Technology Tarena 达内科技

Qt编程

QT PROGRAMMING DAY02

内容

上午	09:00 ~ 09:30	作业讲解和回顾
	09:30 ~ 10:20	从BOP到OOP
	10:30 ~ 11:20	
	11:30 ~ 12:20	信号和槽的其它用法
下午	14:00 ~ 14:50	
	15:00 ~ 15:50	Qt设计师
	16:00 ~ 16:50	
	17:00 ~ 17:30	总结和答疑

从BOP到OOP

Technology Tarena 达内科技

从BOP到OOP

基于对象的Qt编程

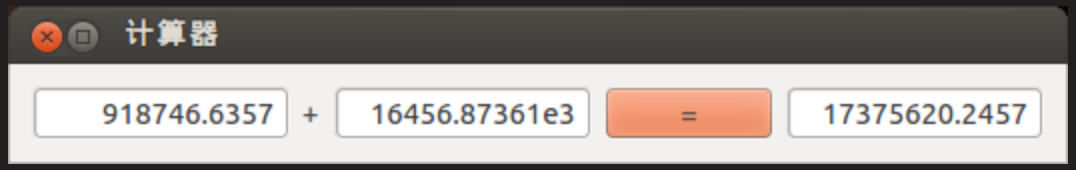
- 完全不使用任何面向对象技术，而只是利用Qt所提供的类创建对象，并操作对象的接口以满足用户的需求是可能的，但这样构建的应用程序其功能必然是十分有限的
  - 首先，不使用面向对象技术通常意味着除了接受Qt所提供的默认外观和缺省行为外，很难扩展出太多定制化的特性
  - 其次，Qt类保护成员中的诸多实现无法在类的外部被复用；Qt试图通过多态实现的很多机制，如事件，完全无法奏效
  - 再次，Qt提供的信号和槽再丰富，也不可能面面俱到满足用户一切可能的需求，自定义信号和槽需要面向对象技术
  - 最后，Qt设计师、Qt创造器，乃至整个Qt工具链都在鼓励以面向对象的方式使用Qt，反其道而行之不会有好结果

知识讲解

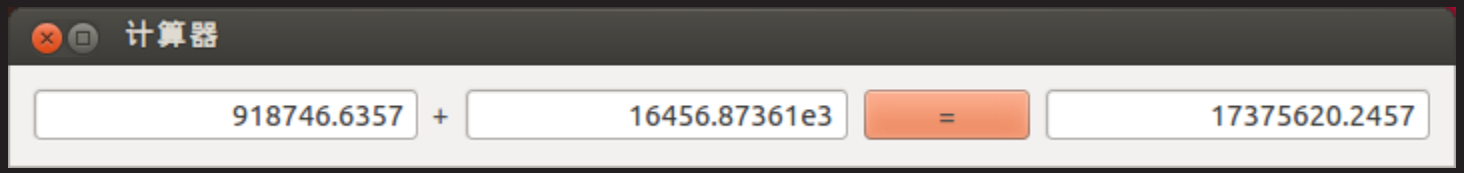
面向对象的Qt编程

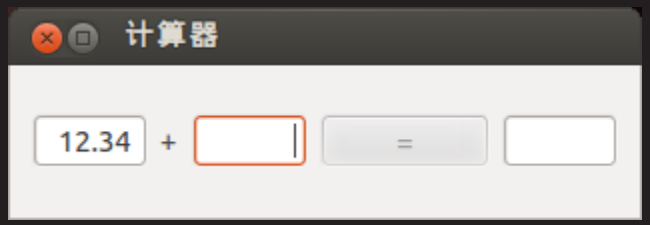
Technology Tarena 达内科技

- 简单(两个实数相加)计算器的面向对象实现



  - 输入两个实数，支持指数形式，按"="按钮显示计算结果
  - 两个操作数必须都是合法的实数，拒绝接受任何非法字符
  - 两个操作数必须全部合法，"="按钮才被激活，否则禁用
  - 显示结果的部件只可查看不可修改，但支持复制到剪贴板
  - 所有子窗口的大小和位置随主窗口的缩放自动调整至最佳





知识讲解

面向对象的Qt编程（续1）

知识讲解

- 创建Calculator目录作为计算器的工程目录
  - `$ mkdir Calculator`
- 在工程目录下编辑CalculatorDlg.h文件
  - `$ cd Calculator`
    `$ vi CalculatorDlg.h`
- 加入头文件卫士防止钻石包含
  - ```
    #ifndef _CALCULATORDLG_H
    #define _CALCULATORDLG_H
    ...
    #endif // _CALCULATORDLG_H
    ```
- 包含QDialog类的头文件
  - `#include <QDialog>`


    ```

面向对象的Qt编程（续2）

- 加入两个前置声明
  - `class QLineEdit;`
    `class QPushButton;`
- 从QDialog基类中派生出自己的子类CalculatorDlg
  - `class CalculatorDlg : public QDialog { ... };`
- 将需要在不同成员函数中访问的子窗口定义为成员变量
  - `QLineEdit*  m_editX;  // 左操作数编辑框`
    `QLineEdit*  m_editY;  // 右操作数编辑框`
    `QPushButton* m_btnCalc; // "="按钮`
    `QLineEdit*  m_editZ;  // 计算结果编辑框`
- 声明构造函数，参数为父窗口指针，NULL表示顶层窗口
  - `CalculatorDlg (QWidget* parent = NULL);`

知识讲解

面向对象的Qt编程（续3）

- 在工程目录下编辑CalculatorDlg.cpp文件
  - `vi CalculatorDlg.cpp`
- 包含QtCore和QtGui的总头文件及类声明头文件
  - `#include <QtGui>`
    `#include "CalculatorDlg.h"`
- 定义CalculatorDlg类的构造函数，初始化对话框
  - `CalculatorDlg::CalculatorDlg (QWidget* parent) :`
    `QDialog (parent) { ... }`
- CalculatorDlg类的构造函数首先设置窗口标题
  - `setWindowTitle (tr ("计算器"));`

知识讲解

面向对象的Qt编程（续4）

- 其次创建左操作数编辑框
  -
```
m_editX = new QLineEdit;
m_editX->setAlignment (Qt::AlignRight);
m_editX->setValidator (new QDoubleValidator (this));
```
  - 其中QDoubleValidator为双精度浮点数验证器，确保使用它的编辑框只能接受合法的数字字符、表示指数的e或者E，以及正负号和小数点
- 创建右操作数和结果编辑框，后者无需验证器且为只读
  -
```
m_editY = new QLineEdit;
m_editY->setAlignment (Qt::AlignRight);
m_editY->setValidator (new QDoubleValidator (this));
```
  -
```
m_editZ = new QLineEdit;
m_editZ->setAlignment (Qt::AlignRight);
m_editZ->setReadOnly (true);
```

知识讲解

面向对象的Qt编程（续5）

- 创建"="按钮，初始禁用
  - `m_btnCalc = new QPushButton ("=");`
    `m_btnCalc->setEnabled (false);`
- 创建水平布局器，将三个编辑框和一个按钮按从左到右的顺序依次放入布局器中，并将该布局器作为对话框的布局
  - `QHBoxLayout* layHor = new QHBoxLayout;`
    `layHor->addWidget (m_editX);`
    `layHor->addWidget (new QLabel ("+"));`
    `layHor->addWidget (m_editY);`
    `layHor->addWidget (m_btnCalc);`
    `layHor->addWidget (m_editZ);`
    `setLayout (layHor);`

知识讲解

面向对象的Qt编程（续6）

知识讲解

- 在CalculatorDlg.h的类声明中增加两个私有槽函数，分别用于在适当的时机激活"="按钮和执行计算
  -
```
private slots:
    void enableCalcButton (void);
    void calcClicked (void);
```
- 这里所用的"private slots:"并非标准C++语法，因此需要借助Qt的元对象编译器(moc)编译为标准语法的形式。为此应该在类声明的最开始处加上一个名为Q_OBJECT的宏
  -
```
class CalculatorDlg : public QDialog {
    Q_OBJECT
    ...
};
```

面向对象的Qt编程（续7）

- 然后在CalculatorDlg.cpp中给出这两个槽函数的定义

```
– void CalculatorDlg::enableCalcButton (void) {
    bool bXOk, bYOk;
    m_editX->text ().toDouble (&bXOk);
    m_editY->text ().toDouble (&bYOk);
    m_btnCalc->setEnabled (bXOk && bYOk);
    if (! m_btnCalc->isEnabled ())
      m_editZ->setText ("");
  }
– void CalculatorDlg::calcClicked (void) {
    m_editZ->setText (QString::number (
      m_editX->text ().toDouble () +
      m_editY->text ().toDouble (), 'g', 15));
  }
```

知识讲解

面向对象的Qt编程（续8）

- 在CalculatorDlg类的构造函数中建立信号和槽的连接
  -

