CS241 Principles and Practice of Problem Solving

Lecture 20: GUI Programming with Qt III

Yuye Ling, Ph.D.

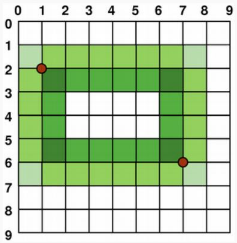
Shanghai Jiao Tong University
John Hopcroft Center for Computer Science

November 21, 2019

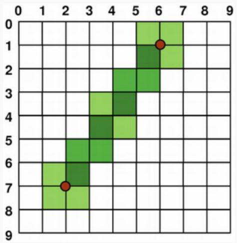
Copyright Notice

Anti-aliasing

We could improve the “rendering” by turning on anti-aliasing



```
QPainter painter(this);
painter.setRenderHint(
    QPainter::Antialiasing);
painter.setPen(Qt::darkGreen);
// Using the (x  y  w  h) overload
painter.drawRect(1, 2, 6, 4);
```



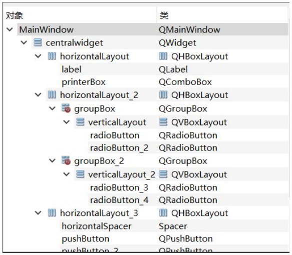
```
QPainter painter(this);
painter.setRenderHint(
    QPainter::Antialiasing);
painter.setPen(Qt::darkGreen);
painter.drawLine(2, 7, 6, 1);
```

Question: what might be the problem for this type of practice?

- In the last lecture, we have discussed the objects model and widgets.

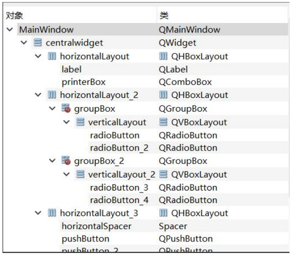
- In the last lecture, we have discussed the objects model and widgets.
- We know that the Qt organizes the widget by using

- In the last lecture, we have discussed the objects model and widgets.
- We know that the Qt organizes the widget by using a hierarchical structure (parent child relationship)



对象	类
MainWindow	QMainWindow
centralwidget	QWidget
horizontalLayout	QHBoxLayout
label	QLabel
printerBox	QComboBox
horizontalLayout_2	QHBoxLayout
groupBox	QGroupBox
verticalLayout	QVBoxLayout
radioButton	QRadioButton
radioButton_2	QRadioButton
groupBox_2	QGroupBox
verticalLayout_2	QVBoxLayout
radioButton_3	QRadioButton
radioButton_4	QRadioButton
horizontalLayout_3	QHBoxLayout
horizontalSpacer	Spacer
pushButton	QPushButton
pushButton_2	QPushButton

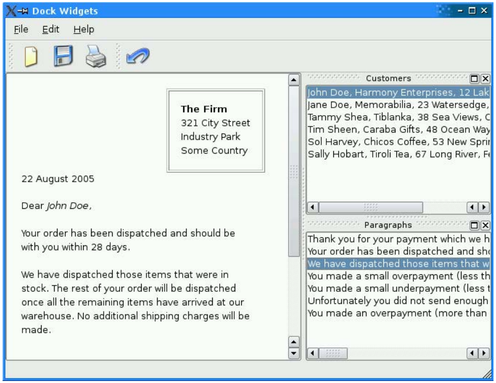
- In the last lecture, we have discussed the objects model and widgets.
- We know that the Qt organizes the widget by using a hierarchical structure (parent child relationship)



对象	类
MainWindow	QMainWindow
centralwidget	QWidget
horizontalLayout	QHBoxLayout
label	QLabel
printerBox	QComboBox
horizontalLayout_2	QHBoxLayout
groupBox	QGroupBox
verticalLayout	QVBoxLayout
radioButton	QRadioButton
radioButton_2	QRadioButton
groupBox_2	QGroupBox
verticalLayout_2	QVBoxLayout
radioButton_3	QRadioButton
radioButton_4	QRadioButton
horizontalLayout_3	QHBoxLayout
horizontalSpacer	Spacer
pushButton	QPushButton
pushButton_2	QPushButton

- What if a widget has no parent?
- It will become a standalone window.

A typical Main Window



The Firm
321 City Street
Industry Park
Some Country

22 August 2005

Dear *John Doe*,

Your order has been dispatched and should be with you within 28 days.

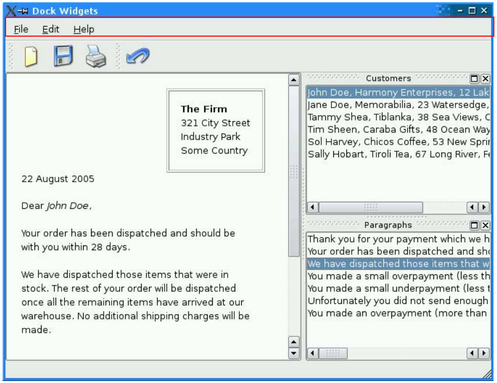
We have dispatched those items that were in stock. The rest of your order will be dispatched once all the remaining items have arrived at our warehouse. No additional shipping charges will be made.

John Doe, Harmony Enterprises, 12 Lak
Jane Doe, Memorabilia, 23 Watersedge,
Tammy Shea, Tiblanka, 38 Sea Views, C
Tim Sheen, Caraba Gifts, 48 Ocean Way
Sol Harvey, Chicos Coffee, 53 New Sprir
Sally Hobart, Tiroli Tea, 67 Long River, Fe

Thank you for your payment which we h
Your order has been dispatched and sho
We have dispatched those items that w
You made a small overpayment (less th
You made a small underpayment (less t
Unfortunately you did not send enough
You made an overpayment (more than

A typical Main Window

- Manu bar



The Firm
321 City Street
Industry Park
Some Country

22 August 2005

Dear *John Doe*,

Your order has been dispatched and should be with you within 28 days.

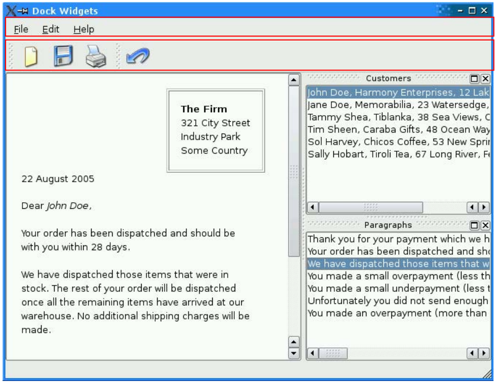
We have dispatched those items that were in stock. The rest of your order will be dispatched once all the remaining items have arrived at our warehouse. No additional shipping charges will be made.

John Doe, Harmony Enterprises, 12 Lak
Jane Doe, Memorabilia, 23 Watersedge,
Tammy Shea, Tiblanka, 38 Sea Views, C
Tim Sheen, Caraba Gifts, 48 Ocean Way
Sol Harvey, Chicos Coffee, 53 New Sprin
Sally Hobart, Tiroli Tea, 67 Long River, Fe

Thank you for your payment which we h
Your order has been dispatched and sho
We have dispatched those items that w
You made a small overpayment (less th
You made a small underpayment (less t
Unfortunately you did not send enough
You made an overpayment (more than

A typical Main Window

- Manu bar
- Tool bar



The Firm
321 City Street
Industry Park
Some Country

22 August 2005

Dear *John Doe*,

Your order has been dispatched and should be with you within 28 days.

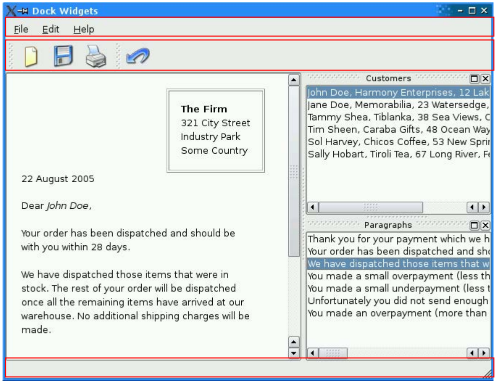
We have dispatched those items that were in stock. The rest of your order will be dispatched once all the remaining items have arrived at our warehouse. No additional shipping charges will be made.

John Doe, Harmony Enterprises, 12 Lak
Jane Doe, Memorabilia, 23 Watersedge,
Tammy Shea, Tiblanka, 38 Sea Views, C
Tim Sheen, Caraba Gifts, 48 Ocean Way
Sol Harvey, Chicos Coffee, 53 New Sprir
Sally Hobart, Tiroli Tea, 67 Long River, Fe

Thank you for your payment which we h
Your order has been dispatched and sho
We have dispatched those items that w
You made a small overpayment (less th
You made a small underpayment (less t
Unfortunately you did not send enough
You made an overpayment (more than

A typical Main Window

- Manu bar
- Tool bar
- Status bar



The Firm
321 City Street
Industry Park
Some Country

22 August 2005

Dear *John Doe*,

Your order has been dispatched and should be with you within 28 days.

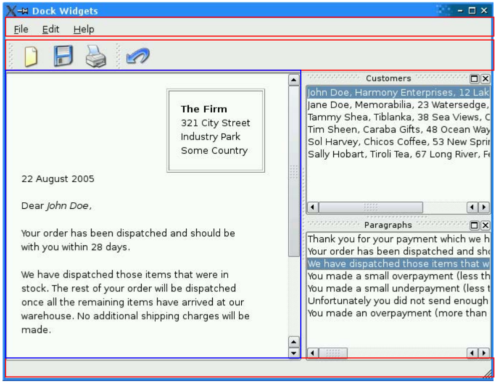
We have dispatched those items that were in stock. The rest of your order will be dispatched once all the remaining items have arrived at our warehouse. No additional shipping charges will be made.

