5, 1)
      var c = Qt.rgba(r, g, b, 1.)
      ctx.fillStyle = c
      ctx.beginPath()
      ctx.ellipse(x, y, d, d)
      ctx.fill()
    }    // for (var i = 0; i < 100; i ++)
  }    // onPaint
}    // Canvas
```

Canvas Example (end)

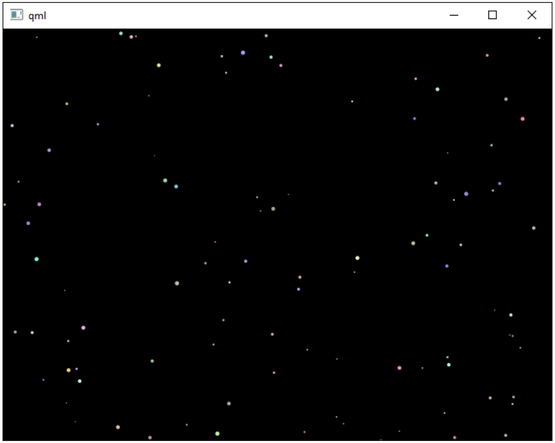


Figure 7: QML application using Canvas

states Example

Example (Main.qml)

```
import QtQuick 2.7

Rectangle
{
  id: root
  width: 200
  height: 200
  focus: true
  state: "red"
```

Example (Main.qml, cont.)

```
states:
[
  State
  {
    name: "red"
    PropertyChanges
    {
      target: root
      color: "red"
    }
  },
```

States Example (cont.)

Example (Main.qml, cont.)

```
    State
    {
      name: "yellow"
      PropertyChanges
      {
        target: root
        color: "yellow"
      }
    },
```

Example (Main.qml, cont.)

```
    State
    {
      name: "green"
      PropertyChanges
      {
        target: root
        color: "green"
      }
    }
  ]
```

States Example (end)

Example (Main.qml, cont.)

```
Keys.onPressed:
{
  switch (event.key)
  {
    case Qt.Key_Left:
      switch (state)
      {
        case "red":
          state = "green"
          break;
        case "green":
          state = "yellow"
```

Example (Main.qml, end)

```
            break;
          case "yellow":
            state = "red"
            break;
        }     // switch (state)
        break;
    }     // switch (event.key)
  }     // Keys.onPressed
  Keys.onReturnPressed:
    state = "red"
}     // Rectangle
```

Interaction Between C++ and QML

`QObject` Attributes Available from QML

- Properties;
- Signals;
- Slots;
- `Q_INVOKABLE` methods;
- Exposed enumerations (`Q_ENUM`).

Counter Example

Example (counter.h)

```
#ifndef COUNTER_H__
#define COUNTER_H__

#include <QObject>

class QTimer;

class Counter : public QObject
{
  Q_OBJECT
  Q_PROPERTY(int value READ value WRITE setValue NOTIFY valueChanged)
  //
```

Counter Example (cont.)

Example (counter.h, cont.)

```
public:
  //
  enum Direction
  {
    Ascending,
    Descending
  };
  //
  Q_ENUM(Direction)
  //
  explicit Counter(QObject *pParent = Q_NULLPTR);
  //
```

Counter Example (cont.)

Example (counter.h, cont.)

```
  int value() const;
  void setValue(int nValue);
  Q_INVOKABLE void setDirection(Direction nDirection);
  //
public slots:
  //
  void stepValue();
  //
signals:
  //
  void valueChanged(int nValue);
  //
```

Counter Example (cont.)

Example (counter.h, end)

```
private:
  //
  QTimer *m_pTimer;
  //
  int m_nValue;
  Direction m_nDirection;
};    // class Counter

#endif    // COUNTER_H__
```

Counter Example (cont.)

Example (counter.cpp)

```
#include "counter.h"

#include <QObject>
#include <QTimer>

Counter::Counter(QObject *pParent)
  : QObject(pParent),
    //
    m_pTimer(new QTimer(this)),
    //
    m_nValue(0),
    m_nDirection(Ascending)
```

Example (counter.cpp, cont.)

```
{
  connect(
    m_pTimer, SIGNAL(timeout()),
    this, SLOT(stepValue()));
  //
  m_pTimer->setInterval(1000);
  m_pTimer->setSingleShot(false);
  m_pTimer->start();
}    // Counter::Counter()
```

Counter Example (cont.)

Example (counter.cpp, cont.)

```
int Counter::value() const
{
  return m_nValue;
}
```

Example (counter.cpp, cont.)

```
void Counter::setValue(int nValue)
{
  if (nValue != m_nValue)
  {
    m_nValue = nValue;
    emit valueChanged(nValue);
  }
}
```

Counter Example (cont.)

Example (counter.cpp, cont.)