```
connect (m_editX,
  SIGNAL (textChanged (const QString&)),
  this, SLOT (enableCalcButton (void)));
connect (m_editY,
  SIGNAL (textChanged (const QString&)),
  this, SLOT (enableCalcButton (void)));
connect (m_btnCalc,
  SIGNAL (clicked (void)),
  this, SLOT (calcClicked (void)));
```

知识讲解

面向对象的Qt编程（续9）

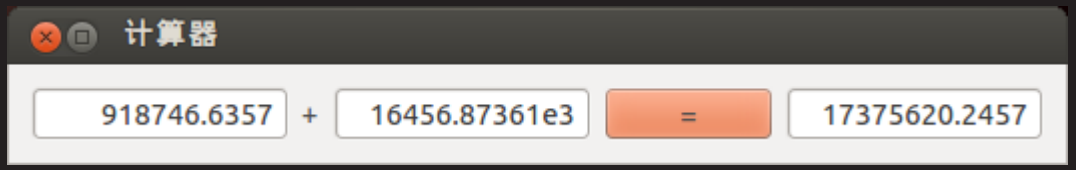
- 在Calculator.cpp中定义main函数，创建并显示窗口
  -
```
#include <QTextCodec>
#include <QApplication>
#include "CalculatorDlg.h"
int main (int argc, char* argv[]) {
  QTextCodec::setCodecForTr (
    QTextCodec::codecForName ("utf-8"));
  QApplication app (argc, argv);
  CalculatorDlg dlg;
  dlg.show ();
  return app.exec ();
}
```
- 最后用qmake生成Makefile，用make编译链接，运行测试

面向对象的计算器

【参见：TTS COOKBOOK 】

课堂练习

- 面向对象的计算器



信号和槽的其它用法


信号和槽的其它用法
信号和槽的其它用法
一个信号对多个槽
多个信号对一个槽
信号级联
参数一致与中间槽
在更大范围内使用信号和槽

Technology Tarena 达内科技

信号和槽的其它用法

一个信号对多个槽

- 一个信号可以被连接到多个槽
  - connect (a, SIGNAL (f (void)), a, SLOT (x (void)));
    connect (a, SIGNAL (f (void)), b, SLOT (y (void)));
    connect (a, SIGNAL (f (void)), c, SLOT (z (void)));
  - 当f信号被从a对象发出时，会以不确定的顺序一个接一个地调用a对象的x槽、b对象的y槽和c对象的z槽
- 例如
  - connect (slider, SIGNAL (valueChanged (int)),
    spin, SLOT (setValue (int)));
  - connect (slider, SIGNAL (valueChanged (int)),
    this, SLOT (updateStatusBarIndicator (int)));

知识讲解

多个信号对一个槽

- 多个信号可以被连接到一个槽
  - connect (a, SIGNAL (f (void)), a, SLOT (x (void)));