John Doe, Harmony Enterprises, 12 Lak
Jane Doe, Memorabilia, 23 Watersedge,
Tammy Shea, Tiblanka, 38 Sea Views, C
Tim Sheen, Caraba Gifts, 48 Ocean Way
Sol Harvey, Chicos Coffee, 53 New Sprin
Sally Hobart, Tiroli Tea, 67 Long River, Fe

Thank you for your payment which we h
Your order has been dispatched and sh
We have dispatched those items that w
You made a small overpayment (less th
You made a small underpayment (less t
Unfortunately you did not send enough
You made an overpayment (more than

A typical Main Window

- Manu bar
- Tool bar
- Status bar
- Central widget



The Firm
321 City Street
Industry Park
Some Country

22 August 2005

Dear *John Doe*,

Your order has been dispatched and should be with you within 28 days.

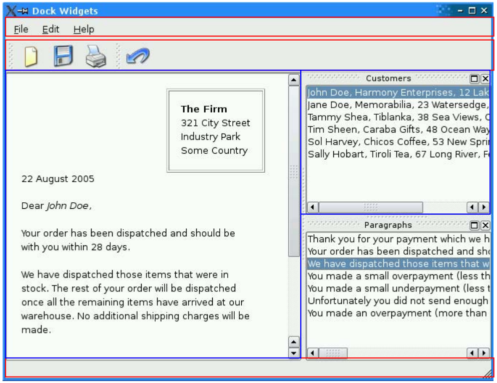
We have dispatched those items that were in stock. The rest of your order will be dispatched once all the remaining items have arrived at our warehouse. No additional shipping charges will be made.

John Doe, Harmony Enterprises, 12 Lak
Jane Doe, Memorabilia, 23 Watersedge,
Tammy Shea, Tiblanka, 38 Sea Views, C
Tim Sheen, Caraba Gifts, 48 Ocean Way
Sol Harvey, Chicos Coffee, 53 New Sprin
Sally Hobart, Tiroli Tea, 67 Long River, Fe

Thank you for your payment which we h
Your order has been dispatched and sho
We have dispatched those items that w
You made a small overpayment (less th
You made a small underpayment (less t
Unfortunately you did not send enough
You made an overpayment (more than

A typical Main Window

- Manu bar
- Tool bar
- Status bar
- Central widget
- Dockable window



The Firm
321 City Street
Industry Park
Some Country

22 August 2005

Dear *John Doe*,

Your order has been dispatched and should be with you within 28 days.

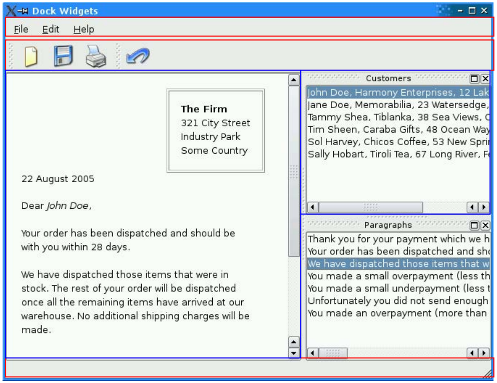
We have dispatched those items that were in stock. The rest of your order will be dispatched once all the remaining items have arrived at our warehouse. No additional shipping charges will be made.

John Doe, Harmony Enterprises, 12 Lak
Jane Doe, Memorabilia, 23 Watersedge,
Tammy Shea, Tiblanka, 38 Sea Views, C
Tim Sheen, Caraba Gifts, 48 Ocean Way
Sol Harvey, Chicos Coffee, 53 New Sprii
Sally Hobart, Tiroli Tea, 67 Long River, F

Thank you for your payment which we h
Your order has been dispatched and sho
We have dispatched those items that w
You made a small overpayment (less th
You made a small underpayment (less t
Unfortunately you did not send enough
You made an overpayment (more than

A typical Main Window

- Manu bar
- Tool bar
- Status bar
- Central widget
- Dockable window



The Firm
321 City Street
Industry Park
Some Country

22 August 2005

Dear *John Doe*,

Your order has been dispatched and should be with you within 28 days.

We have dispatched those items that were in stock. The rest of your order will be dispatched once all the remaining items have arrived at our warehouse. No additional shipping charges will be made.

John Doe, Harmony Enterprises, 12 Lak
Jane Doe, Memorabilia, 23 Watersedge,
Tammy Shea, Tiblanka, 38 Sea Views, C
Tim Sheen, Caraba Gifts, 48 Ocean Way
Sol Harvey, Chicos Coffee, 53 New Sprin
Sally Hobart, Tiroli Tea, 67 Long River, F

Thank you for your payment which we h
Your order has been dispatched and sho
We have dispatched those items that w
You made a small overpayment (less th
You made a small underpayment (less t
Unfortunately you did not send enough
You made an overpayment (more than

In Qt, `QMainWindow` class provide a one-stop template for all the gradients mentioned above.

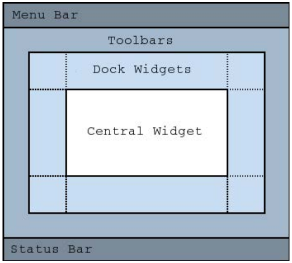
QMainWindow

Has its own layout

- Central Widget

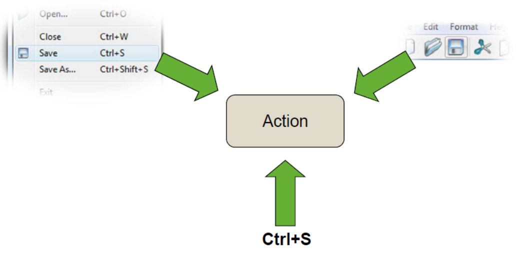
```
QMainWindow::setCentralWidget(
      widget)
```

- `QMenuBar`
- `QToolBar`
- `QDockWidget`
- `QStatusBar`



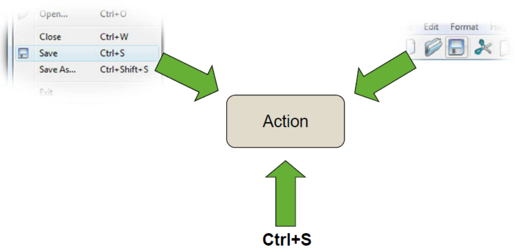
Introducing QAction

- Many UI elements refer to the same user action



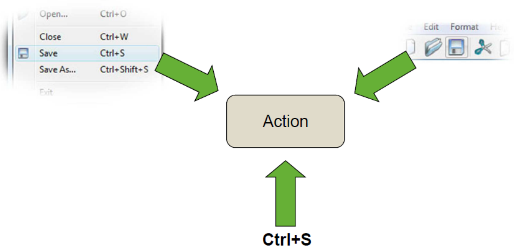
Introducing QAction

- Action is an abstract user interface command
- Many UI elements refer to the same user action



Introducing QAction

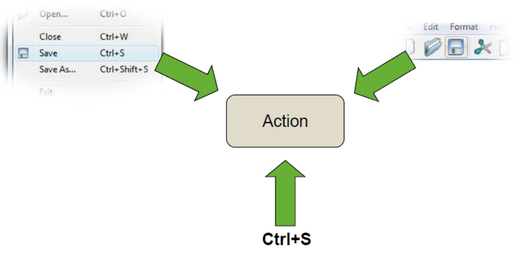
- Action is an abstract user interface command
- Many UI elements refer to the same user action



- A `QAction` object can represent all these access ways
  - and hold tool tips, status bar hints, etc

Introducing QAction

- Action is an abstract user interface command
- Many UI elements refer to the same user action



- A `QAction` object can represent all these access ways
  - and hold tool tips, status bar hints, etc
- Question: why do we want to do this?

Creating QAction

- Recall the problem we were confronted in last lecture (“disable on the GUI”)

Creating QAction

- Recall the problem we were confronted in last lecture (“disable on the GUI”)
- A `QAction` encapsulates all settings needed for menus, tool bars and keyboard shortcuts.

Creating QAction

- Recall the problem we were confronted in last lecture ("disable on the GUI")
- A `QAction` encapsulates all settings needed for menus, tool bars and keyboard shortcuts. You couldn't add it to a random push button!

Creating QAction

- Recall the problem we were confronted in last lecture ("disable on the GUI")
- A `QAction` encapsulates all settings needed for menus, tool bars and keyboard shortcuts. You couldn't add it to a random push button!
- Emits signal triggered on execution

Creating QAction

- Recall the problem we were confronted in last lecture ("disable on the GUI")
- A `QAction` encapsulates all settings needed for menus, tool bars and keyboard shortcuts. You couldn't add it to a random push button!
- Emits signal triggered on execution
- Connected slot performs action defined by `QAction`

Creating QAction

- Recall the problem we were confronted in last lecture ("disable on the GUI")
- A `QAction` encapsulates all settings needed for menus, tool bars and keyboard shortcuts. You couldn't add it to a random push button!
- Emits signal triggered on execution
- Connected slot performs action defined by `QAction`

```
void MainWindow::setupActions() {
    QAction* action = new QAction(tr("Open ..."), this);
    action->setIcon(QIcon(":/images/open.png"));
    action->setShortcut(QKeySequence::Open);
    action->setStatusTip(tr("Open file"));
    connect(action, SIGNAL(triggered()), this, SLOT(onOpen()));
    menu->addAction(action);
    toolbar->addAction(action);
    ...