```
void Counter::setDirection(
  Direction nDirection)
{
  m_nDirection = nDirection;
}
```

Example (counter.cpp, end)

```
void Counter::stepValue()
{
  switch (m_nDirection)
  {
    case Ascending:
      setValue(m_nValue + 1);
      break;
    case Descending:
      setValue(m_nValue - 1);
      break;
  }
}    // Counter::stepValue()
```

Counter Example (cont.)

Example (example-29.cpp)

```
#include "counter.h"

#include <QGuiApplication>
#include <QQuickView>

int main(int nArgC, char *apszArgV[])
{
  QGuiApplication app(nArgC, apszArgV);
  //
```

Counter Example (cont.)

Example (example-29.cpp, end)

```
  qmlRegisterType <Counter> ("it.mmcs.counter", 1, 0, "Counter");
  //
  QUrl url(QStringLiteral("qrc:/forms/resources/Main.qml"));
  QQuickView view(url);
  //
  view.show();
  //
  return app.exec();
}    // main()
```

Counter Example (cont.)

Example (resources/Main.qml)

```
import QtQuick 2.5
import it.mmcs.counter 1.0

Rectangle
{
  id: root
  width: 200
  height: width
  color: "steelblue"
  border.width: 5
  border.color: Qt.lighter(color)
  //
```

Example (resources/Main.qml, cont.)

```
property bool ascending: true
//
Text
{
  id: text
  anchors.centerIn: parent
  text: counter.value
  color: "white"
  //
```

Counter Example (cont.)

Example (resources/Main.qml, cont.)

```
    font
    {
      family: "Arial"
      pointSize: 30
      bold: true
    }
  }
  //
```

Example (resources/Main.qml, cont.)

```
MouseArea
{
  anchors.fill: parent
  //
  onClicked:
  {
    ascending = !ascending
    var direction =
      ascending ?
      Counter.Ascending :
      Counter.Descending
```

Counter Example (cont.)

Example (resources/Main.qml, end)

```
      counter.setDirection(direction)
    }
  }
  //
  Counter
  {
    id: counter
    onValueChanged: root.color =
      Qt.rgba(Math.random(), 0.1, 0.5, 1.0)
  }
}
```

Counter Example (end)

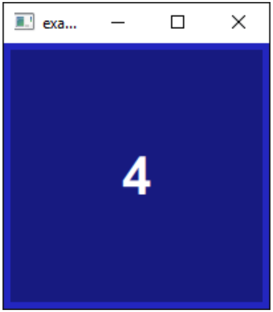


Figure 8: main window of QML Counter application

Buttons Example

```
⟨working directory⟩
└─ 30-quickview
   ├─ imports
   │  ├─ happy.png
   │  ├─ sad.png
   │  ├─ Main.qml
   │  └─ MyButton.qml
   ├─ example-30.cpp
   ├─ example-30.qrc
   └─ CMakeLists.txt
```

Figure 9: directory structure for the project with QML files

Buttons Example (cont.)

Example (example-30.qrc)

```
<RCC>
    <qresource prefix="/qt-project.org">
        <file>imports/happy.png</file>
        <file>imports/Main.qml</file>
        <file>imports/MyButton.qml</file>
        <file>imports/sad.png</file>
    </qresource>
</RCC>
```

Buttons Example (cont.)

Example (imports/MyButton.qml)

```
import QtQuick 2.7

Rectangle
{
  property alias text: text.text
  property alias imageSource: image.source
  signal clicked
  //
```

Buttons Example (cont.)

Example (MyButton.qml, cont.)

```
id: root
color: "lightgray"
border
{
  width: 3
  color: Qt.darker(color)
}
//
```

Example (MyButton.qml, cont.)

```
Image
{
  id: image
  height: parent.height - 6
  width: height
  anchors
  {
    left: parent.left
    top: parent.top
    margins: 3
  }
}
```

Buttons Example (cont.)

Example (imports/MyButton.qml, cont.)

```
//
Text
{
  id: text
  horizontalAlignment: Text.AlignHCenter
  anchors
  {
    left: image.right
    right: parent.right
    verticalCenter: parent.verticalCenter
  }
}
//
```

Buttons Example (cont.)

Example (imports/MyButton.qml, cont.)

```
MouseArea
{
  anchors.fill: parent
  //
  property color oldColor
  //
```

Buttons Example (cont.)

Example (`imports/MyButton.qml`, end)

```
    onPressed:
    {
      oldColor = root.color
      root.color = Qt.darker(oldColor)
    }
    //
    onReleased: root.color = oldColor
    //
    onClicked: root.clicked()
  }     // MouseArea
}    // Rectangle
```

Buttons Example (cont.)

Example (`imports/Main.qml`)

```
import QtQuick 2.7
import QtQuick.Window 2.0
import QtQuick.Layouts 1.3
import QtQuick.Controls 1.4

Window
{
  width: 200
  height: 200
  visible: true
  //
```

Buttons Example (cont.)

Example (imports/Main.qml, cont.)