    connect (b, SIGNAL (g (void)), a, SLOT (x (void)));
    connect (c, SIGNAL (h (void)), a, SLOT (x (void)));
  - 无论所发出的是a对象的f信号、b对象的g信号，还是c对象的h信号，最终被调用的都是a对象的x槽
- 例如
  - connect (lcd, SIGNAL (overflow (void)),
    this, SLOT (handleMathError (void)));
  - connect (calculator, SIGNAL (divByZero (void)),
    this, SLOT (handleMathError (void)));

知识讲解

多个信号对一个槽（续1）

- 当多个信号被连接到同一个槽时，槽函数可能需要针对不同的信号发送者做出不同的响应
- 在槽函数中可以调用QObject类的sender()成员函数获取信号发送者对象
  - QObject* QObject::sender (void) const;
- 任何QObject及其子类的对象都具有objectName属性，可以为每个对象命名，并通过名称加以区分
  - QString objectName (void) const
  - void setObjectName (const QString& name);

知识讲解

信号级联

- 一个信号可以与另外一个信号相连接
  - connect (a, SIGNAL (f (void)), a, SIGNAL (g (void)));
    connect (a, SIGNAL (g (void)), b, SIGNAL (g (void)));
    connect (b, SIGNAL (g (void)), c, SIGNAL (h (void)));
  - 当a对象发射f信号时，该对象的g信号也会发出，并导致b对象g信号和c对象h信号相继发出
- 例如
  - connect (lineEdit,
    SIGNAL (textChanged (QString const&)),
    this, SIGNAL (updateRecord (QString const&)));

知识讲解

参数一致与中间槽

- 要把信号成功地连接到槽或另外一个信号，它们的参数必须具有相同的顺序和类型，但可以带有缺省值
  - `signal (QString, int) -> slot (QString, int)` Ok
    `signal (QString, int) -> slot (int, QString)` No
    `signal (QString) -> slot (QString, int)` No
    `signal (QString) -> slot (QString, int = 0)` Ok
- 如果信号参数多于相连的槽参数，多余的参数将被忽略
  - connect (ftp,
    SIGNAL (rawCommandReply (int, QString)),
    this, SLOT (checkErrorCode (int)));