```

QAction capabilities

- `setEnabled(bool)`
  - Enables and disables actions in menu and toolbars.

QAction capabilities

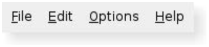
- `setEnabled(bool)`
  - Enables and disables actions in menu and toolbars. A safer operation

QAction capabilities

- `setEnabled(bool)`
  - Enables and disables actions in menu and toolbars. A safer operation
- `setCheckable(bool)`
  - Switches checkable state (on/off)
  - `setChecked(bool)` toggles checked state

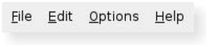
Creating menu bar

- `QMenuBar`: a horizontal menu bar



Creating menu bar

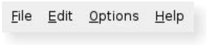
- `QMenuBar`: a horizontal menu bar



- `QMenu`: represents a menu

Creating menu bar

- QMenuBar: a horizontal menu bar

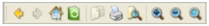


- QMenu: represents a menu
- QAction: menu items added to QMenu

```
void MainWindow::setupMenuBar() {
    QMenuBar* bar = menuBar();
    QMenu* menu = bar->addMenu(tr("&File"));
    menu->addAction(action);
    menu->addSeparator();
    QMenu* subMenu = menu->addMenu(tr("Sub Menu"));
    ...

```

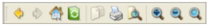
Creating tool bar

- Tool bar is a *movable panel* ...
  - Contains set of controls



Creating tool bar

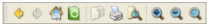
- Tool bar is a *movable panel* ...
  - Contains set of controls



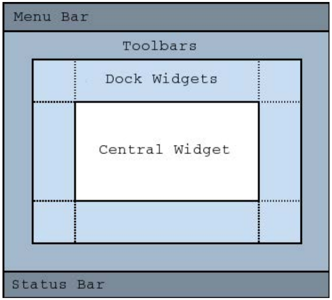
  - Can be horizontal or vertical

Creating tool bar

- Tool bar is a *movable panel* ...
  - Contains set of controls



  - Can be horizontal or vertical



Tool bar implementation

Implemented in Qt by QToolBar class

- `QMainWindow::addToolbar(toolbar)`
  - Adds toolbar to main window

Tool bar implementation

Implemented in Qt by QToolBar class

- `QMainWindow::addToolbar(toolbar)`
  - Adds toolbar to main window
- `QMainWindow::addToolBarBreak()`
  - Adds section splitter

Tool bar implementation

Implemented in Qt by QToolBar class

- `QMainWindow::addToolbar(toolbar)`
  - Adds toolbar to main window
- `QMainWindow::addToolBarBreak()`
  - Adds section splitter
- `QToolBar::addAction(action)`
  - Adds action to toolbar

Tool bar implementation

Implemented in Qt by QToolBar class

- `QMainWindow::addToolbar(toolbar)`
  - Adds toolbar to main window
- `QMainWindow::addToolBarBreak()`
  - Adds section splitter
- `QToolBar::addAction(action)`
  - Adds action to toolbar
- `QToolBar::addWidget(widget)`
  - Adds widget to toolbar

Tool bar implementation

Implemented in Qt by QToolBar class

- `QMainWindow::addToolbar(toolbar)`
  - Adds toolbar to main window
- `QMainWindow::addToolBarBreak()`
  - Adds section splitter
- `QToolBar::addAction(action)`
  - Adds action to toolbar
- `QToolBar::addWidget(widget)`
  - Adds widget to toolbar

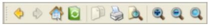
```
void MainWindow::setupToolBar() {
    QToolBar* bar = addToolBar(tr("File"));
    bar->addAction(action);
    bar->addSeparator();
    bar->addWidget(new QLineEdit(tr("Find ...")));

```

QToolButton

Question: do you think the codes in the last slide would work?

QToolButton

Question: do you think the codes in the last slide would work?



QToolButton

Question: do you think the codes in the last slide would work?

- We need to use `QToolButton` to get access to the `action` we just defined

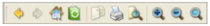
QToolButton

Question: do you think the codes in the last slide would work?

- We need to use `QToolButton` to get access to the `action` we just defined
- Question: can we use `QPushButton` instead?

QToolButton

Question: do you think the codes in the last slide would work?



- We need to use `QToolButton` to get access to the `action` we just defined
- Question: can we use `QPushButton` instead?
- Yes and no.

QToolButton

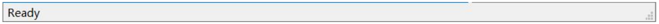
Question: do you think the codes in the last slide would work?

- We need to use `QToolButton` to get access to the `action` we just defined
- Question: can we use `QPushButton` instead?
- Yes and no.

```
QToolButton* button = new QToolButton(this);
button->setDefaultAction(action);
// Can have a menu
button->setMenu(menu);
// Shows menu indicator on button
button->setPopupMode(QToolButton::MenuButtonPopup);
// Control over text + icon placements
button->setToolButtonStyle(Qt::ToolButtonTextUnderIcon);

```

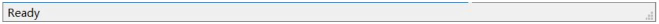
The status bar

- Horizontal bar: Suitable for presenting status information


Ready

The status bar

- Horizontal bar: Suitable for presenting status information

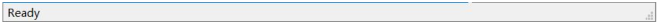


Ready

- `showMessage(message, timeout)`
  - Displays temporary message for specified milli-seconds

The status bar

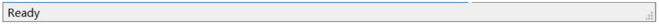
- Horizontal bar: Suitable for presenting status information


Ready

- `showMessage(message, timeout)`
  - Displays temporary message for specified milli-seconds
- `clearMessage()`
  - Removes any temporary message

The status bar

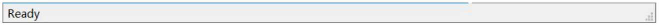
- Horizontal bar: Suitable for presenting status information



- `showMessage(message, timeout)`
  - Displays temporary message for specified milli-seconds
- `clearMessage()`
  - Removes any temporary message
- `addWidget()` or `addPermanentWidget()`
  - Normal, permanent messages displayed by widget

The status bar

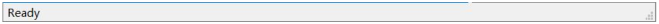
- Horizontal bar: Suitable for presenting status information


Ready

- `showMessage(message, timeout)`
  - Displays temporary message for specified milli-seconds
- `clearMessage()`
  - Removes any temporary message
- `addWidget()` or `addPermanentWidget()`
  - Normal, permanent messages displayed by widget
  - For example: progress bar

The status bar

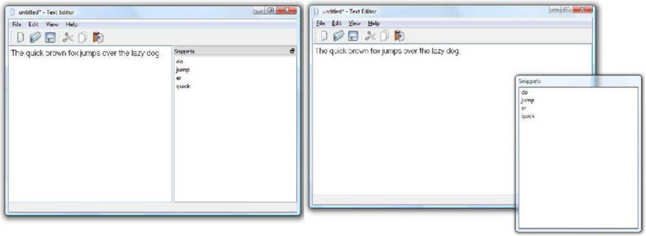
- Horizontal bar: Suitable for presenting status information



- `showMessage(message, timeout)`
  - Displays temporary message for specified milli-seconds
- `clearMessage()`
  - Removes any temporary message
- `addWidget()` or `addPermanentWidget()`
  - Normal, permanent messages displayed by widget
  - For example: progress bar

```
void MainWindow::createStatusBar() {
    QStatusBar* bar = statusBar();
    bar->showMessage(tr("Ready"));
    bar->addWidget(new QLabel(tr("Label on StatusBar")));

```

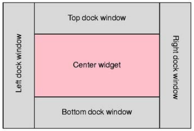
Dock widgets



- Dock widgets are detachable widgets placed around the edges of a `QMainWindow`
- Simply place your widget inside a `QDockWidget`
- `QMainWindow::addDockWidget` adds the docks to the window

Creating Dock Windows

- Window docked into main window
- `Qt::DockWidgetArea` enum
  - Left, Right, Top, Bottom dock areas
- `QMainWindow::setCorner(corner,area)`
  - Sets area to occupy specified corner
- `QMainWindow::setDockOptions(options)`
  - Specifies docking behavior (animated, nested, tabbed, ...)



```
void MainWindow::createDockWidget() {
    QDockWidget *dock = new QDockWidget(tr("Title"), this);
    dock->setAllowedAreas(Qt::LeftDockWidgetArea);
    QListWidget *widget = new QListWidget(dock);
    dock->setWidget(widget);
    addDockWidget(Qt::LeftDockWidgetArea, dock);
}

```

There is an excellent tutorial available online (in Chinese)

Files and file systems

Challenges

Referring to files and directories in a cross platform manner poses a number of problems.

Files and file systems

Challenges

Referring to files and directories in a cross platform manner poses a number of problems. Why?

Files and file systems

Challenges

Referring to files and directories in a cross platform manner poses a number of problems. Why?

- Does the system have drives, or just a root?

Files and file systems

Challenges

Referring to files and directories in a cross platform manner poses a number of problems. Why?

- Does the system have drives, or just a root?
- Are paths separated by "/" or "\"?

Files and file systems

Challenges

Referring to files and directories in a cross platform manner poses a number of problems. Why?

- Does the system have drives, or just a root?
- Are paths separated by "/" or "\"?
- Where does the system store temporary files?

Files and file systems

Challenges

Referring to files and directories in a cross platform manner poses a number of problems. Why?

- Does the system have drives, or just a root?
- Are paths separated by “/” or “\”?
- Where does the system store temporary files?
- Where does the user store documents?

Files and file systems

Challenges

Referring to files and directories in a cross platform manner poses a number of problems. Why?

- Does the system have drives, or just a root?
- Are paths separated by "/" or "\"?
- Where does the system store temporary files?
- Where does the user store documents?
- Where is the application stored?