```
ColumnLayout
{
  anchors.fill: parent
  focus: true
  spacing: 3
  //
  property real preferredHeight: height / 3 - 3
  //
  function addText(text)
  {
    edit.text += text + "\n"
    edit.cursorPosition = edit.text.length
  }
```

Buttons Example (cont.)

Example (imports/Main.qml, cont.)

```
//
MyButton
{
  text: "Button 1"
  imageSource: "happy.png"
  Layout.fillWidth: true
  Layout.preferredHeight: parent.preferredHeight
  onClicked: parent.addText(this.text)
}
```

Buttons Example (cont.)

Example (`imports/Main.qml`, cont.)

```
//
MyButton
{
  text: "Button 2"
  imageSource: "sad.png"
  Layout.fillWidth: true
  Layout.preferredHeight: parent.preferredHeight
  onClicked: parent.addText(this.text)
}
```

Buttons Example (cont.)

Example (imports/Main.qml, end)

```
    //
    TextArea
    {
      id: edit
      readOnly: true
      Layout.fillWidth: true
      Layout.preferredHeight: parent.preferredHeight
    }
  }     // ColumnLayout
}     // Window
```

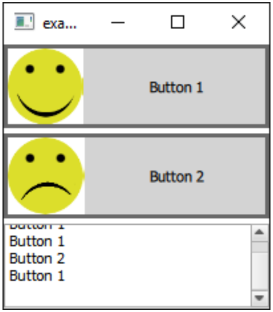
Buttons Example (cont.)

Example (example-30.cpp)

```
#include <QGuiApplication>
#include <QQmlApplicationEngine>

int main(int nArgC, char *apszArgV[])
{
  QGuiApplication app(nArgC, apszArgV);
  //
  QUrl url(QStringLiteral("qrc:/qt-project.org/imports/Main.qml"));
  QQmlApplicationEngine engine(url);
  //
  return app.exec();
}    // main()
```

Buttons Example (end)



```
Button 1
Button 1
Button 2
Button 1
```

Figure 10: main window of the QML application with embedded resources

Model Example

Example (Main.qml)

```
import QtQuick 2.7

Rectangle
{
  width: 240
  height: 320
  color: "white"
  //
```

Example (Main.qml, cont.)

```
  ListView
  {
    id: listView
    anchors.fill: parent
    anchors.margins: 4
    spacing: 4
    focus: true
    clip: true
    //
```

Model Example (cont.)

Example (Main.qml, cont.)

```
    model: listModel
    delegate: delegateComponent
    highlight:
      highlightComponent
    header: headerComponent
    footer: footerComponent
  }    // ListView
  //
  ListModel
  {
    id: listModel
    //
```

Example (Main.qml, cont.)

```
    ListElement
    {
      name: "Item Zero"
    }
    //
    Component.onCompleted:
    {
      for (var i = 1; i < 101; ++ i)
        listModel.append(
          { "name": "Item " + i })
    }
  }    // ListModel
```

Model Example (cont.)

Example (`Main.qml`, cont.)

```
//
Component
{
  id: delegateComponent
  //
  Item
  {
    width: parent.width
    height: 20
```

Example (`Main.qml`, cont.)

```
    //
    Text
    {
      anchors.centerIn: parent
      text:
        index + ") " + modelData
    }    // Text
  }    // Item
}    // Component
//
```

Model Example (cont.)

Example (`Main.qml`, cont.)

```
Component
{
  id: highlightComponent
  //
  Rectangle
  {
    width: parent.width
    color: "lightgreen"
  }
}
//
```

Model Example (cont.)

Example (Main.qml, cont.)

```
Component
{
  id: headerComponent
  //
  Rectangle
  {
    width: ListView.view.width
    height: 20
    color: "silver"
    //
```

Example (Main.qml, cont.)

```
      Text
      {
        anchors.centerIn: parent
        text: "Items"
      }     // Text
    }     // Rectangle
  }     // Component
//
```

Model Example (cont.)

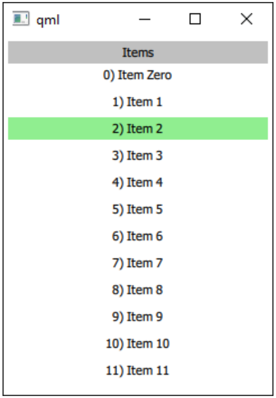
Example (`Main.qml`, cont.)

```
Component
{
  id: footerComponent
  //
  Rectangle
  {
    width: ListView.view.width
    height: 20
    color: "lightyellow"
    //
```

Example (`Main.qml`, end)

```
      Text
      {
        anchors.centerIn: parent
        text:
          "Total:  " +
          listView.count
      }    // Text
    }    // Rectangle
  }    // Component
}    // Rectangle
```

Model Example (end)



0) Item Zero
1) Item 1
2) Item 2
3) Item 3
4) Item 4
5) Item 5
6) Item 6
7) Item 7
8) Item 8
9) Item 9
10) Item 10
11) Item 11

Figure 11: main window of the list view QML application