- 如果信号和槽的参数类型不匹配、信号或槽不存在、信号和槽的签名中包含了参数名，Qt会在运行时发出警告

知识讲解

参数一致与中间槽（续1）

- 如果信号和槽的参数表不一致，可以借助中间槽转发
  - connect (btnTime, SIGNAL (clicked (void)),
    this, SLOT (timeClicked (void)));
  - 在中间槽内部可以直接调用目标槽
    - ✓ void TimeDlg::timeClicked (void) {
      m_labTime->setText (
      QTime::currentTime ().toString ("h:mm:ss")); }
  - 也可以发送自定义信号，并令自定义信号连接目标槽
    - ✓ connect (this, SIGNAL (setText (QString const&)),
      m_labTime, SLOT (setText (QString const&)));
    - ✓ void TimeDlg::timeClicked (void) {
      emit setText (
      QTime::currentTime ().toString ("h:mm:ss")); }
  - 自定义信号要事先在头文件中声明
    - ✓ signals:
      void setText (QString const&);

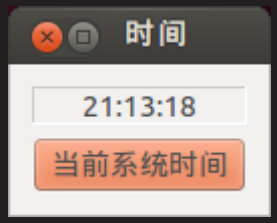
知识讲解

显示系统时间

【参见：TTS COOKBOOK 】

课堂练习

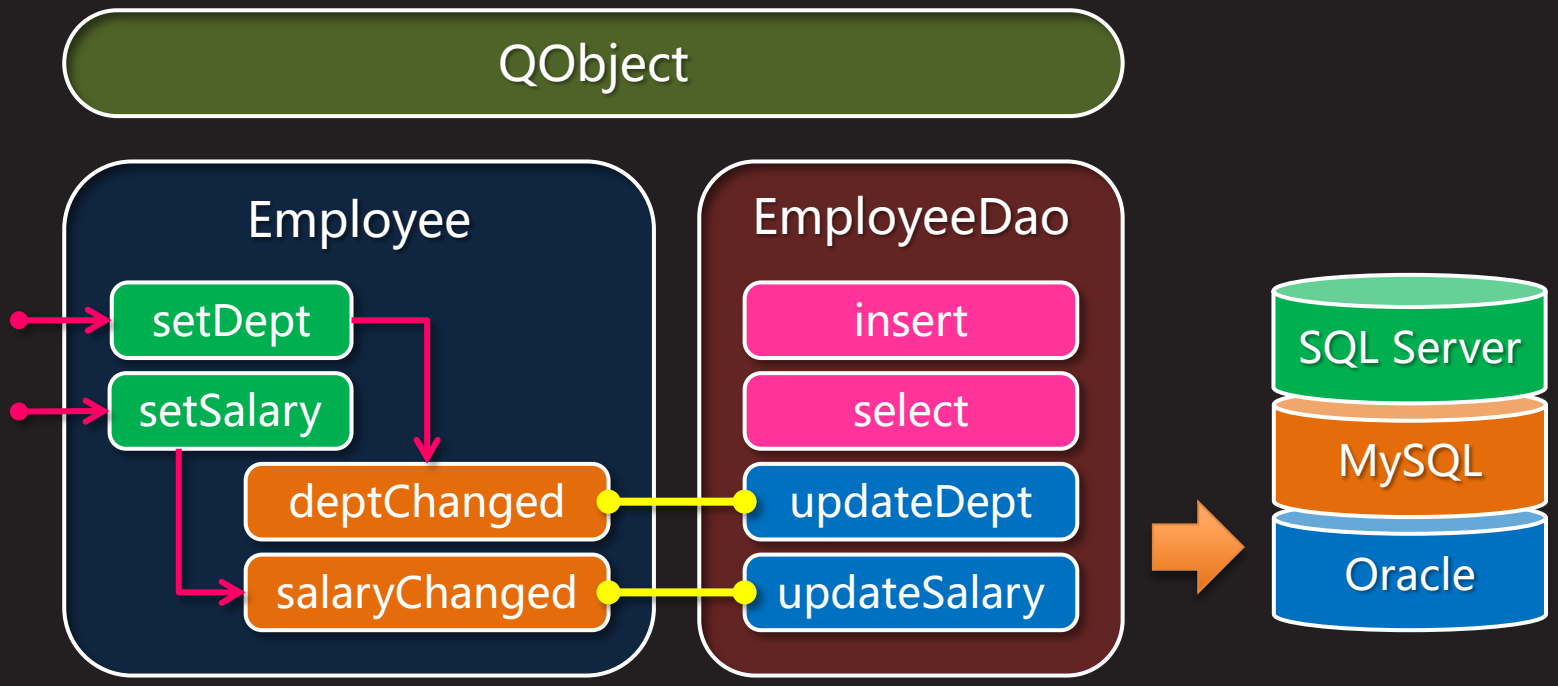
- 显示系统时间



在更大范围内使用信号和槽

- 信号和槽实际上是Qt提供的一种对象间通信机制，这种机制本身是在QObject中实现的，任何QObject的子类都可以使用这种机制，并不只局限于图形用户界面编程

知识讲解



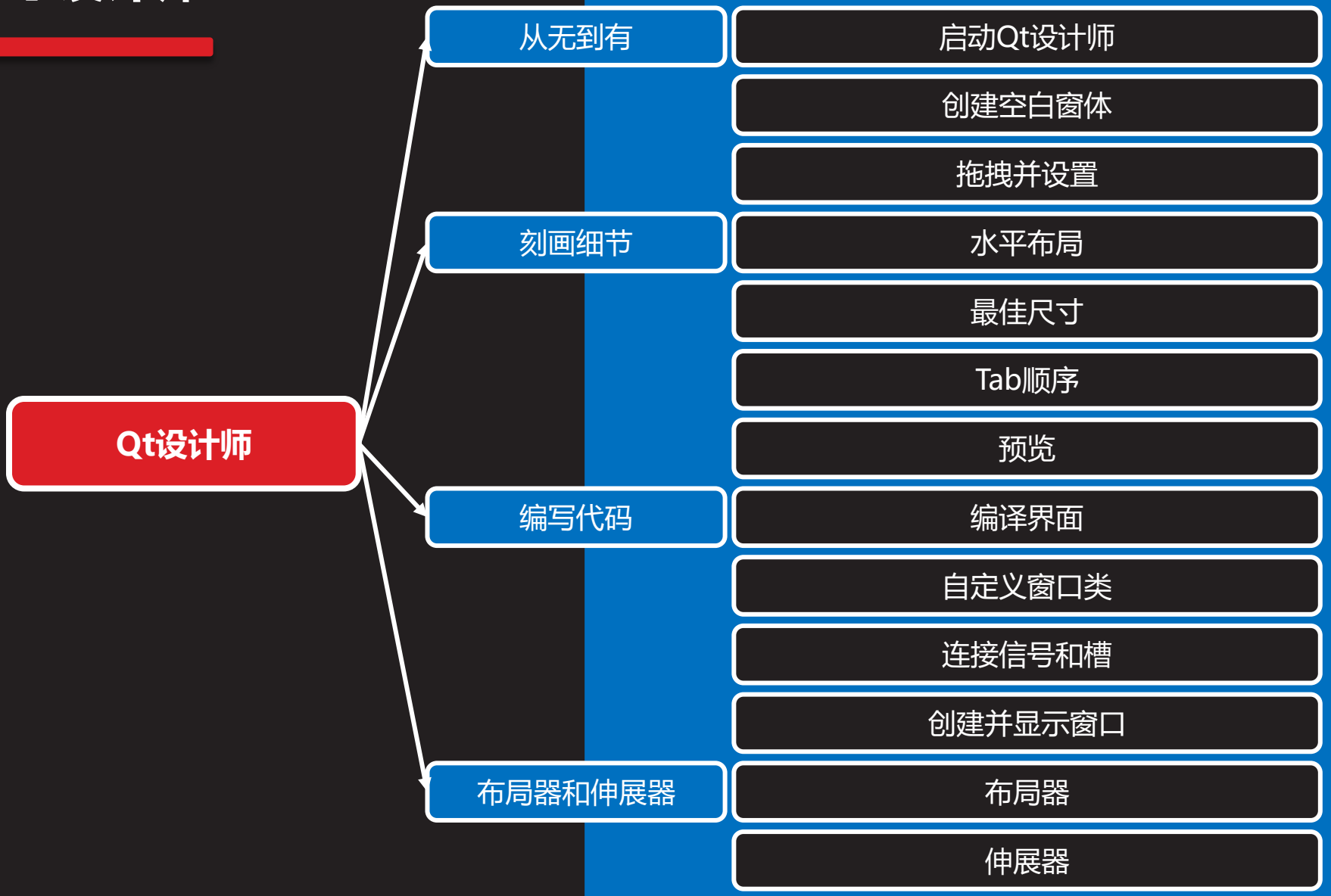
更新员工信息

【参见：TTS COOKBOOK 】

课堂练习

- 更新员工信息

Qt设计师

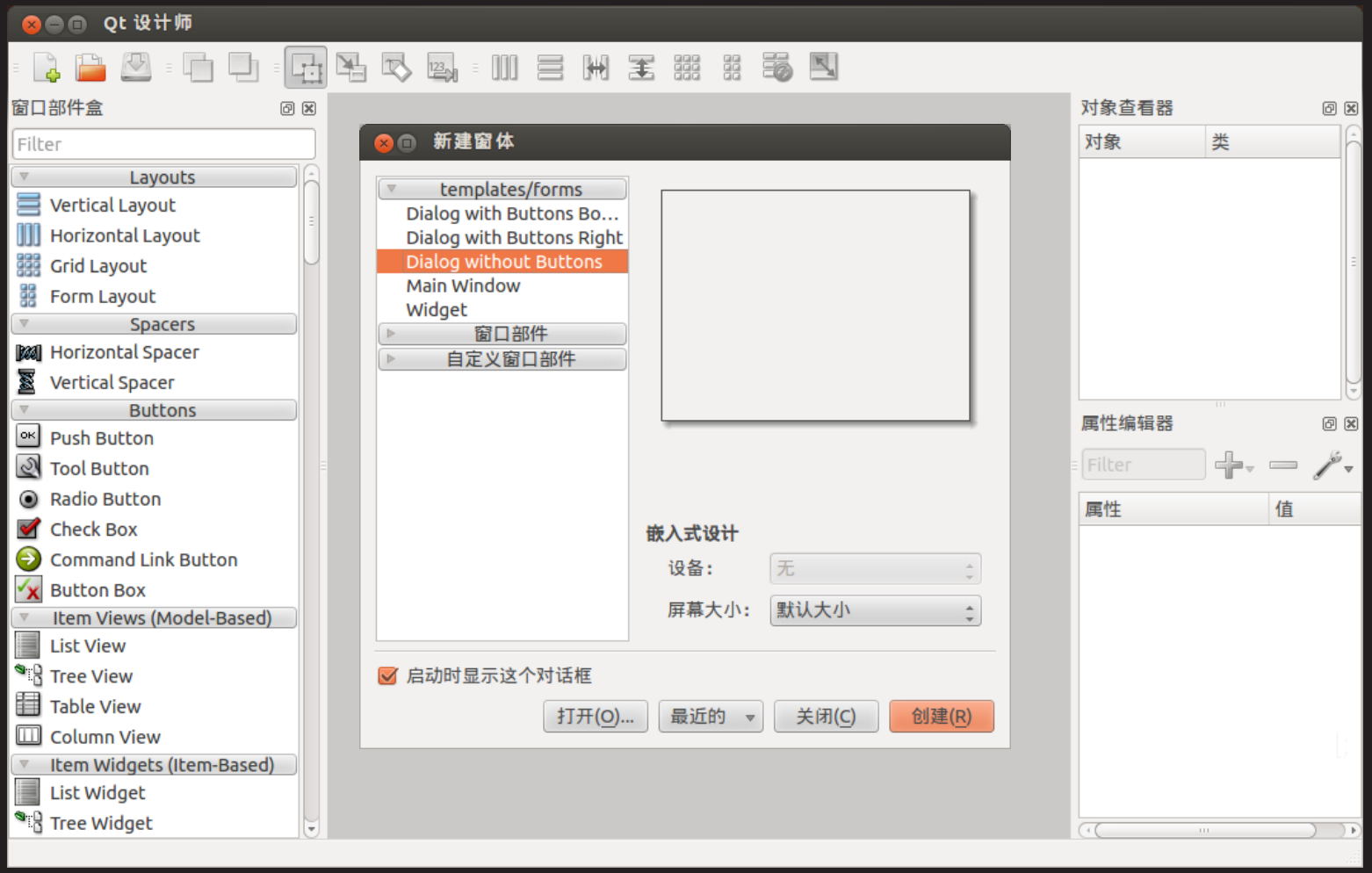

Qt设计师
从无到有
启动Qt设计师
创建空白窗体
拖拽并设置
刻画细节
水平布局
最佳尺寸
Tab顺序
预览
编写代码
编译界面
自定义窗口类
连接信号和槽
创建并显示窗口
布局器和伸展器
布局器
伸展器

Technology Tarena 达内科技

从无到有

启动Qt设计师

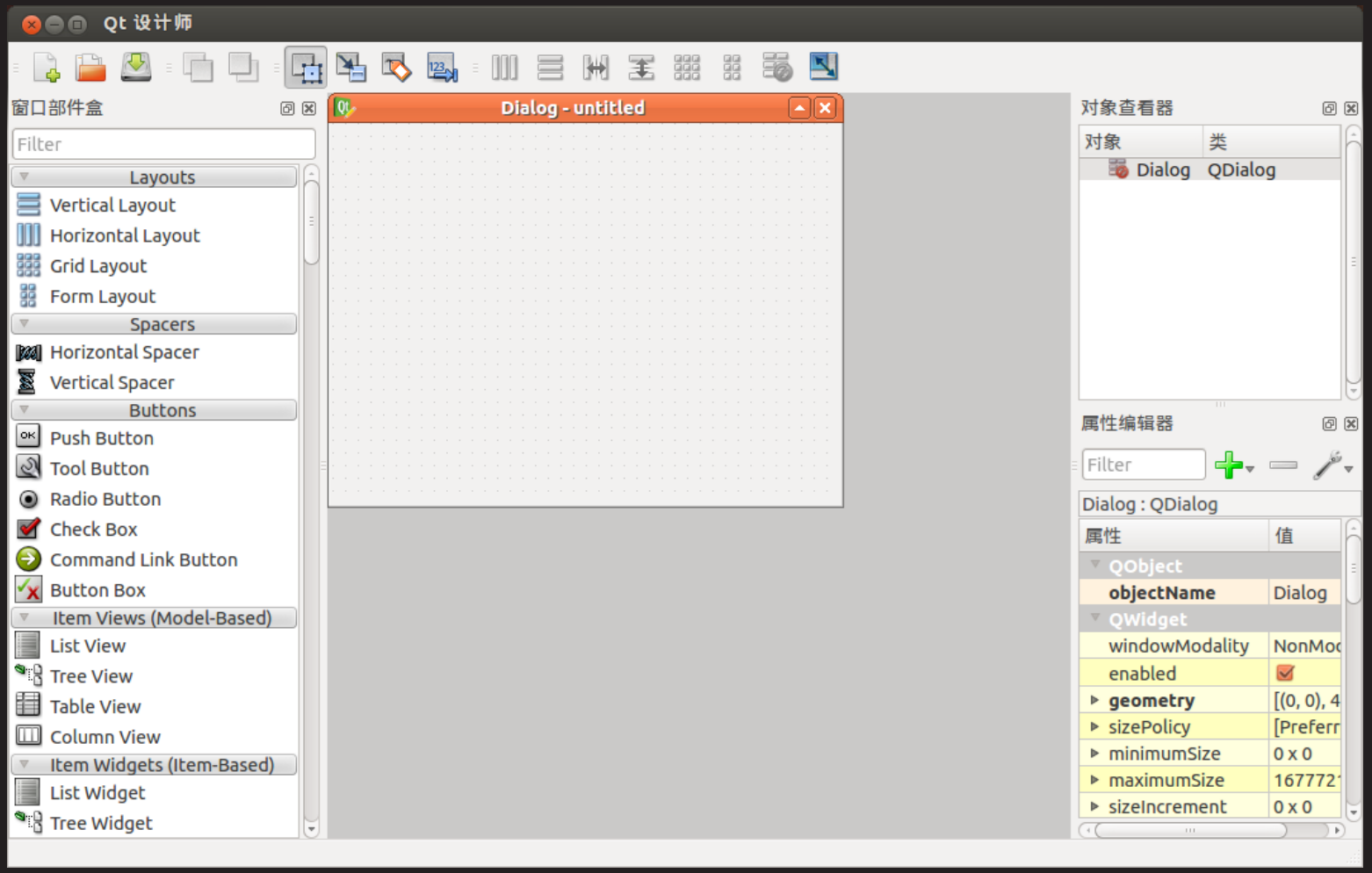
- 从命令行终端创建并进入工程目录Calculator，执行designer命令，启动Qt设计师



知识讲解

创建空白窗体

- 在"新建窗体"对话框中选择"Dialog without Buttons"按"创建"，显示空白窗体



知识讲解

拖拽并设置

- 选中新建窗体，设置属性
  - objectName : CalculatorDlg