Path

- Use the `QDir` class to handle paths

```
QDir d = QDir("C:/");

```

Path

- Use the `QDir` class to handle paths

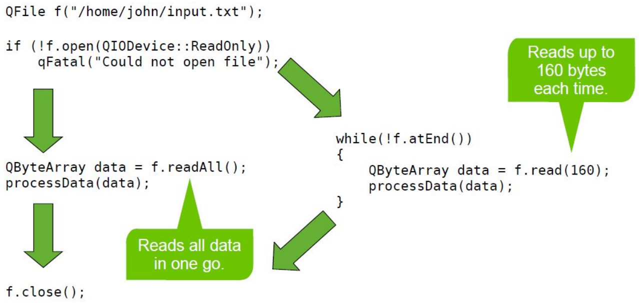
```
QDir d = QDir("C:/");

```

- Or, use the static methods to initialize

```
QDir d = QDir::root(); // C:/ on windows
QDir::current() // Current directory
QDir::home() // Home directory
QDir::temp() // Temporary directory
// Executable directory path
QDir(QApplication::applicationDirPath())

```

Opening and reading files

The `QFile` is used to access files



```
QFile f("/home/john/input.txt");

if (!f.open(QIODevice::ReadOnly))
    qFatal("Could not open file");

QByteArray data = f.readAll();
processData(data);

f.close();
```

```
while(!f.atEnd())
{
    QByteArray data = f.read(160);
    processData(data);
}
```

Writing to files

- When writing files, open it in `WriteOnly` mode and use the write method to add data to the file

Writing to files

- When writing files, open it in `WriteOnly` mode and use the write method to add data to the file

```
QFile f("/home/john/input.txt");
if (!f.open(QIODevice::WriteOnly))
    qFatal("Could not open file");
QByteArray data = createData();
f.write(data);
f.close();

```

Writing to files

- When writing files, open it in `WriteOnly` mode and use the write method to add data to the file

```
QFile f("/home/john/input.txt");
if (!f.open(QIODevice::WriteOnly))
    qFatal("Could not open file");
QByteArray data = createData();
f.write(data);
f.close();

```

- Files can also be opened in `ReadWrite` mode

Writing to files

- When writing files, open it in `WriteOnly` mode and use the write method to add data to the file

```
QFile f("/home/john/input.txt");
if (!f.open(QIODevice::WriteOnly))
    qFatal("Could not open file");
QByteArray data = createData();
f.write(data);
f.close();

```

- Files can also be opened in `ReadWrite` mode
- The flags `Append` or `Truncate` can be used in combination with write-enabled modes to either append data to the file or to truncate it (i.e. clear the file from its previous contents)

Writing to files

- When writing files, open it in `WriteOnly` mode and use the write method to add data to the file

```
QFile f("/home/john/input.txt");
if (!f.open(QIODevice::WriteOnly))
    qFatal("Could not open file");
QByteArray data = createData();
f.write(data);
f.close();

```

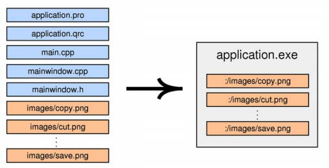
- Files can also be opened in `ReadWrite` mode
- The flags `Append` or `Truncate` can be used in combination with write-enabled modes to either append data to the file or to truncate it (i.e. clear the file from its previous contents)
- Question: what else can we rely on to interact with files?

Resource system

- Unlike CLI, GUI have, for example, a lot of icon pictures. Where are we supposed to put those resource files?

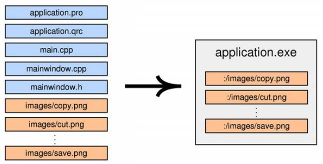
Resource system

- Unlike CLI, GUI have, for example, a lot of icon pictures. Where are we supposed to put those resource files?
- Qt solution: Putting icons in a resource file lets Qt embed them into the executable



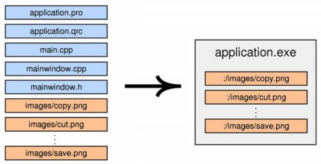
Resource system

- Unlike CLI, GUI have, for example, a lot of icon pictures. Where are we supposed to put those resource files?
- Qt solution: Putting icons in a resource file lets Qt embed them into the executable
  - Avoid having to deploy multiple files



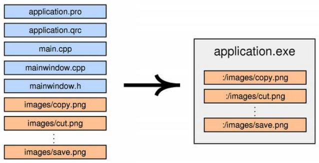
Resource system

- Unlike CLI, GUI have, for example, a lot of icon pictures. Where are we supposed to put those resource files?
- Qt solution: Putting icons in a resource file lets Qt embed them into the executable
  - Avoid having to deploy multiple files
  - No need to try to determine the path for the icons for each specific install type



Resource system

- Unlike CLI, GUI have, for example, a lot of icon pictures. Where are we supposed to put those resource files?
- Qt solution: Putting icons in a resource file lets Qt embed them into the executable
  - Avoid having to deploy multiple files
  - No need to try to determine the path for the icons for each specific install type
  - You can add anything into resources, not only icons



Ways of deployment

- Question: now we have compiled all the resources into an executable, can we just distribute this single executable?

Ways of deployment

- Question: now we have compiled all the resources into an executable, can we just distribute this single executable?
- Now really.

Ways of deployment

- Question: now we have compiled all the resources into an executable, can we just distribute this single executable?
- Now really. Distinction between resources and dependents.

Ways of deployment

- Question: now we have compiled all the resources into an executable, can we just distribute this single executable?
- Now really. Distinction between resources and dependents.
- The executable depends on Qt libraries to function.

Ways of deployment

- Question: now we have compiled all the resources into an executable, can we just distribute this single executable?
- Now really. Distinction between resources and dependents.
- The executable depends on Qt libraries to function.

Static Linking:

Shared Libraries:

Ways of deployment

- Question: now we have compiled all the resources into an executable, can we just distribute this single executable?
- Now really. Distinction between resources and dependents.
- The executable depends on Qt libraries to function.

Static Linking: Results in stand-alone executable

Shared Libraries: Qt has to be pre-installed

Ways of deployment

- Question: now we have compiled all the resources into an executable, can we just distribute this single executable?
- Now really. Distinction between resources and dependents.
- The executable depends on Qt libraries to function.

Static Linking: Results in stand-alone executable

+ Only few files to deploy
- Executables are large
- No flexibility
- You cannot deploy plugins

Shared Libraries: Qt has to be pre-installed

Ways of deployment

- Question: now we have compiled all the resources into an executable, can we just distribute this single executable?
- Now really. Distinction between resources and dependents.
- The executable depends on Qt libraries to function.

Static Linking: Results in stand-alone executable

+ Only few files to deploy
- Executables are large
- No flexibility
- You cannot deploy plugins

Shared Libraries: Qt has to be pre-installed

+ Can deploy plugins
+ Qt libs shared between applications
+ Smaller, more flexible executables
- More files to deploy

What is a dialog

Dialog is a special window that requires immediate attention from the user.

What is a dialog

Dialog is a special window that requires immediate attention from the user.
General Dialogs can have 2 modes.

What is a dialog

Dialog is a special window that requires immediate attention from the user.
General Dialogs can have 2 modes.

- Modal dialog
- Modeless dialog

What is a dialog

Dialog is a special window that requires immediate attention from the user.
General Dialogs can have 2 modes.

- Modal dialog
  - Remains in foreground, until closed
  - Blocks input to remaining application
  - Example: Configuration dialog
- Modeless dialog

What is a dialog

Dialog is a special window that requires immediate attention from the user.
General Dialogs can have 2 modes.

- Modal dialog
  - Remains in foreground, until closed
  - Blocks input to remaining application
  - Example: Configuration dialog
- Modeless dialog
  - Operates independently in application
  - Example: Find/Search dialog

What is a dialog

Dialog is a special window that requires immediate attention from the user.

General Dialogs can have 2 modes.

- Modal dialog
  - Remains in foreground, until closed
  - Blocks input to remaining application
  - Example: Configuration dialog
- Modeless dialog
  - Operates independently in application
  - Example: Find/Search dialog

Qt provides a base class `QDialog`

Modal dialog

- There are two ways to set a dialog modal

Modal dialog

- There are two ways to set a dialog modal
  - Use `exec()`

```
MyDialog dialog(this);
dialog.setMyInput(text);
if(dialog.exec() == Dialog::Accepted) {
    // exec blocks until user closes dialog
}
```

Modal dialog

- There are two ways to set a dialog modal
  - Use `exec()`

```
MyDialog dialog(this);
dialog.setMyInput(text);
if(dialog.exec() == Dialog::Accepted) {
    // exec blocks until user closes dialog
}

```

  - Use `show()`, and set the `modal` property of the dialog

Modeless dialog

- Only use `show()`
  - Displays dialog
  - Returns control to caller

Modeless dialog

- Only use `show()`
  - Displays dialog
  - Returns control to caller

```
void EditorWindow::find() {
    if (!m_findDialog) {
        m_findDialog = new FindDialog(this);
        connect(m_findDialog, SIGNAL(findNext()),
            this, SLOT(onFindNext()));
    }
    m_findDialog->show(); // returns immediately
    m_findDialog->raise(); // on top of other windows
    m_findDialog->activateWindow(); // keyboard focus
}

```

Asking for Files – QFileDialog

Allow users to select files or directories

Asking for Files – QFileDialog

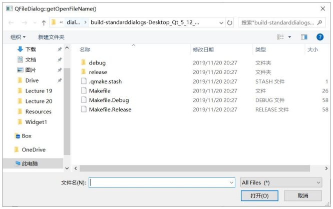
Allow users to select files or directories – include `<QFileDialog>`

