Qt Training – Widget Edition

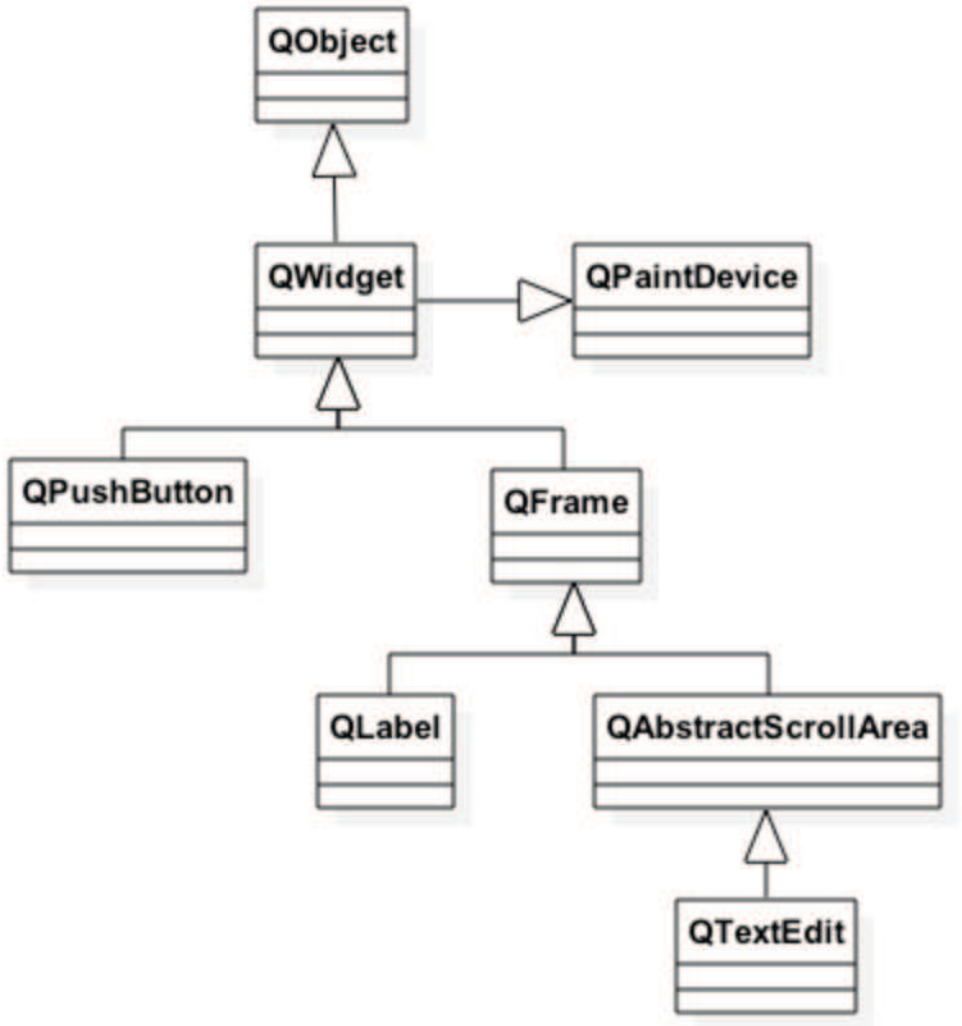
December 2016
Based on Qt 5.8

Objectives

- **Common Widgets**
  - Text widgets
  - Value based widgets
  - Organizer widgets
  - Item-based widgets
- **Layout Management**
  - Geometry management
  - Advantages of layout managers
  - Qt's layout managers
  - Size policies
- **Custom Widgets**
  - Rules for creating own widgets

Qt's Widget Model - QWidget

- **Derived from** `QObject`
  - Adds visual representation
- **Base of user interface objects**
- **Receives events**
  - e.g. mouse, keyboard events
- **Paints itself on screen**
  - Using styles



Object Tree and `QWidget`

- `new QWidget(0)`
  - Widget with no parent = "window" – `QWindow` created in the platform abstraction
  - AKA top-level widget
- `QWidget's` children
  - Positioned in parent's coordinate system
  - Clipped by parent's boundaries
- `QWidget` parent
  - Propagates state changes
  - Hides/shows children when it is hidden/shown itself
  - Enables/disables children when it is enabled/disabled itself

- ***Tristate* mechanism**
  - For hide/show and enable/disable, ensures that e.g. an explicitly hidden child is not shown when the parent is shown.