  - windowTitle : 计算器
- 从工具箱中拖出"Line Edit"部件放入窗体，设置属性
  - objectName : m_editX
  - alignment : AlignRight, AlignVCenter
- 从工具箱中拖出"Label"部件放入窗体，设置属性
  - objectName : m_labOp
  - text : +
  - alignment : AlignHCenter, AlignVCenter

知识讲解

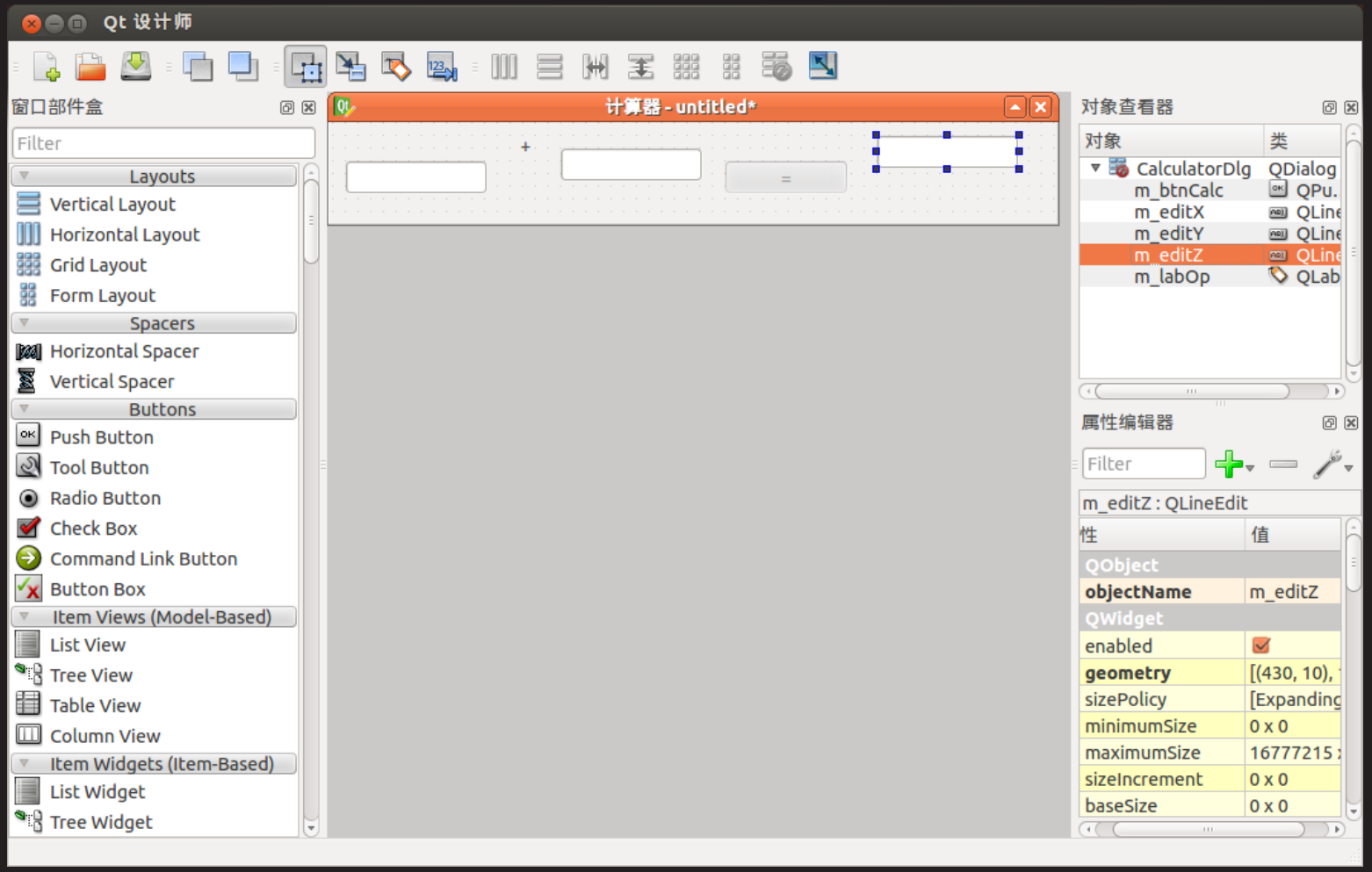
拖拽并设置（续1）

- 从工具箱中拖出"Line Edit"部件放入窗体，设置属性
  - objectName : m_editY
  - alignment : AlignRight, AlignVCenter
- 从工具箱中拖出"Push Button"部件放入窗体，设置属性
  - objectName : m_btnCalc
  - enabled : false
  - text : =
- 从工具箱中拖出"Line Edit"部件放入窗体，设置属性
  - objectName : m_editZ
  - alignment : AlignRight, AlignVCenter
  - readOnly : true

知识讲解

拖拽并设置（续2）

- 此时不必太刻意于对话框窗体及其子部件的位置和大小，只需粗略摆放到一个大概的相对位置即可



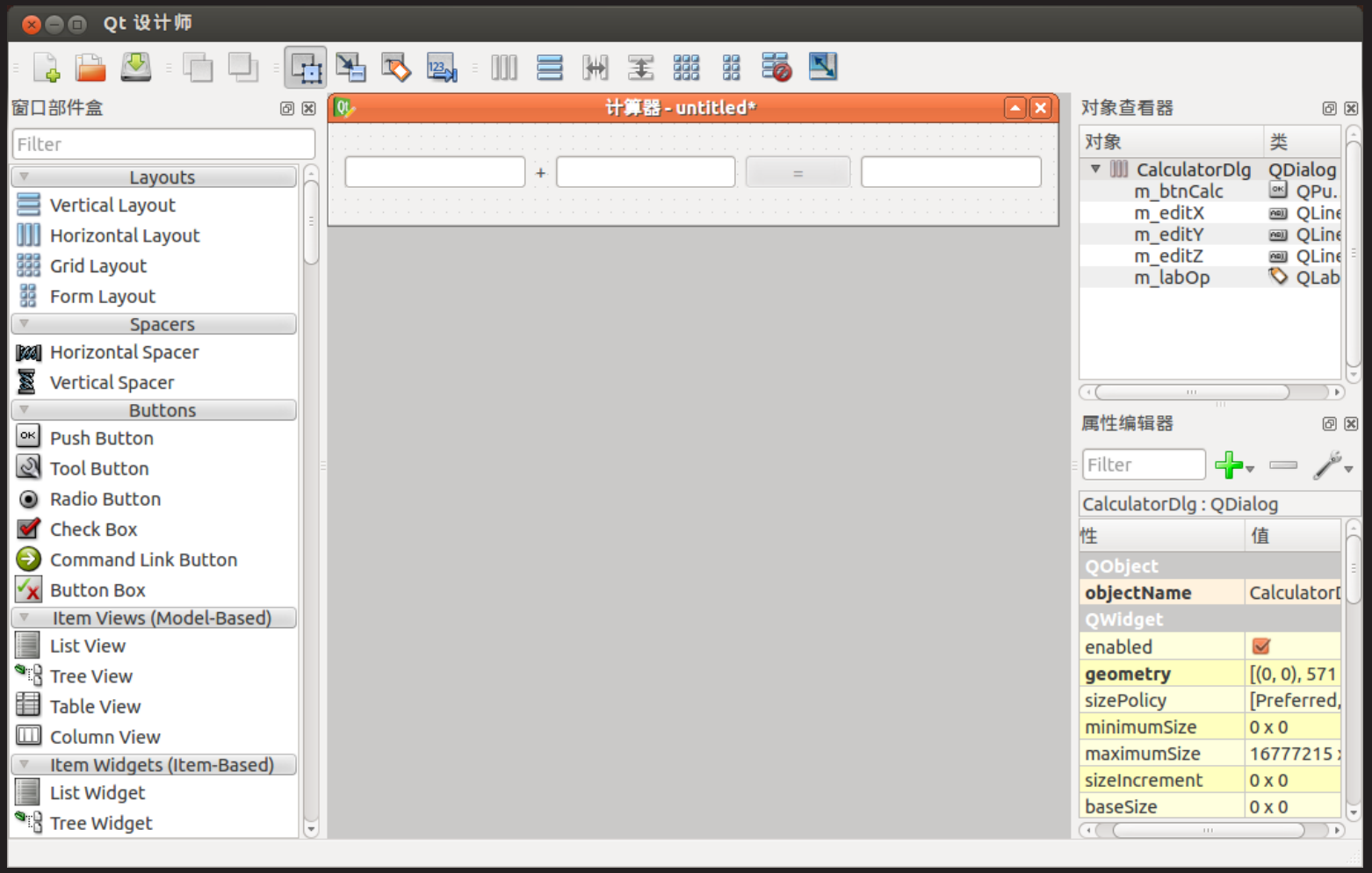
知识讲解

Technology Tarena 达内科技

刻画细节

水平布局

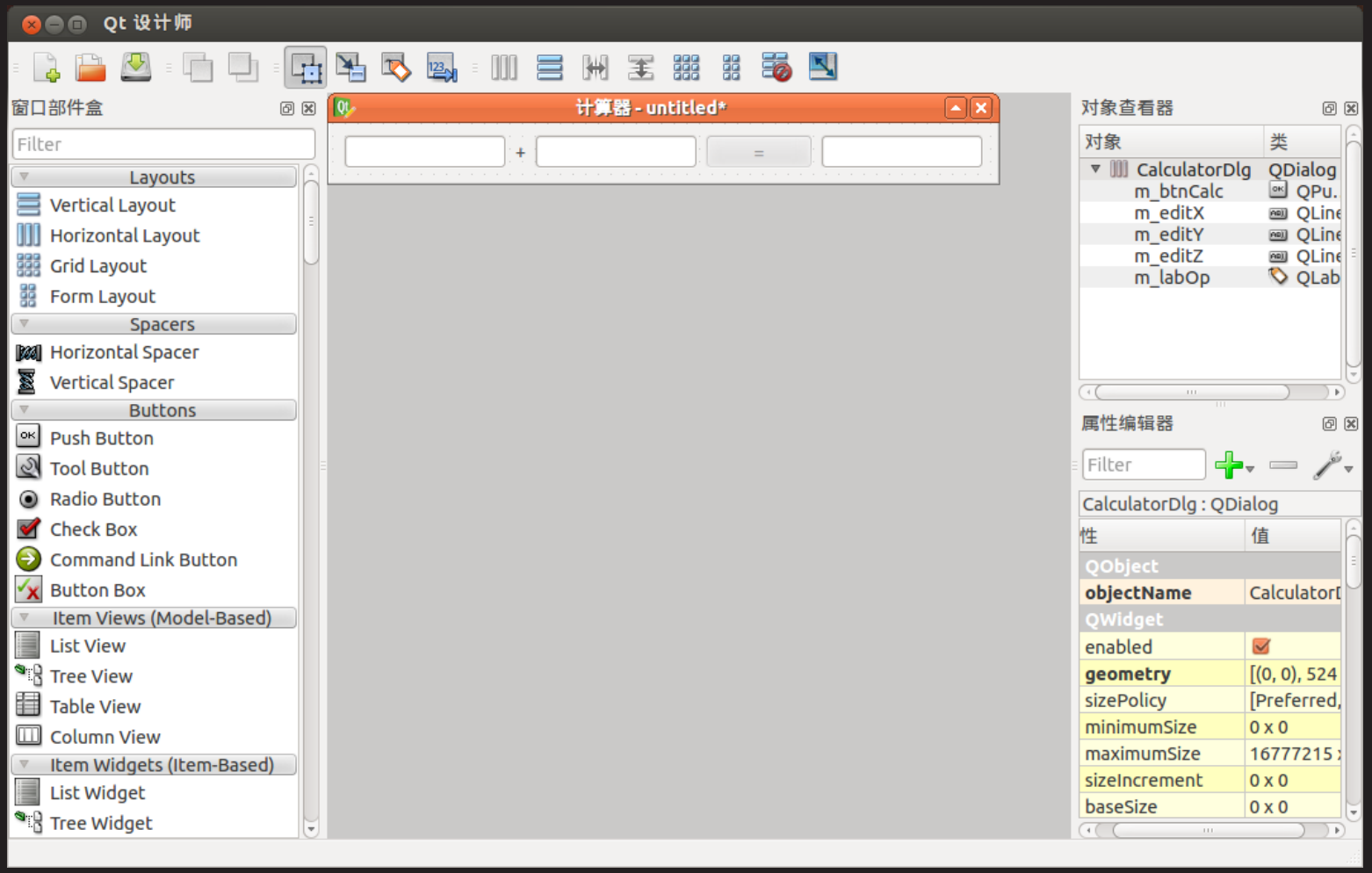
- 点击窗体空白区域，选择右键菜单"布局/水平布局"，对话框窗体中的子部件自动沿水平方向排列整齐



知识讲解

最佳尺寸

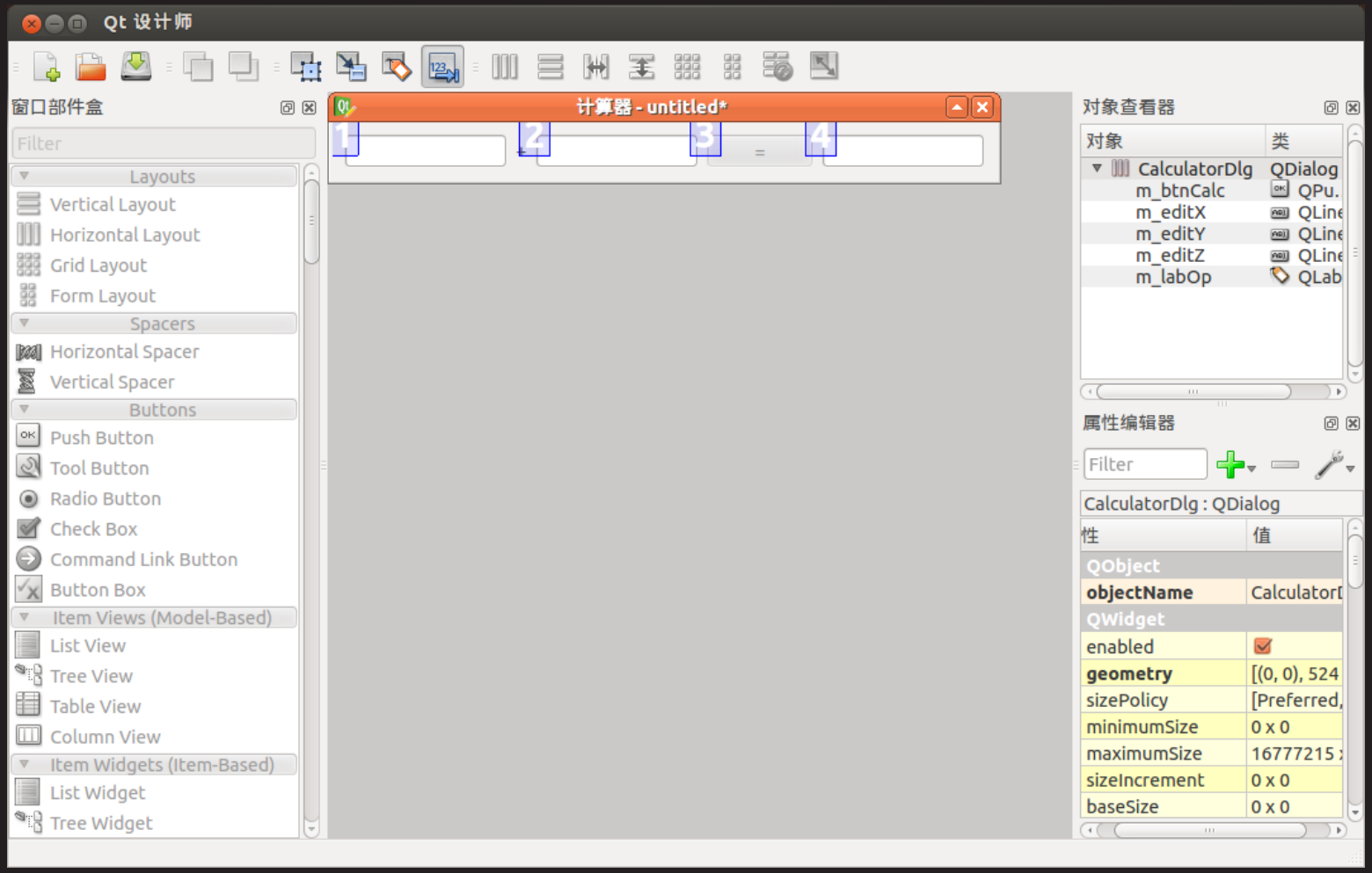
- 点击窗体空白区域，选择右键菜单"布局/调整大小"，对话框窗体的大小自动调整至最佳状态



知识讲解

Tab顺序

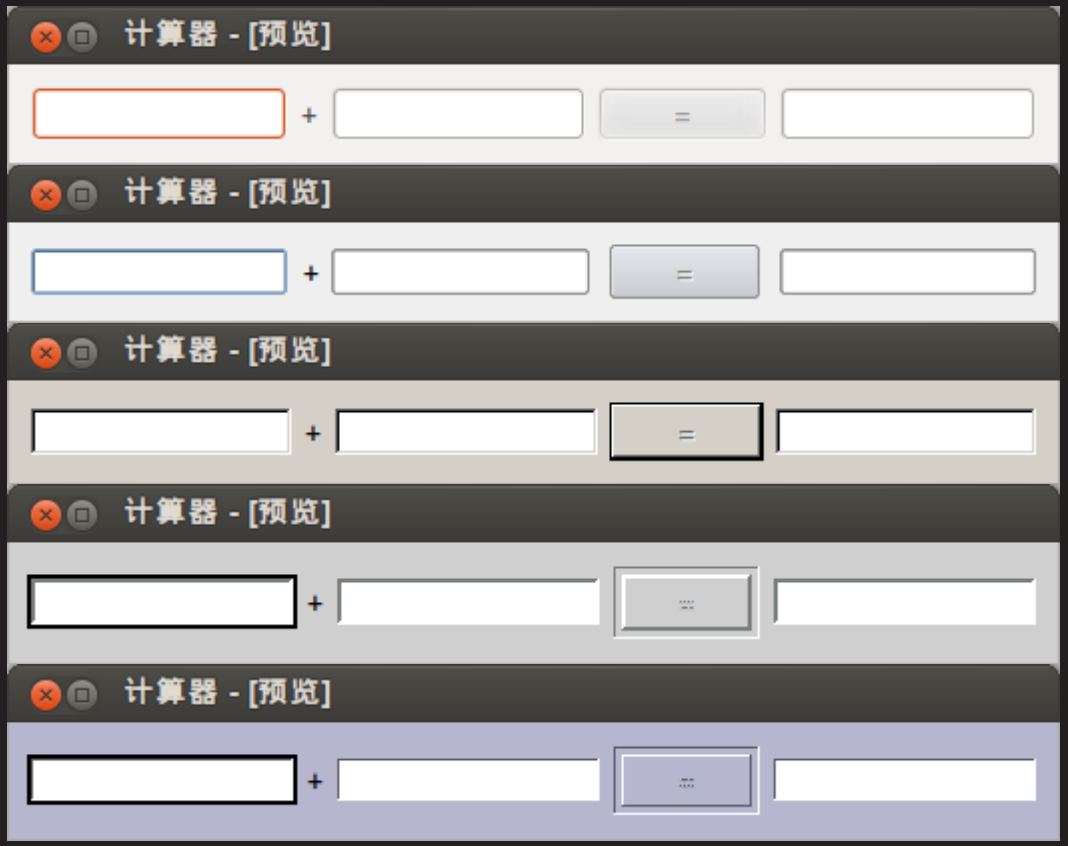
- 选择顶层菜单"编辑/编辑Tab顺序"，进入Tab键顺序设置模式，按照所期望的焦点切换顺序依次单击每个部件



知识讲解

预览

- 选择顶层菜单"编辑/编辑窗口部件"，退出Tab键顺序设置模式。选择"窗体/预览于/<某某>风格"，可以预览对话框在不同风格桌面系统上的显示外观
  - GTK+风格
  - Plastique风格
  - Windows风格
  - Motif风格
  - CDE风格



知识讲解

Technology Tarena 达内科技

编写代码

编译界面

- 在Qt设计师中选择菜单"文件/保存"，将用户界面保存到工程目录下的CalculatorDlg.ui文件中。至此界面设计工作完成，可以退出Qt设计师了。以后如果需要修改界面，只需在Qt设计师中重新打开CalculatorDlg.ui文件即可