Asking for Files – QFileDialog

Allow users to select files or directories – include `<QFileDialog>`

- Asking for a file name

```
QString fileName =
    QFileDialog::getOpenFileName(this, tr("Open
     File"));
if(!fileName.isNull()) {
    // do something useful
}
```



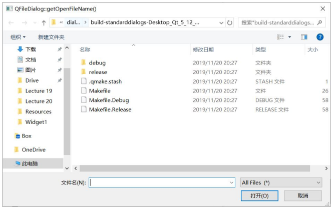
名称	修改日期	类型	大小
debug	2019/11/20 20:27	文件夹	
release	2019/11/20 20:27	文件夹	
.qmake.stash	2019/11/20 20:27	STASH 文件	1
Makefile	2019/11/20 20:27	文件	26
Makefile.Debug	2019/11/20 20:27	DEBUG 文件	58
Makefile.Release	2019/11/20 20:27	RELEASE 文件	58

Asking for Files – QFileDialog

Allow users to select files or directories – include `<QFileDialog>`

- Asking for a file name

```
QString fileName =
    QFileDialog::getOpenFileName(this, tr("Open
     File"));
if(!fileName.isNull()) {
    // do something useful
}
```



名称	修改日期	类型	大小
debug	2019/11/20 20:27	文件夹	
release	2019/11/20 20:27	文件夹	
.qmake.stash	2019/11/20 20:27	STASH 文件	1
Makefile	2019/11/20 20:27	文件	26
Makefile.Debug	2019/11/20 20:27	DEBUG 文件	58
Makefile.Release	2019/11/20 20:27	RELEASE 文件	58

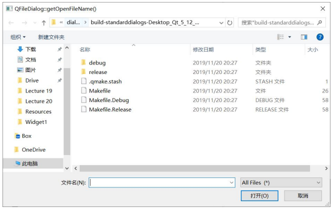
- `QFileDialog::getOpenFileNames()`

Asking for Files – QFileDialog

Allow users to select files or directories – include `<QFileDialog>`

- Asking for a file name

```
QString fileName =
    QFileDialog::getOpenFileName(this, tr("Open
     File"));
if(!fileName.isNull()) {
    // do something useful
}

```



名称	修改日期	类型	大小
debug	2019/11/20 20:27	文件夹	
release	2019/11/20 20:27	文件夹	
.qmake.stash	2019/11/20 20:27	STASH 文件	1
Makefile	2019/11/20 20:27	文件	26
Makefile.Debug	2019/11/20 20:27	DEBUG 文件	58
Makefile.Release	2019/11/20 20:27	RELEASE 文件	58

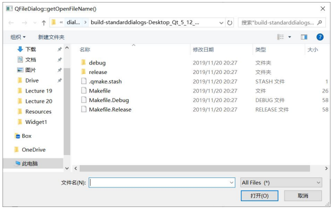
- `QFileDialog::getOpenFileNames()` Returns one or more selected existing files

Asking for Files – QFileDialog

Allow users to select files or directories – include `<QFileDialog>`

- Asking for a file name

```
QString fileName =
    QFileDialog::getOpenFileName(this, tr("Open File"));
if(!fileName.isNull()) {
    // do something useful
}
```



QFileDialog::getOpenFileName()

dial... › build-standarddialogs-Desktop_Qt_5_12_...

名称	修改日期	类型	大小
debug	2019/11/20 20:27	文件夹	
release	2019/11/20 20:27	文件夹	
.qmake.stash	2019/11/20 20:27	STASH 文件	1
Makefile	2019/11/20 20:27	文件	26
Makefile.Debug	2019/11/20 20:27	DEBUG 文件	58
Makefile.Release	2019/11/20 20:27	RELEASE 文件	58

文件名(N):
All Files (*)

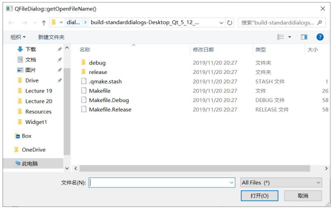
- `QFileDialog::getOpenFileNames()` Returns one or more selected existing files
- `QFileDialog::getSaveFileName()`

Asking for Files – QFileDialog

Allow users to select files or directories – include `<QFileDialog>`

- Asking for a file name

```
QString fileName =
    QFileDialog::getOpenFileName(this, tr("Open
     File"));
if(!fileName.isNull()) {
    // do something useful
}
```



QFileDialog::getOpenFileName()

dial... › build-standarddialogs-Desktop_Qt_5_12_...

搜索"build-standarddialogs...

名称	修改日期	类型	大小
debug	2019/11/20 20:27	文件夹	
release	2019/11/20 20:27	文件夹	
.qmake.stash	2019/11/20 20:27	STASH 文件	1
Makefile	2019/11/20 20:27	文件	26
Makefile.Debug	2019/11/20 20:27	DEBUG 文件	58
Makefile.Release	2019/11/20 20:27	RELEASE 文件	58

文件名(N):

All Files (*)

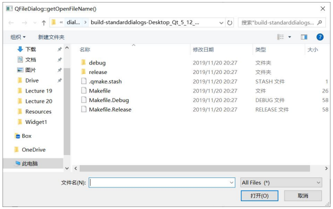
- `QFileDialog::getOpenFileNames()` Returns one or more selected existing files
- `QFileDialog::getSaveFileName()` Returns a file name. File does not have to exist.

Asking for Files – QFileDialog

Allow users to select files or directories – include `<QFileDialog>`

- Asking for a file name

```cpp
QString fileName =
    QFileDialog::getOpenFileName(this, tr("Open
     File"));
if(!fileName.isNull()) {
    // do something useful
}
```


QFileDialog::getOpenFileName()

dial... › build-standarddialogs-Desktop_Qt_5_12_...

搜索"build-standarddialogs...

组织 ▾ 新建文件夹

下载
文档
图片
Drive
Lecture 19
Lecture 20
Resources
Widget1
Box
OneDrive
此电脑

名称	修改日期	类型	大小
debug	2019/11/20 20:27	文件夹	
release	2019/11/20 20:27	文件夹	
.qmake.stash	2019/11/20 20:27	STASH 文件	1
Makefile	2019/11/20 20:27	文件	26
Makefile.Debug	2019/11/20 20:27	DEBUG 文件	58
Makefile.Release	2019/11/20 20:27	RELEASE 文件	58

文件名(N):

All Files (*)

打开(O) 取消

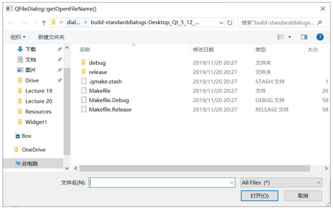
- `QFileDialog::getOpenFileNames()` Returns one or more selected existing files
- `QFileDialog::getSaveFileName()` Returns a file name. File does not have to exist.
- `QFileDialog::getExistingDirectory()`

Asking for Files – QFileDialog

Allow users to select files or directories – include `<QFileDialog>`

- Asking for a file name

```cpp
QString fileName =
    QFileDialog::getOpenFileName(this, tr("Open
     File"));
if(!fileName.isNull()) {
    // do something useful
}
```



名称	修改日期	类型	大小
debug	2019/11/20 20:27	文件夹	
release	2019/11/20 20:27	文件夹	
.qmake.stash	2019/11/20 20:27	STASH 文件	1
Makefile	2019/11/20 20:27	文件	26
Makefile.Debug	2019/11/20 20:27	DEBUG 文件	58
Makefile.Release	2019/11/20 20:27	RELEASE 文件	58

- `QFileDialog::getOpenFileNames()` Returns one or more selected existing files
- `QFileDialog::getSaveFileName()` Returns a file name. File does not have to exist.
- `QFileDialog::getExistingDirectory()` Returns an existing directory.

Showing Messages – QMessageBox

Provides a modal dialog for

- informing the user

Showing Messages – QMessageBox

Provides a modal dialog for

- informing the user
- asking a question and receiving an answer

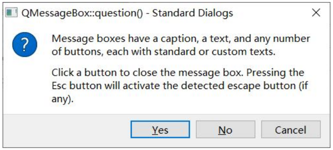
Showing Messages – QMessageBox

Provides a modal dialog for

- informing the user
- asking a question and receiving an answer

- Include <QMessageBox>

```
QMessageBox::StandardButton ret =
    QMessageBox::question(parent, title,
     text);
if(ret == QMessageBox::Ok) {
    // do something useful
}
```


QMessageBox::question() - Standard Dialogs

Message boxes have a caption, a text, and any number of buttons, each with standard or custom texts.

Click a button to close the message box. Pressing the Esc button will activate the detected escape button (if any).

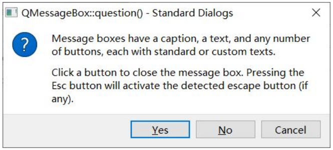
Showing Messages – QMessageBox

Provides a modal dialog for

- informing the user
- asking a question and receiving an answer

- Include `<QMessageBox>`

```
QMessageBox::StandardButton ret =
    QMessageBox::question(parent, title,
     text);
if(ret == QMessageBox::Ok) {
    // do something useful
}
```


QMessageBox::question() - Standard Dialogs

Message boxes have a caption, a text, and any number of buttons, each with standard or custom texts.

Click a button to close the message box. Pressing the Esc button will activate the detected escape button (if any).

- Other convenience methods

```
QMessageBox::information(...)
QMessageBox::warning(...)
QMessageBox::critical(...)
QMessageBox::about(...)
```

Feedback on progress – QProgressDialog

Provides feedback on the progress of a slow operation.