Text Widgets

- **QLabel**

```
label = new QLabel("Text", parent);
```

  - `setPixmap( pixmap )` - as content

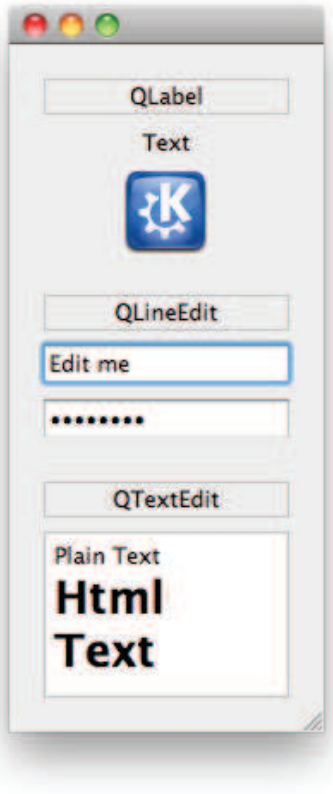
- **QLineEdit**

```
line = new QLineEdit(parent);
line->setText("Edit me");
line->setEchoMode(QLineEdit::Password);
connect(line, SIGNAL(textChanged(QString)) ...
connect(line, SIGNAL(editingFinished()) ...
```

  - `setInputMask( mask )`
  - `setValidator( validator )`

- **QTextEdit**

```
edit = new QTextEdit(parent);
edit->setPlainText("Plain Text");
edit->append("<h1>Html Text</h1>");
connect(edit, SIGNAL(textChanged(QString)) ...
```



Button Widgets

- **QAbstractButton**
  - Abstract base class of buttons
- **QPushButton**

```
button = new QPushButton("Push Me", parent);
button->setIcon(QIcon("images/icon.png"));
connect(button, SIGNAL(clicked()) ...
```

  - setCheckable(bool)-togglebutton
- **QRadioButton**

```
radio = new QRadioButton("Option 1", parent);
```

- **QCheckBox**

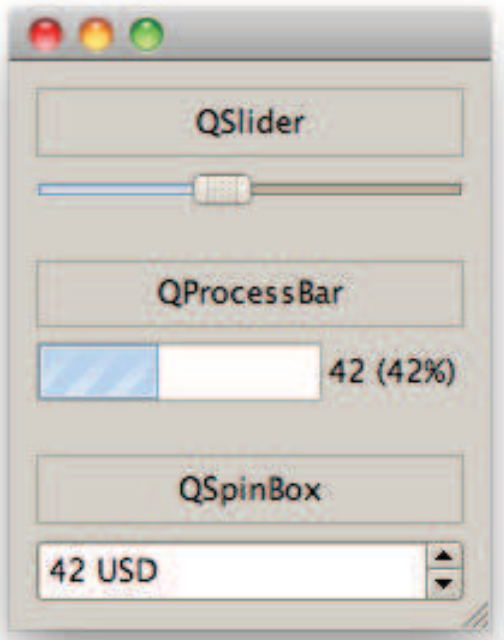
```
check = new QCheckBox("Choice 1", parent);
```

- **QButtonGroup** - non-visual button manager

```
group = new QButtonGroup(parent);
group->addButton(button); // add more buttons
group->setExclusive(true);
connent(group, SIGNAL(buttonClicked(QAbstractButton*))
```