- 界面描述文件CalculatorDlg.ui是一个XML格式的文本文件，可用任何一款文本编辑器打开、浏览并修改(不建议)，其中包含了对用户界面的完整描述
- C++编译器无法理解界面描述文件，C++语法也不支持从这些描述性信息中获取类型抽象，因此必须借助于界面编译器uic，把.ui文件编译为等价的.h文件，才能放到C++代码中使用
  - $ uic CalculatorDlg.ui -o ui_CalculatorDlg.h

知识讲解

自定义窗口类

知识讲解

- 打开uic编译生成的ui_CalculatorDlg.h头文件
  - class Ui_CalculatorDlg {
    void setupUi (QDialog* CalculatorDlg) { ... } };
  - namespace Ui {
    class CalculatorDlg : public Ui_CalculatorDlg {}; }
- 不难看出ui_CalculatorDlg.h文件中的Ui::CalculatorDlg类其实就是CalculatorDlg.ui文件所描述界面的C++翻版，其中既有与所有窗口部件相对应的成员变量，也有对界面进行初始化的setupUi()成员函数
- 剩下的工作就是从QDialog和Ui::CalculatorDlg类共同派生出自己的子类，以补充应用程序所需要的定制特性

自定义窗口类（续1）

- 在工程目录下编辑CalculatorDlg.h文件
  -
```
#include <QDialog>
#include "ui_CalculatorDlg.h"
class CalculatorDlg : public QDialog,
  public Ui::CalculatorDlg {
  Q_OBJECT
public:
  CalculatorDlg (QWidget* parent = NULL);
private slots:
  void enableCalcButton (void);
  void calcClicked (void);
};
```

知识讲解

连接信号和槽

- 在CalculatorDlg的构造函数中初始化界面，连接信号和槽
  - CalculatorDlg::CalculatorDlg (QWidget* parent) :
    QDialog (parent) {
    setupUi (this);
    m_editX->setValidator (new QDoubleValidator (this));
    connect (m_editX,
      SIGNAL (textChanged (const QString&)),
      this, SLOT (enableCalcButton (void)));
    m_editY->setValidator (new QDoubleValidator (this));
    connect (m_editY,
      SIGNAL (textChanged (const QString&)),
      this, SLOT (enableCalcButton (void)));