Feedback on progress – QProgressDialog

Provides feedback on the progress of a slow operation. This operation by default is modal

Feedback on progress – QProgressDialog

Provides feedback on the progress of a slow operation. This operation by default is modal

Syntax:

```
QProgressDialog(const QString &labelText, const QString &cancelButtonText, int
      minimum, int maximum, QWidget *parent = nullptr, Qt::WindowFlags f = Qt::
      WindowFlags())

```

Feedback on progress – QProgressDialog

Provides feedback on the progress of a slow operation. This operation by default is modal

Syntax:

```
QProgressDialog(const QString &labelText, const QString &cancelButtonText, int
      minimum, int maximum, QWidget *parent = nullptr, Qt::WindowFlags f = Qt::
      WindowFlags())

```

Example:

```
QProgressDialog dialog("Copy", "Abort", 0, count, this);
dialog.setWindowModality(Qt::WindowModal);
for (int i = 0; i < count; i++) {
    dialog.setValue(i);
    if (dialog.wasCanceled()) { break; }
    //... copy one file
}
dialog.setValue(count); // ensure set to maximum

```

Feedback on progress – QProgressDialog

Provides feedback on the progress of a slow operation. This operation by default is modal
Syntax:

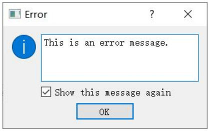
```
QProgressDialog(const QString &labelText, const QString &cancelButtonText, int
     minimum, int maximum, QWidget *parent = nullptr, Qt::WindowFlags f = Qt::
     WindowFlags())
```

Example:

```
QProgressDialog dialog("Copy", "Abort", 0, count, this);
dialog.setWindowModality(Qt::WindowModal);
for (int i = 0; i < count; i++) {
    dialog.setValue(i);
    if (dialog.wasCanceled()) { break; }
    //... copy one file
}
dialog.setValue(count); // ensure set to maximum
```

Modal vs modeless

Providing error messages - QErrorMessage

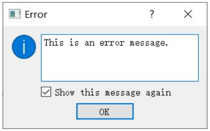


This is an error message.

☑ Show this message again

- Similar to QMessageBox

Providing error messages - QErrorMessage

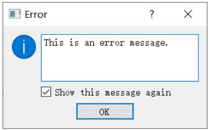


This is an error message.

☑ Show this message again

- Similar to QMessageBox but with a checkbox

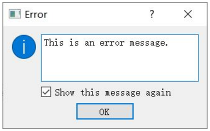
Providing error messages - QErrorMessage



This is an error message.

- Similar to QMessageBox but with a checkbox and is modeless

Providing error messages - QErrorMessage

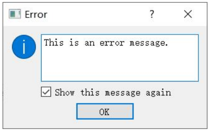


This is an error message.

☑ Show this message again

- Similar to QMessageBox but with a checkbox and is modeless
- Asks if message shall be displayed again

Providing error messages - QErrorMessage

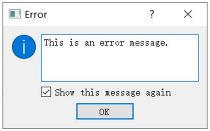


This is an error message.

- Similar to QMessageBox but with a checkbox and is modeless
- Asks if message shall be displayed again

```
m_error = new QErrorMessage(this);
m_error->showMessage(message, type);

```

Providing error messages - QErrorMessage


Error
This is an error message.

- Similar to QMessageBox but with a checkbox and is modeless
- Asks if message shall be displayed again

```
m_error = new QErrorMessage(this);
m_error->showMessage(message, type);

```

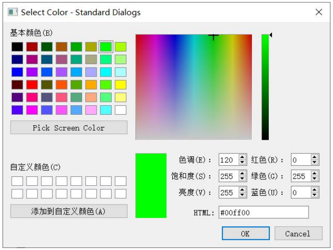
- Messages will be queued
  - Very useful, if there is no dedicated console to save the error messages

Other common dialogs

- Asking for Input - `QInputDialog`
  - `QInputDialog::getText(...)`
  - `QInputDialog::getInt(...)`
  - `QInputDialog::getDouble(...)`
  - `QInputDialog::getItem(...)`

Other common dialogs

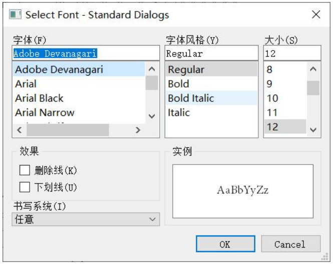
- Asking for Input - `QInputDialog`
  - `QInputDialog::getText(...)`
  - `QInputDialog::getInt(...)`
  - `QInputDialog::getDouble(...)`
  - `QInputDialog::getItem(...)`
- Selecting Color - `QColorDialog`
  - `QColorDialog::getColor(...)`



色调(H): 120 红色(R): 0
饱和度(S): 255 绿色(G): 255
亮度(V): 255 蓝色(U): 0
HTML: #00ff00

Other common dialogs

- Selecting Font - `QFontDialog`
  - `QFontDialog::getFont(...)`



Custom dialogs

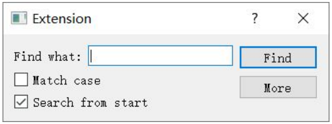
- Inherit from `QDialog`
- Create and layout widgets
- Use `QDialogButtonBox` for dialog buttons
    - Connect buttons to accept()/reject()

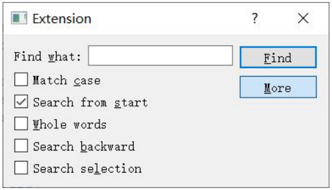
Custom dialogs

- Inherit from `QDialog`
- Create and layout widgets
- Use `QDialogButtonBox` for dialog buttons
  - Connect buttons to accept()/reject()

```
MyDialog::MyDialog(QWidget *parent) : QDialog(parent) {
    m_label = new QLabel(tr("Input Text"), this);
    m_edit = new QLineEdit(this);
    m_box = new QDialogButtonBox( QDialogButtonBox::Ok|
    QDialogButtonBox::Cancel, this);
    connect(m_box, SIGNAL(accepted()), this, SLOT(accept()));
    connect(m_box, SIGNAL(rejected()), this, SLOT(reject()));
    ... // layout widgets
}
void MyDialog::accept() { // customize close behaviour
    if(isDataValid()) { QDialog::accept() }
}

```

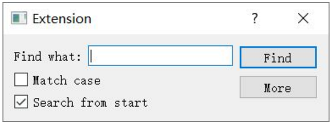
Dialogs with extensions

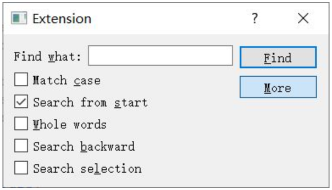




Dialogs with extensions

Use `QWidget::show()/hide()`

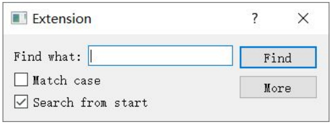


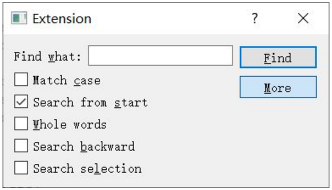


Dialogs with extensions

Use `QWidget::show()/hide()`

```
m_more = new QPushButton(tr("&More"));
m_more->setCheckable(true);

```



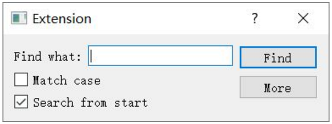


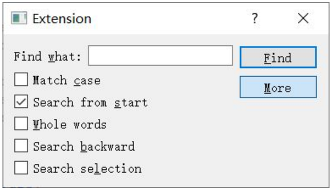
Dialogs with extensions

Use `QWidget::show()/hide()`

```
m_more = new QPushButton(tr("&More"));
m_more->setCheckable(true);

```

```
m_extension = new QWidget(this);
// add your widgets to extension
m_extension->hide();
connect(m_more, SIGNAL(toggled(bool)),
m_extension, SLOT(setVisible(bool)));

```





Qt Designer

In the past couple lectures, we have discussed how to create

- Widgets
- Layouts
- Main window
- Dialog
- Signal and slots

by coding.

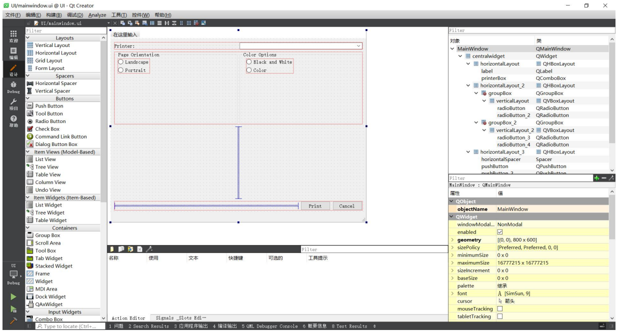
Qt Designer

In the past couple lectures, we have discussed how to create

- Widgets
- Layouts
- Main window
- Dialog
- Signal and slots

by coding.
We could also design the UI forms visually via Qt Designer

Designer views



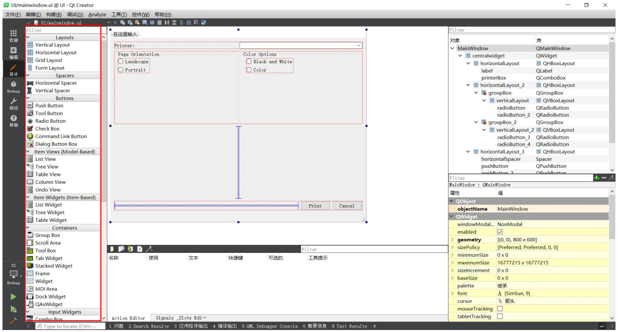
对象	类
MainWindow	QMainWindow
centralwidget	QWidget
horizontalLayout	QHBoxLayout
label	QLabel
printerBox	QComboBox
horizontalLayout_2	QHBoxLayout
groupBox	QGroupBox
verticalLayout	QVBoxLayout
radioButton	QRadioButton
radioButton_2	QRadioButton
groupBox_2	QGroupBox
verticalLayout_2	QVBoxLayout
radioButton_3	QRadioButton
radioButton_4	QRadioButton
horizontalLayout_3	QHBoxLayout
horizontalSpacer	Spacer
pushButton	QPushButton
pushButton_2	QPushButton