Value Widgets

- **QSlider**
  - `slider = new QSlider(Qt::Horizontal, parent);`
  - `slider->setRange(0, 99);`
  - `slider->setValue(42);`
  - `connect(slider, SIGNAL(valueChanged(int)) ...`
- **QProgressBar**
  - `progress = new QProgressBar(parent);`
  - `progress->setRange(0, 99);`
  - `progress->setValue(42);`
    `// format: %v for value; %p for percentage`
  - `progress->setFormat("%v (%p%)");`
- **QSpinBox**
  - `spin = new QSpinBox(parent);`
  - `spin->setRange(0, 99);`
  - `spin->setValue(42);`
  - `spin->setSuffix(" USD");`
  - `connect(spin, SIGNAL(valueChanged(int))`



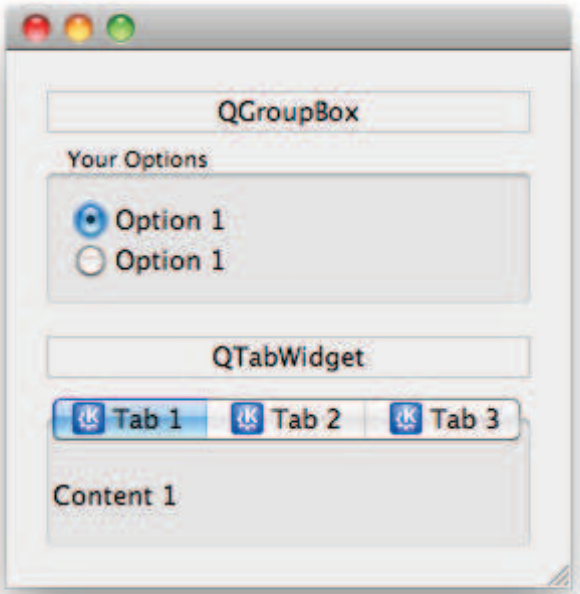
Organizer Widgets

- **QGroupBox**

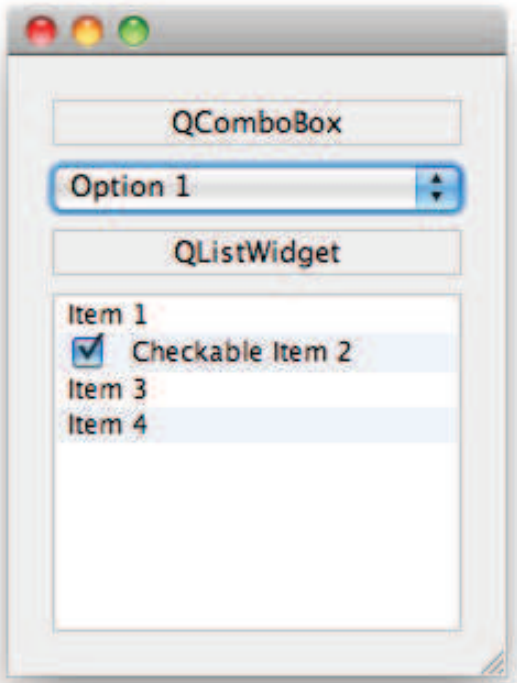
```
box = new QGroupBox("Your Options", parent);
// ... set layout and add widgets
```

  - `setCheckable( bool ) - checkbox in title`

- **QTabWidget**
  - `tab = new QTabWidget(parent);`
  - `tab->addWidget(widget, icon, "Tab 1");`
  - `connect(tab, SIGNAL(currentChanged(int)) ...`
  - `setCurrentWidget( widget )`
    - Displays page associated by widget
  - `setTabPosition( position )`
    - Defines where tabs are drawn
  - `setTabsClosable( bool )`
    - Adds close buttons



Item Widgets



- **QComboBox**

```
combo = new QComboBox(parent);
combo->addItem("Option 1", data);
connect(combo, SIGNAL(activated(int)) ...
QVariant data = combo->itemData(index);
```

  - `setCurrentIndex( index )`

- **QListWidget**

```
list = new QListWidget(parent);
list->addItem("Item 1");
// ownership of items with list
tem = new QListWidgetItem("Item 2", list); item->setCheckState(Qt::Checked);
connect(list, SIGNAL(itemActivated(QListWidgetItem*)) ...
```