    connect (m_btnCalc, SIGNAL (clicked (void)),
      this, SLOT (calcClicked (void)));
  }

知识讲解

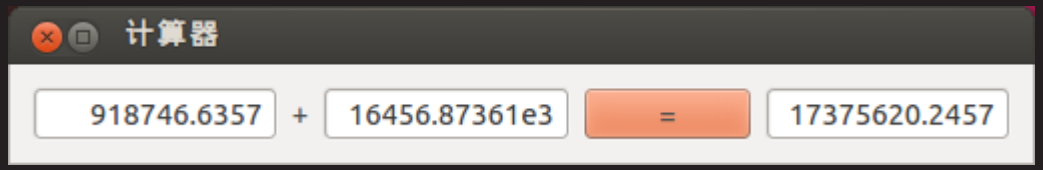
连接信号和槽（续1）

- 在CalculatorDlg.cpp中给出这两个槽函数的定义
  -
```
void CalculatorDlg::enableCalcButton (void) {
  bool bXOk, bYOk;
  m_editX->text ().toDouble (&bXOk);
  m_editY->text ().toDouble (&bYOk);
  m_btnCalc->setEnabled (bXOk && bYOk);
  if (! m_btnCalc->isEnabled ())
    m_editZ->setText ("");
}
```
  -
```
void CalculatorDlg::calcClicked (void) {
  m_editZ->setText (QString::number (
    m_editX->text ().toDouble () +
    m_editY->text ().toDouble (), 'g', 15));
}
```

知识讲解

创建并显示窗口

- 在Calculator.cpp中定义main函数，创建并显示窗口
  -
```
#include <QApplication>
#include "CalculatorDlg.h"
int main (int argc, char* argv[]) {
  QApplication app (argc, argv);
  CalculatorDlg dlg;
  dlg.show ();
  return app.exec ();
}
```
- 最后用qmake生成Makefile，用make编译链接，运行测试



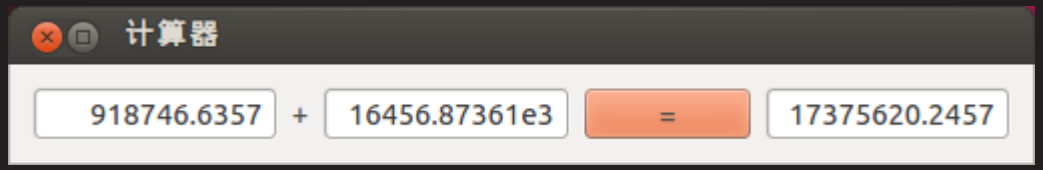
知识讲解

利用Qt设计师重构计算器

【参见：TTS COOKBOOK 】

课堂练习

- 利用Qt设计师重构计算器



Technology Tarena 达内科技

布局器和伸展器

布局器

- Qt中的布局器就是一个能够对其所负责窗口部件的尺寸大小和位置进行设置的对象。Qt提供了三个布局器类型
  - **QHBoxLayout**：水平布局器
    - ✓ 在水平方向上，按照从左到右的顺序，排列窗口部件
  - **QVBoxLayout**：垂直布局器
    - ✓ 在垂直方向上，按照从上到下的顺序，排列窗口部件
  - **QGridLayout**：栅格布局器
    - ✓ 将窗口部件排列在一个表格中
- 当把一个包含了若干窗口部件的布局器安装到窗口上时，这些窗口部件会自动成为被安装布局器的窗口的子窗口
- 程序员所要做的只是将需要被布局的部件放到一个布局器里，它会自动维持窗口大小改变时部件布局的平稳性

知识讲解

布局器（续1）

Technology Tarena 达内科技

知识讲解