MainWindow : QMainWindow

属性	值
QObject	
objectName	MainWindow
QWidget	
windowModal...	NonModal
enabled	☑
geometry	[(0, 0), 800 x 600]
sizePolicy	[Preferred, Preferred, 0, 0]
minimumSize	0 x 0
maximumSize	16777215 x 16777215
sizeIncrement	0 x 0
baseSize	0 x 0
palette	继承
font	[SimSun, 9]
cursor	箭头
mouseTracking	☐
tabletTracking	☐

Designer views

- Widget box



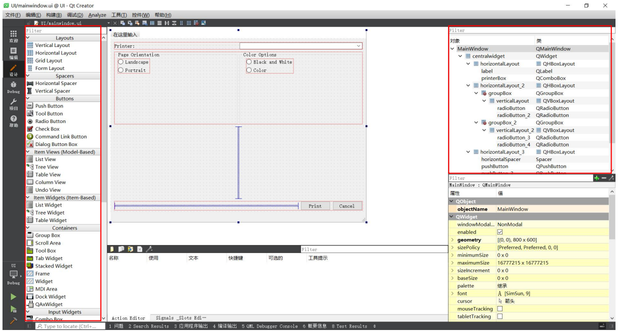
对象	类
MainWindow	QMainWindow
centralwidget	QWidget
horizontalLayout	QHBoxLayout
label	QLabel
printerBox	QComboBox
horizontalLayout_2	QHBoxLayout
groupBox	QGroupBox
verticalLayout	QVBoxLayout
radioButton	QRadioButton
radioButton_2	QRadioButton
groupBox_2	QGroupBox
verticalLayout_2	QVBoxLayout
radioButton_3	QRadioButton
radioButton_4	QRadioButton
horizontalLayout_3	QHBoxLayout
horizontalSpacer	Spacer
pushButton	QPushButton

MainWindow : QMainWindow

属性	值
QObject	
objectName	MainWindow
QWidget	
windowModal...	NonModal
enabled	☑
geometry	[(0, 0), 800 x 600]
sizePolicy	[Preferred, Preferred, 0, 0]
minimumSize	0 x 0
maximumSize	16777215 x 16777215
sizeIncrement	0 x 0
baseSize	0 x 0
palette	继承
font	[SimSun, 9]
cursor	箭头
mouseTracking	☐
tabletTracking	☐

Designer views

- Widget box
- Object inspector



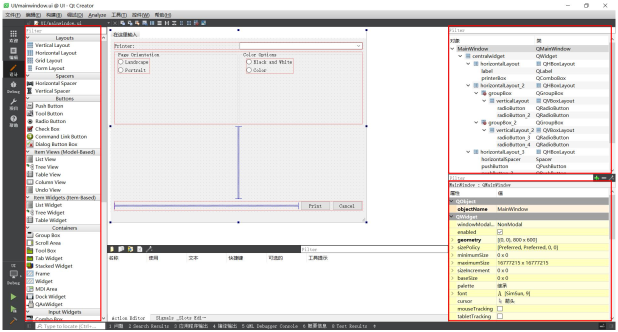
对象	类
MainWindow	QMainWindow
centralwidget	QWidget
horizontalLayout	QHBoxLayout
label	QLabel
printerBox	QComboBox
horizontalLayout_2	QHBoxLayout
groupBox	QGroupBox
verticalLayout	QVBoxLayout
radioButton	QRadioButton
radioButton_2	QRadioButton
groupBox_2	QGroupBox
verticalLayout_2	QVBoxLayout
radioButton_3	QRadioButton
radioButton_4	QRadioButton
horizontalLayout_3	QHBoxLayout
horizontalSpacer	Spacer
pushButton	QPushButton

MainWindow : QMainWindow

属性	值
QObject	
objectName	MainWindow
QWidget	
windowModal...	NonModal
enabled	✓
geometry	[(0, 0), 800 x 600]
sizePolicy	[Preferred, Preferred, 0, 0]
minimumSize	0 x 0
maximumSize	16777215 x 16777215
sizeIncrement	0 x 0
baseSize	0 x 0
palette	继承
font	[SimSun, 9]
cursor	箭头
mouseTracking	☐
tabletTracking	☐

Designer views

- Widget box
- Object inspector
- Property editor



对象	类
MainWindow	QMainWindow
centralwidget	QWidget
horizontalLayout	QHBoxLayout
label	QLabel
printerBox	QComboBox
horizontalLayout_2	QHBoxLayout
groupBox	QGroupBox
verticalLayout	QVBoxLayout
radioButton	QRadioButton
radioButton_2	QRadioButton
groupBox_2	QGroupBox
verticalLayout_2	QVBoxLayout
radioButton_3	QRadioButton
radioButton_4	QRadioButton
horizontalLayout_3	QHBoxLayout
horizontalSpacer	Spacer
pushButton	QPushButton

MainWindow : QMainWindow

属性	值
QObject	
objectName	MainWindow
QWidget	
windowModal...	NonModal
enabled	☑
geometry	[(0, 0), 800 x 600]
sizePolicy	[Preferred, Preferred, 0, 0]
minimumSize	0 x 0
maximumSize	16777215 x 16777215
sizeIncrement	0 x 0
baseSize	0 x 0
palette	继承
font	[SimSun, 9]
cursor	箭头
mouseTracking	☐
tabletTracking	☐

Designer's UI form files

Question: what's the difference between the conventional method and visual designment?

Designer's UI form files

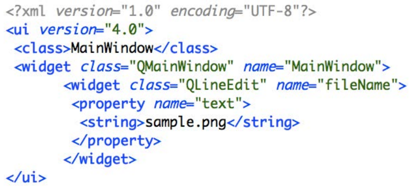
Question: what's the difference between the conventional method and visual designment?

- Form stored in .ui file

Designer's UI form files

Question: what's the difference between the conventional method and visual designment?

- Form stored in .ui file in XML format

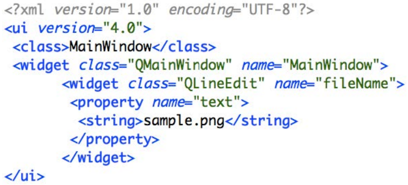


```xml
<?xml version="1.0" encoding="UTF-8"?>
<ui version="4.0">
 <class>MainWindow</class>
 <widget class="QMainWindow" name="MainWindow">
       <widget class="QLineEdit" name="fileName">
        <property name="text">
         <string>sample.png</string>
        </property>
       </widget>
</ui>
```

Designer's UI form files

Question: what's the difference between the conventional method and visual designment?

- Form stored in .ui file in XML format
- `uic` tool is called to generate codes

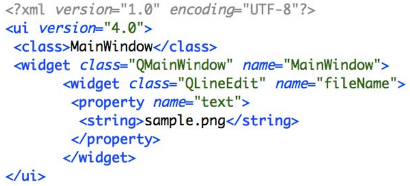


```
<?xml version="1.0" encoding="UTF-8"?>
<ui version="4.0">
 <class>MainWindow</class>
 <widget class="QMainWindow" name="MainWindow">
       <widget class="QLineEdit" name="fileName">
        <property name="text">
         <string>sample.png</string>
        </property>
       </widget>
</ui>
```

Designer's UI form files

Question: what's the difference between the conventional method and visual designment?

- Form stored in .ui file in XML format
- `uic` tool is called to generate codes from myform.ui to ui_myform.h

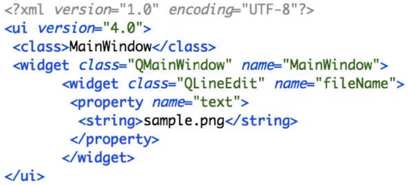


```
<?xml version="1.0" encoding="UTF-8"?>
<ui version="4.0">
 <class>MainWindow</class>
 <widget class="QMainWindow" name="MainWindow">
       <widget class="QLineEdit" name="fileName">
        <property name="text">
         <string>sample.png</string>
        </property>
       </widget>
</ui>
```

Designer's UI form files

Question: what's the difference between the conventional method and visual designment?