  - `QListWidgetItem::setData(Qt::UserRole, data)`

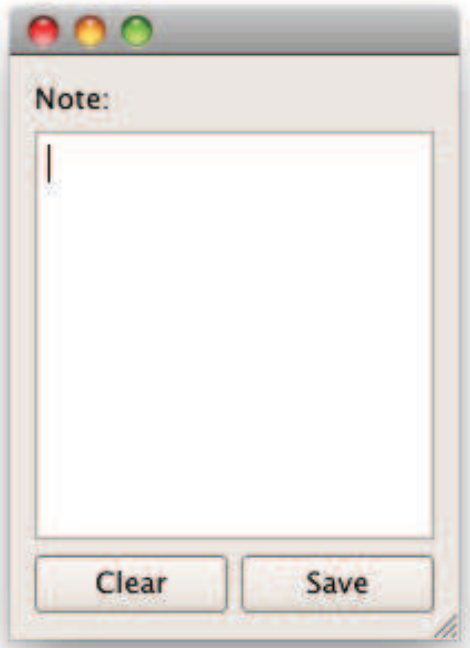
- *Other Item Widgets*: `QTableWidget`, `QTreeWidget`

Other Widgets

- **`QToolBox`**
  - Column of tabbed widget items
- **`QDateEdit`, `QTimeEdit`, `QDateTimeEdit`**
  - Widget for editing date and times
- **`QCalendarWidget`**
  - Monthly calendar widget
- **`QToolButton`**
  - Quick-access button to commands
- **`QSplitter`**
  - Implements a splitter widget
- **`QStackedWidget`**
  - Stack of widgets
  - Only one widget visible at a time

Widgets that Contain Other Widgets

- Container Widget
  - Aggregates other child-widgets
- Use layouts for aggregation
  - In this example: `QHBoxLayout` and `QVBoxLayout`
  - Note: Layouts are *not* widgets
- Layout Process
  - Add widgets to layout
  - Layouts may be nested
  - Set layout on container widget



Layout Management – Doing it Yourself

- Place and resize widgets
  - `move()`
  - `resize()`
  - `setGeometry()`
- Example:

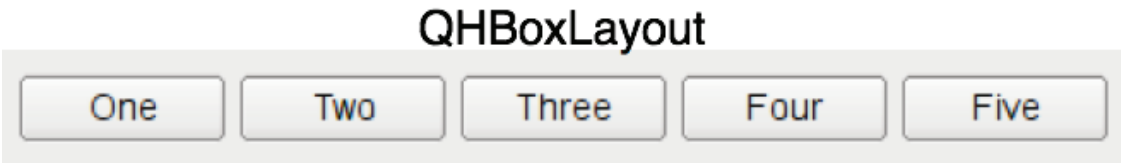
```
QWidget *parent = new QWidget(...);
parent->resize(400,400);

QCheckBox *cb = new QCheckBox(parent);
cb->move(10, 10);
```

Layout Management Classes

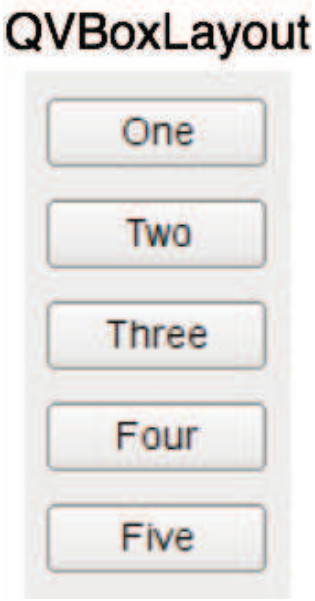
- **`QHBoxLayout`**
  - Lines up widgets horizontally
- **`QVBoxLayout`**
  - Lines up widgets vertically
- **`QGridLayout`**
  - Arranges the widgets in a grid
- **`QFormLayout`**
  - Lines up a (label, widget) pairs in two columns.
- **`QStackedLayout`**
  - Arranges widgets in a stack
    - Only top most is visible

QHBoxLayout and QVBoxLayout

- Lines up widgets horizontally or vertically
- Divides space into boxes
- Each managed widgets fills in one box

