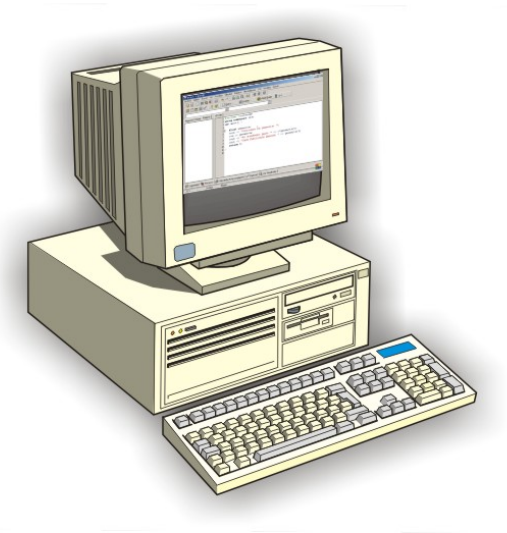


# Guión 2

## *Interfaces gráficos en Qt con Qt-designer*

*Noviembre de 2011*

DECSAI
Departamento de Ciencias de la Computación e I.A.
Universidad de Granada

Nuevas Tecnologías de la Programación

Curso 2011/2012

# Índice

# 1. Introducción

Qt designer es una utilidad de Trolltech (`http://trolltech.com`), empresa comprada recientemente por Nokia (`http://qt.nokia.com/`), para diseñar y construir interfaces gráficos de usuario (GUIs) con componentes Qt. Permite diseñar y construir widgets y diálogos usando *forms* con los mismos widgets que se usarán en la aplicación. Los componentes creados con Qt designer pueden hacer uso del mecanismo de señales-slots. El GUI resultante puede previsualizarse para comprobar que tiene el aspecto y se comporta según deseábamos.

Qt Designer puede usarse para desarrollar la aplicación completa o algunos de los forms solamente. Los forms que se crean usando Qt Designer acaban traduciéndose en código C++.

Existe otra utilidad llamada Qt Creator que incluye a Qt Designer como una de sus utilidades. Qt Creator permite el desarrollo de la aplicación completa, y no sólo el diseño de los forms.

Para programar en qt en las aulas de prácticas de la ETSIIT, usaremos Ubuntu 10.04. Comenzaremos descargando alguno de los programas de ejemplo del libro *C++ GUI Programming with Qt 4* (2ª edición), que están disponibles en la página web de la asignatura, en el apartado de otros temas docentes (`http://decsai.ugr.es/~acu/NTP/CursoActual/otros.html`). Concretamente aparecen en el último enlace del **Módulo 2**. Descarga por ejemplo el fichero `hello.cpp` que está en la carpeta `chap01/hello`. Debes guardar este fichero en tu cuenta, en una carpeta llamada `hello`. Los pasos para obtener el ejecutable de este programa serían:

- Colocándote dentro de la carpeta `hello`, ejecuta la utilidad **qmake** para obtener el fichero de proyecto (`hello.pro`):

  ```
  > qmake-qt4 -project
  ```

- Ejecuta la utilidad **qmake** para obtener el fichero `Makefile`:

  ```
  > qmake-qt4
  ```

- Compila el programa a partir del anterior fichero `Makefile`:

  ```
  > make
  ```

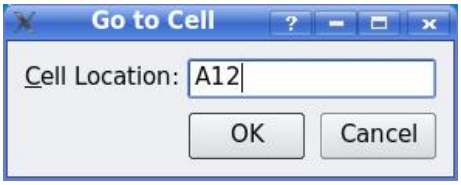
- Ejecuta el programa generado:

  ```
  > ./hello
  ```

Puedes probar ahora a descargar otros de los programas de ejemplo, para obtener el ejecutable de la misma forma.