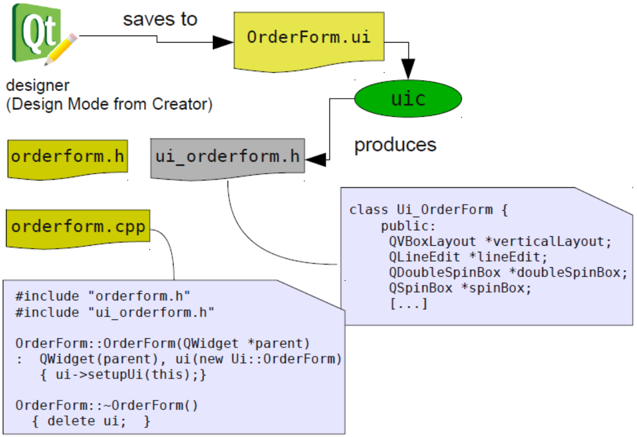
- Form stored in .ui file in XML format
- uic tool is called to generate codes from myform.ui to ui_myform.h



```xml
<?xml version="1.0" encoding="UTF-8"?>
<ui version="4.0">
 <class>MainWindow</class>
 <widget class="QMainWindow" name="MainWindow">
      <widget class="QLineEdit" name="fileName">
       <property name="text">
        <string>sample.png</string>
       </property>
      </widget>
</ui>
```

```cpp
// ui_mainwindow.h
class Ui_MainWindow {
public:
    QLineEdit *fileName;
    ... // simplified code
    void setupUi(QWidget *) { /* setup widgets */ }
};

```

From .ui to C++

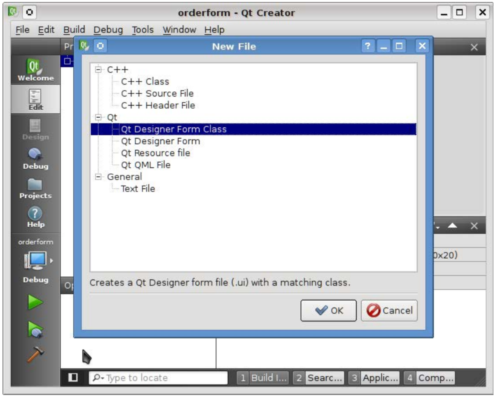


```
class Ui_OrderForm {
    public:
     QVBoxLayout *verticalLayout;
     QLineEdit *lineEdit;
     QDoubleSpinBox *doubleSpinBox;
     QSpinBox *spinBox;
     [...]
```

```
#include "orderform.h"
#include "ui_orderform.h"

OrderForm::OrderForm(QWidget *parent)
:  QWidget(parent), ui(new Ui::OrderForm)
   { ui->setupUi(this);}

OrderForm::~OrderForm()
  { delete ui;  }
```

Qt Creator - Form Wizards

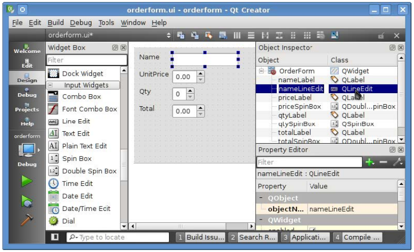
- Add New... "Designer Form"
- or "Designer Form Class" (for C++ integration)



Creates a Qt Designer form file (.ui) with a matching class.

Naming widgets

- Place widgets on form
- Edit `objectName` property



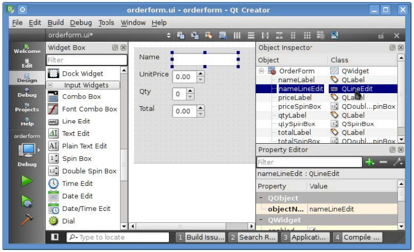
Object	Class
OrderForm	QWidget
nameLabel	QLabel
nameLineEdit	QLineEdit
priceLabel	QLabel
priceSpinBox	QDoubl...pinBox
qtyLabel	QLabel
qtySpinBox	QSpinBox
totalLabel	QLabel
totalSpinBox	QDoubl...pinBox

nameLineEdit : QLineEdit

Property	Value
QObject	
objectN...	nameLineEdit
QWidget	

Naming widgets

- Place widgets on form
- Edit `objectName` property



Object	Class
OrderForm	QWidget
nameLabel	QLabel
nameLineEdit	QLineEdit
priceLabel	QLabel
priceSpinBox	QDoubl...pinBox
qtyLabel	QLabel
qtySpinBox	QSpinBox
totalLabel	QLabel
totalSpinBox	QDoubl...pinBox

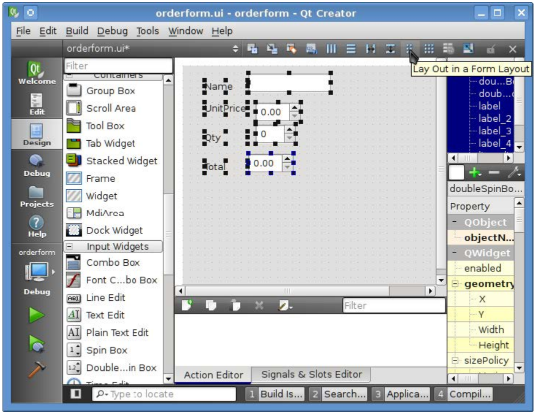
nameLineEdit : QLineEdit

Property	Value
QObject	
objectN...	nameLineEdit
QWidget	

- objectName defines member name in generated code

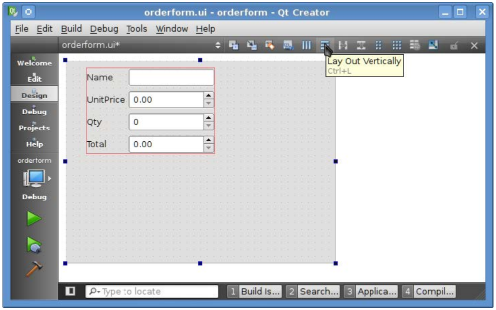
Form layout

QFormLayout: Suitable for most input forms



Top-level layout

- First layout child widgets
- Finally select empty space and set top-level layout



Preview

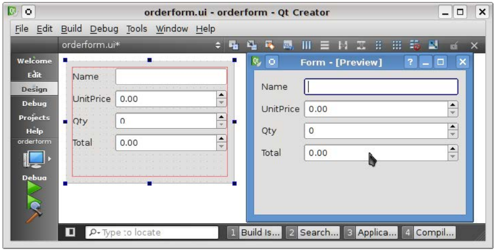
Check whether the widget is nicely resizable

Preview

Check whether the widget is nicely resizable
Tool→Form Editor→Preview

Preview

Check whether the widget is nicely resizable
Tool→Form Editor→Preview



Name
UnitPrice 0.00
Qty 0
Total 0.00

Header

```
// orderform.h
class Ui_OrderForm;
class OrderForm : public QDialog {
private:
    Ui_OrderForm *ui; // pointer to UI object
};

```

Header

```cpp
// orderform.h
class Ui_OrderForm;
class OrderForm : public QDialog {
private:
    Ui_OrderForm *ui; // pointer to UI object
};

```

- “Your Widget” derives from appropriate base class

Header

```
// orderform.h
class Ui_OrderForm;
class OrderForm : public QDialog {
private:
    Ui_OrderForm *ui; // pointer to UI object
};

```

- "Your Widget" derives from appropriate base class
- `*ui` member encapsulate UI class

Header

```cpp
// orderform.h
class Ui_OrderForm;
class OrderForm : public QDialog {
private:
    Ui_OrderForm *ui; // pointer to UI object
};

```

- "Your Widget" derives from appropriate base class
- `*ui` member encapsulate UI class
- Question: what's the benefit of encapsulation?

Header

```
// orderform.h
class Ui_OrderForm;
class OrderForm : public QDialog {
private:
    Ui_OrderForm *ui; // pointer to UI object
};

```

- “Your Widget” derives from appropriate base class
- `*ui` member encapsulate UI class
- Question: what’s the benefit of encapsulation?
- Makes header independent of designer generated code

Implementation file

```
// orderform.cpp
#include "ui_orderform.h"
OrderForm::OrderForm(QWidget *parent)
: QDialog(parent), ui(new Ui_OrderForm) {
    ui->setupUi(this);
}
OrderForm::~OrderForm() {
    delete ui; ui=0;
}

```

Signal and slots in Designer

There are two ways to implement

Signal and slots in Designer

There are two ways to implement

- Traditionally
- Visually

Signal and slots in Designer

There are two ways to implement

- Traditionally
  - Widgets created in UI could be accessed as public members
- Visually

Signal and slots in Designer

There are two ways to implement

- Traditionally
  - Widgets created in UI could be accessed as public members
  - Then setup conventionally
- Visually

Signal and slots in Designer

There are two ways to implement

- Traditionally
  - Widgets created in UI could be accessed as public members
  - Then setup conventionally
  - `connect(ui->okButton, SIGNAL(clicked()), ...`
- Visually

Signal and slots in Designer

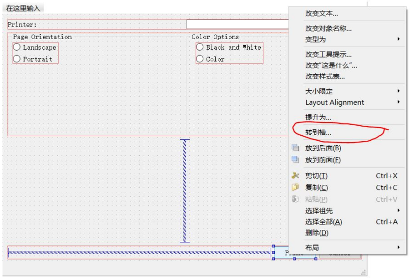
There are two ways to implement

- Traditionally
  - Widgets created in UI could be accessed as public members
  - Then setup conventionally
  - `connect(ui->okButton, SIGNAL(clicked()), ...`
- Visually
  - Qt Creator: right-click on widget and "Go To Slot"

Signal and slots in Designer

There are two ways to implement

- Traditionally
  - Widgets created in UI could be accessed as public members
  - Then setup conventionally
  - `connect(ui->okButton, SIGNAL(clicked()), ...`
- Visually
  - Qt Creator: right-click on widget and "Go To Slot"


在这里输入
Printer:
Page Orientation
Landscape
Portrait
Color Options
Black and White
Color
改变文本...
改变对象名称...
变型为
改变工具提示...
改变"这是什么"...
改变样式表...
大小限定
Layout Alignment
提升为...
转到槽...
放到后面(B)
放到前面(F)
剪切(T) Ctrl+X
复制(C) Ctrl+C
粘贴(P) Ctrl+V
选择祖先
选择全部(A) Ctrl+A
删除(D)
布局