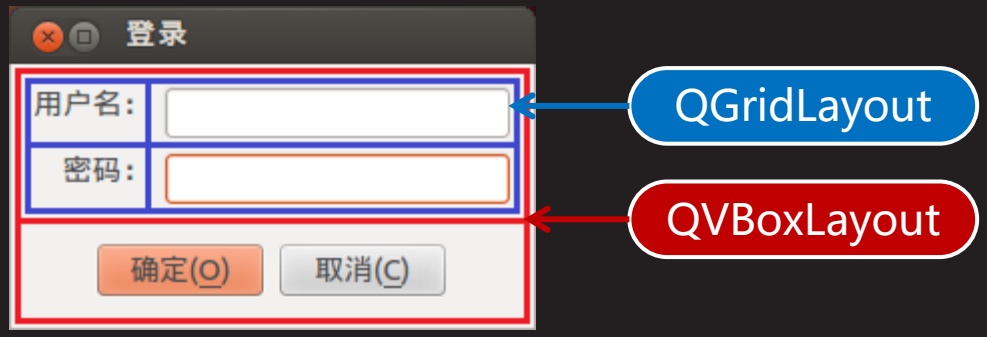
- 例如下面的窗口，如果不使用布局器，而是在代码中把每一个部件的位置和大小全部写死，窗口尺寸一旦改变

- 但如果使用了布局器，情况就不同了，编辑框会随着窗口变宽而变宽，以容纳更多的文本，而按钮则始终居中

- 这种位置和大小同步变化的特性完全由布局器自动处理

布局器（续2）

- 以前面的登录窗口为例，这里实际上使用了两个布局器

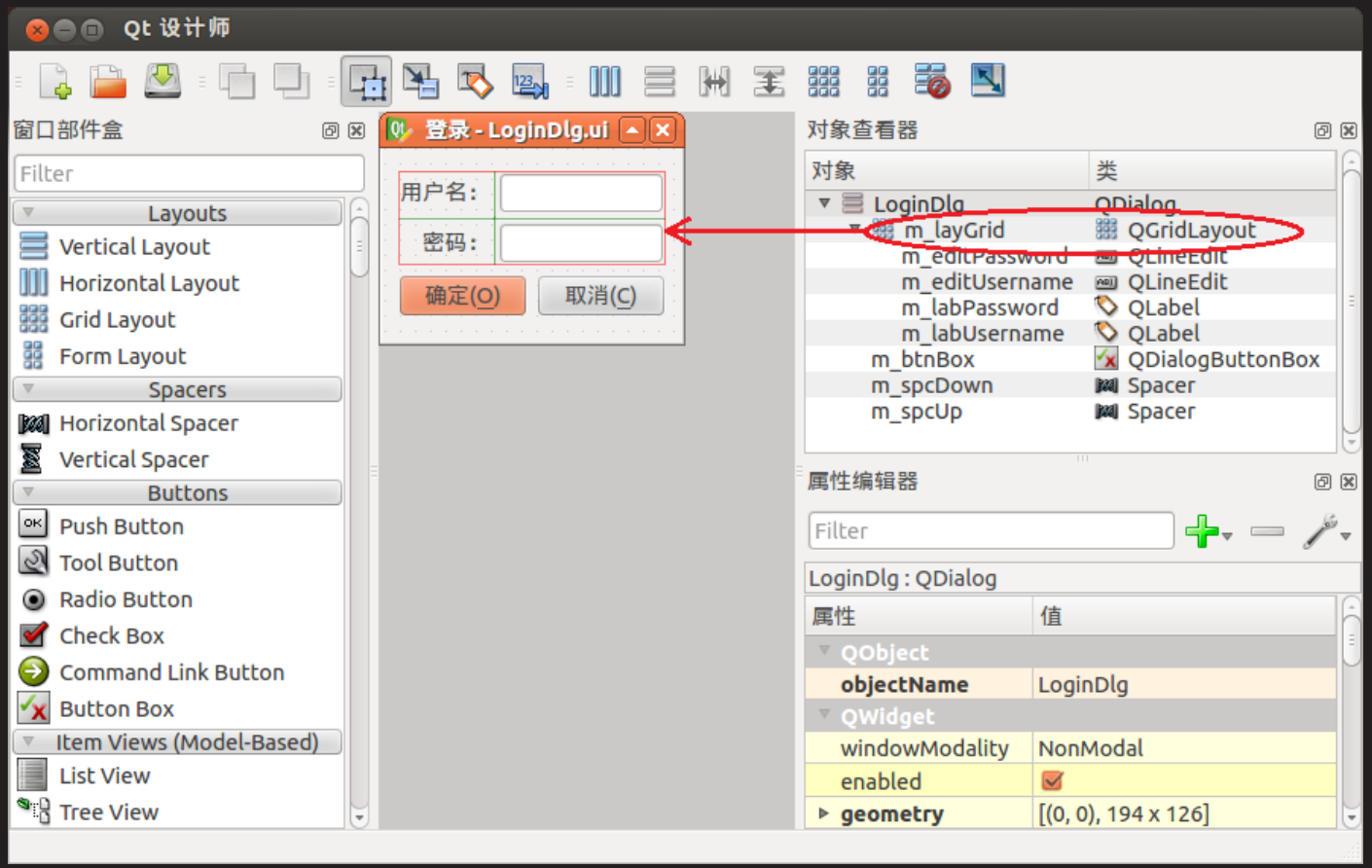


- QGridLayout* layGrid = new QGridLayout;
  layGrid->addWidget (labUsername, 0, 0);
  layGrid->addWidget (m_editUsername, 0, 1);
  layGrid->addWidget (labPassword, 1, 0);
  layGrid->addWidget (m_editPassword, 1, 1);
- QVBoxLayout* layVer = new QVBoxLayout;
  layVer->addLayout (layGrid);
  layVer->addWidget (btnBox);
- setLayout (layVer);

知识讲解

布局器（续3）

- 在Qt设计师中使用布局器，只需在按住<Ctrl>键的同时，用鼠标点选需要放入布局中的部件，在同时选中的多个部件上选择右键菜单"布局/<某某>布局"即可



知识讲解

伸展器

知识讲解

- 默认情况下，布局器总是试图以最均匀的方式排布其中的窗口部件，但有时候这样的效果未必是用户所期望的
- 问题是显见的，当窗口纵向尺寸变大时，垂直布局器将上部的栅格布局和下部的按钮框呈均匀排布，相当于增大了按钮和编辑框的距离，而用户可能更希望它们任何时候都能被紧凑地放在一起

- 伸展器就象一根弹簧，总是试图占据尽可能多的空间，并在其伸展方向上将其它部件挤压到某个固定的位置

伸展器（续1）

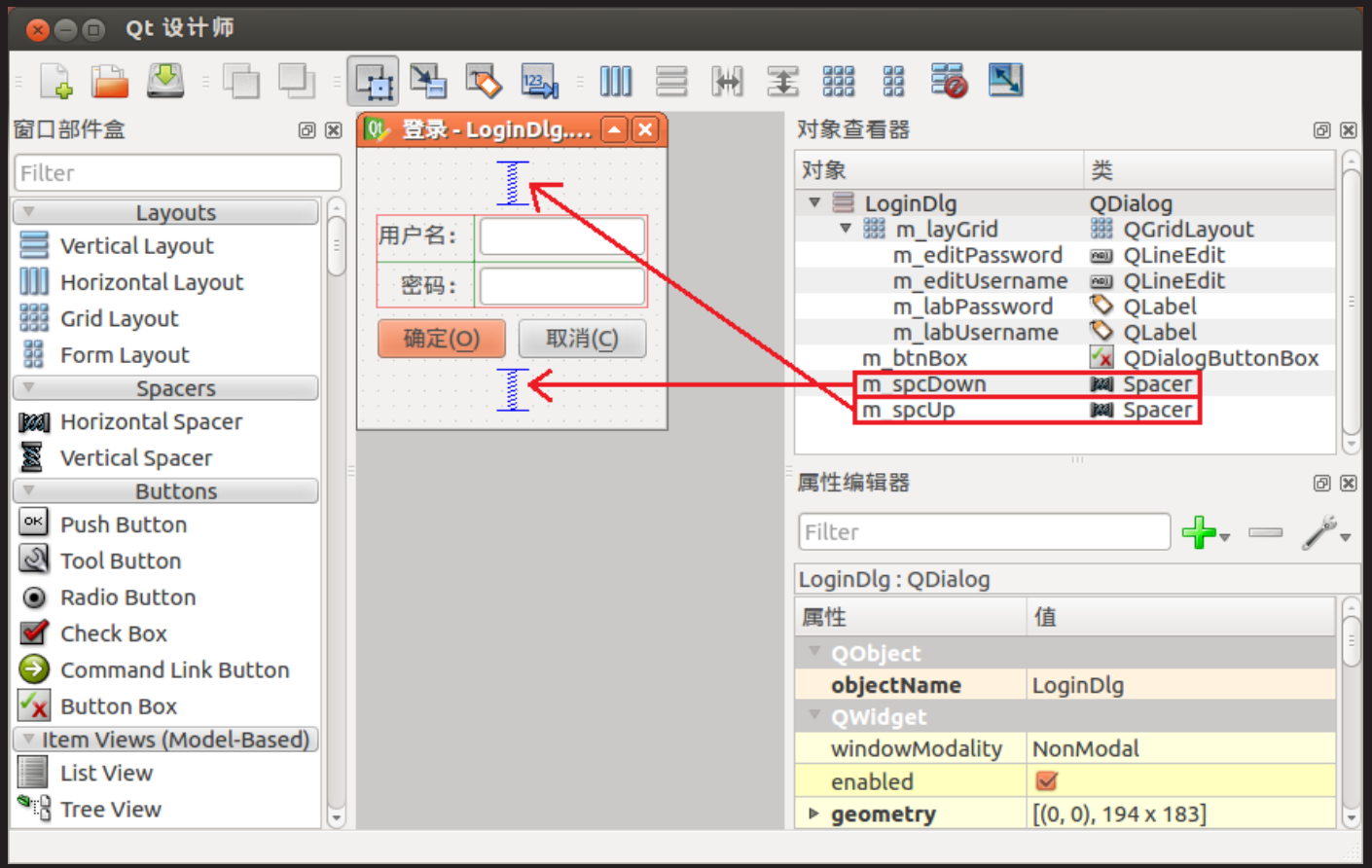
知识讲解

- 以前面的登录窗口为例，不妨在栅格布局之上和按钮框之下各安装一个伸展器，无论窗口纵向尺寸如何变化，两根弹簧在垂直方向上合力将所有的部件紧紧压在一起
  - QVBoxLayout* layVer = new QVBoxLayout;
    layVer->addStretch ();
    layVer->addLayout (layGrid);
    layVer->addWidget (btnBox);
    layVer->addStretch ();
    setLayout (layVer);

- 利用伸展器可以很容易地实现类似左对齐、右对齐、水平居中，或顶对齐、底对齐、垂直居中等显示效果

伸展器（续2）

- 在Qt设计师中使用伸展器，直接从工具箱中拖出"Horizontal Spacer"或者"Vertical Spacer"放到窗体中正确的位置上即可。伸展器通常需要跟布局器配合使用



知识讲解

Technology Tarena 达内科技

总结和答疑