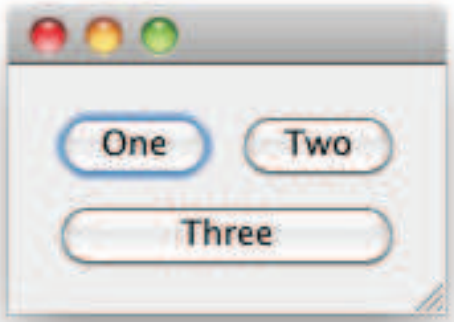



```
QWidget* window = new QWidget;
QPushButton* one = new QPushButton("One");
…
QHBoxLayout* layout = new QHBoxLayout;
layout->addWidget(one);
…
window->setLayout(layout);
```



Widgets in a Grid - QGridLayout

```
QWidget* window= new QWidgetn;
QPushButton* one = new QPushButton("One");
QGridLayout* layout = new QGridLayout;
layout->addWidget(one, 0, 0); // row:0, col:0
layout->addWidget(two, 0, 1); // row:0, col:1
// row:1, col:0, rowSpan:1, colSpan:2
layout->addWidget(three, 1, 0, 1, 2);
window->setLayout(layout)
```



- Additional
  - `setColumnMinimumWidth()` *(minimum width of column)*
  - `setRowMinimumHeight()` *(minimum height of row)*
- *No need to specify rows and columns before adding children*

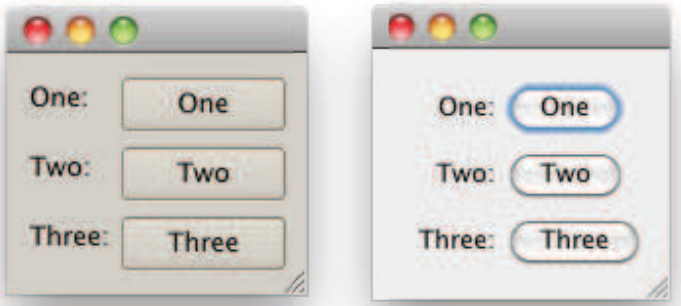
ex-grid-layout

QFormLayout

- A two-column layout
  - Column 1 a label (as annotation)
  - Column 2 a widget (as field)
- Respects style guide of individual platforms.

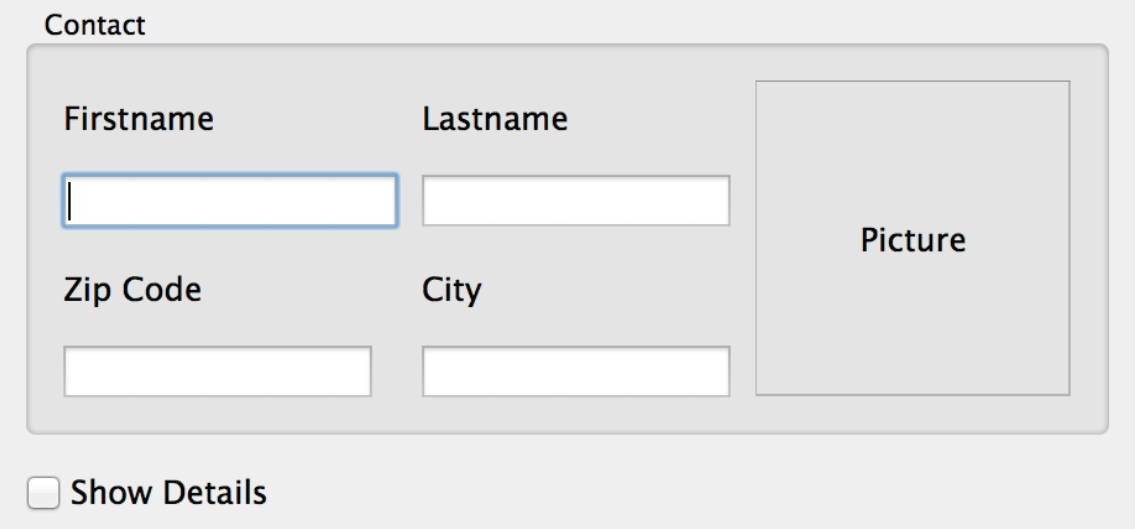
```
QWidget* window = new QWidget();
QPushButton* one = new QPushButton("One"); …
QFormLayout* layout = new QFormLayout();
layout->addRow("One", one);
..
window->setLayout(layout)
```

- Form layout with clean looks and mac style

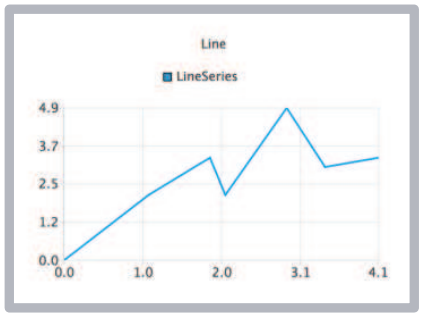


Lab – *Contact Form*

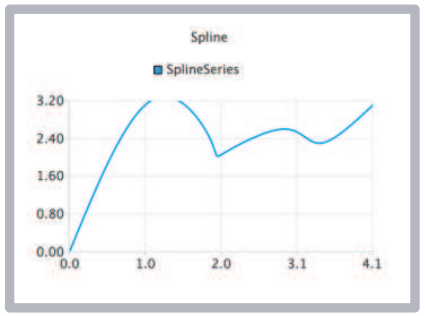
- Specified by graphic designer
- Your task: implement it
  - Focus on correct layout
  - Details disabled by default
  - 'ShowDetails' enables details
- Optional:
  - Click on Picture
  - Lets user choose image
  - See lab description
- Validate Zip-Code as integers



Qt Charts – Chart Types

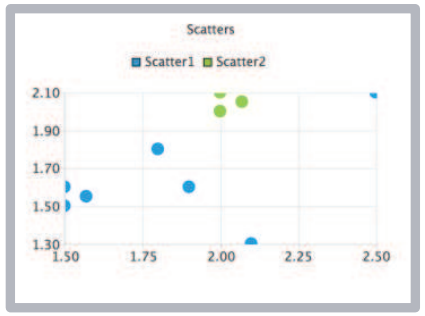


Line chart

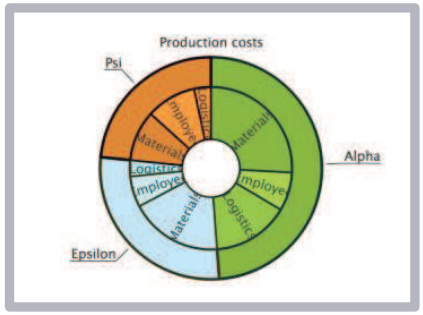


Spline chart

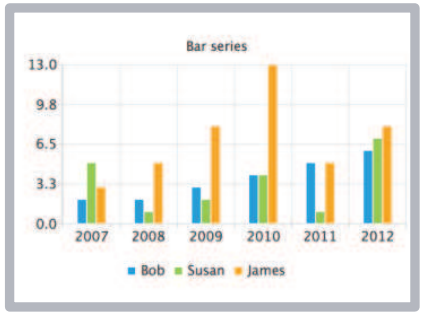
Area chart



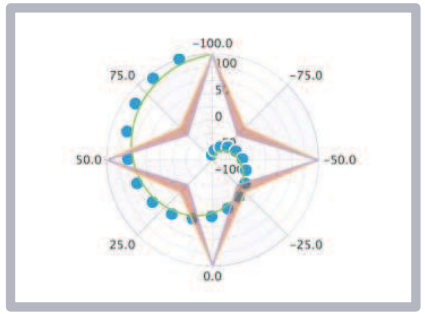
Scatter chart



Pie chart
(pie, donut)



Bar chart
(bar, stacked,
percent)



Polar chart

Candlestick
chart

Charts Architecture

- Charts API is based on the Qt Graphics View framework
- Charts can be displayed as graphics widgets using `QChart`
  - Provides API for zooming and scrolling
- `QChartView` allows using the chart as a widget
- Each chart type is represented by `QAbstractSeries` derived class