# 2. Introducción a Qt designer

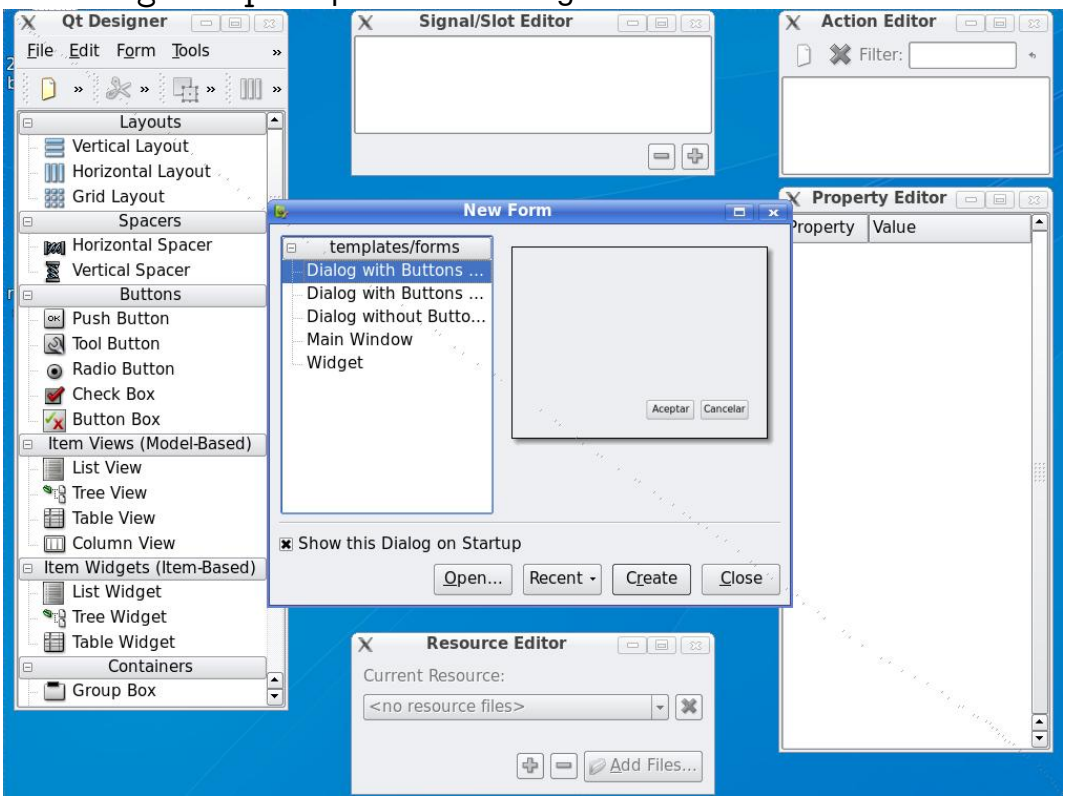
En primer lugar usaremos Qt designer para diseñar el diálogo mostrado en la siguiente figura. Este diálogo se usa en la aplicación *Spreadsheet* (Hoja de cálculo) para solicitar al usuario que introduzca el nombre de una celda en la hoja de cálculo:



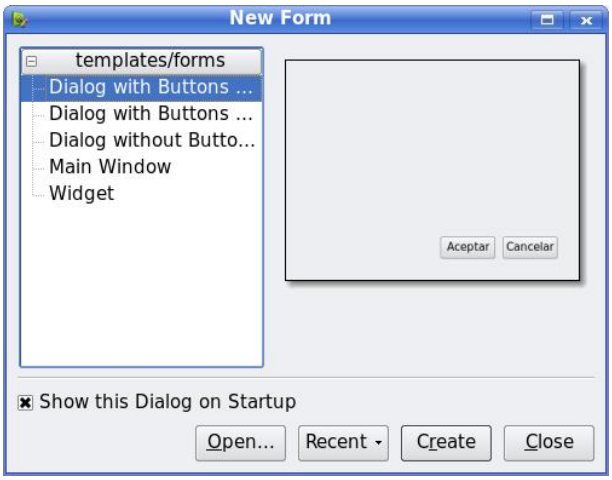

Los pasos que tendremos que llevar a cabo son los siguientes:

- Crear e inicializar los widgets hijos.
- Colocar los widgets hijos en layouts.
- Establecer el orden (*tab*) de los widgets dentro del form.
- Establecer las conexiones señales-slots.
- Implementar los slots.

Para ejecutar Qt designer abre un terminal en linux y ejecuta el comando `designer-qt4`. Aparecerán las siguientes ventanas:

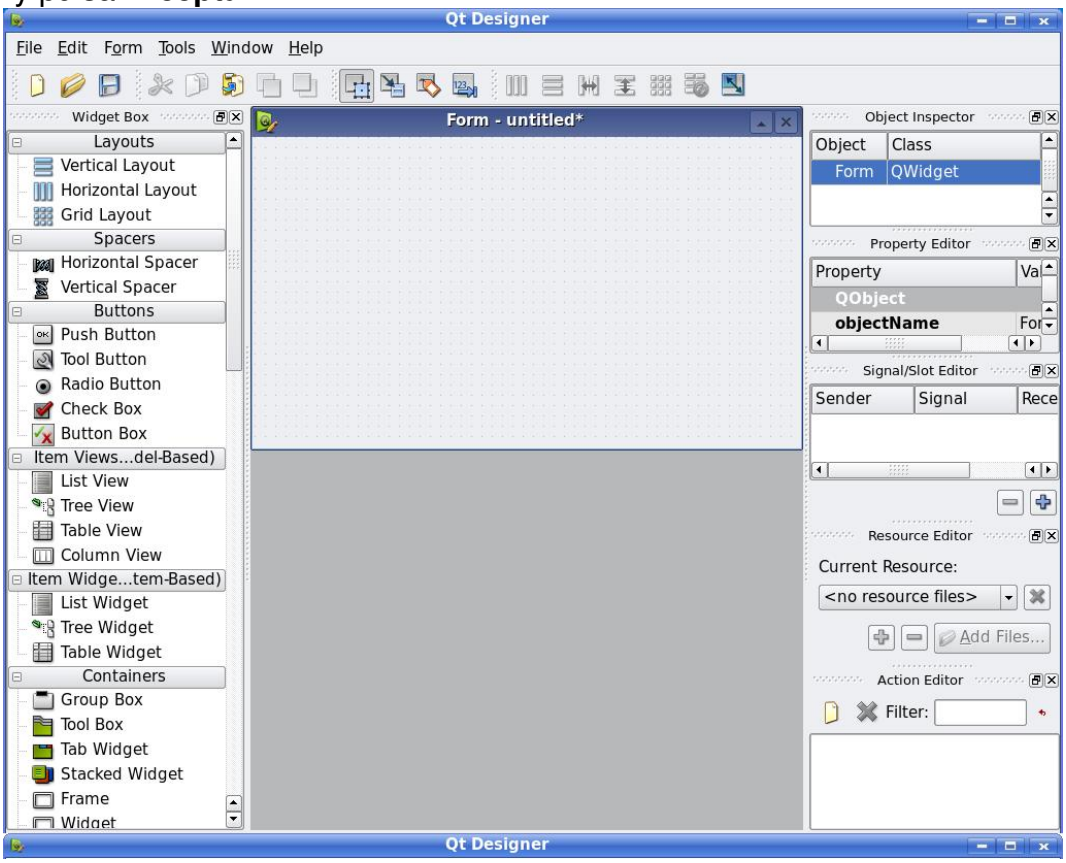



Cuando comienza a ejecutarse Qt Designer, muestra una lista de plantillas (*templates*) para distintos tipos de forms (diálogos, ventana principal, widget).




Selecciona la plantilla **Widget** y pulsa el botón **Create**. Quizás podríamos usar la plantilla *Dialog with Buttons Bottom*, pero para este ejemplo crearemos los botones *OK* y *Cancel* a mano para ver cómo se hace. Ahora debe aparecer una ventana con la cadena *Untitled* en la barra de títulos.

Por defecto, el interfaz de Qt Designer se compone de varias ventanas de alto nivel. Si se prefiere un estilo de interfaz MDI (*Multiple Document Interface*), con una única ventana de alto nivel y varias subventanas, pulsar **Menú Edit** → **Preferences** → **User Interface Mode** → **Docked Window** y pulsar **Aceptar**.



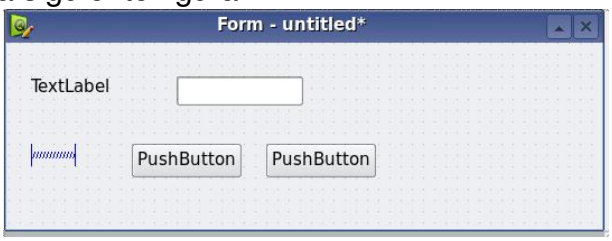

# 3. Creación de widgets hijos

En este paso añadiremos al form los siguientes widgets:

- Una etiqueta (*Label*)
- Un editor de líneas (*Line edit*)
- Espaciado horizontal (*Horizontal spacer*)
- Dos botones (*Push buttons*)

Realizaremos ahora las siguientes etapas:

- Para cada item, arrastrar su nombre o icono desde la caja de Widgets (*Widget Box*) (situada en el lado izquierdo de Qt Designer) hacía el lugar aproximado que debe ocupar en el form. El item de espaciado horizontal será invisible en el form final, pero se muestra en Qt Designer dentro del form como un segmeto de color azul.
- Arrastrar la parte inferior del form, para hacerlo más pequeño según muestra la siguiente figura:




- No emplear mucho tiempo para colocar los items sobre el form, ya que los gestores de posicionamiento se encargarán de ello automáticamente.

# 4. Definición de las propiedades de los widgets

Mediante el editor de propiedades (Property Editor) de Qt Designer definiremos las propiedades de los widgets que hemos colocado previamente en el form. Para ello, seleccionar sucesivamente cada widget e ir cambiando sus propiedades:

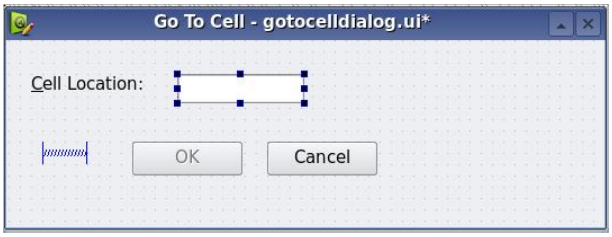
- Etiqueta: Poner la propiedad **objectName** con el valor `label` y la propiedad **text** con `&Cell Location:`.
- Editor de líneas: Poner la propiedad **objectName** con el valor `lineEdit`.
- Botón de la izquierda: Poner la propiedad **objectName** con el valor `okButton`, la propiedad **enabled** al valor `false`, la propiedad **text** al valor `OK` y la propiedad **default** al valor `true`.
- Botón de la derecha: Poner la propiedad **objectName** con el valor `cancelButton` y la propiedad **text** al valor `Cancel`.
- Form: Hacer click sobre el background del form para seleccionarlo, y poner la propiedad **objectName** con el valor `GoToCellDialog` y **windowTitle** con el valor `Go to Cell`.


Para conseguir que el widget Label no muestre el símbolo & haremos lo siguiente:

- Seleccionar **Menú Edit** → **Edit Buddies** para entrar en el *modo de edición de Buddies* (*amiguetes*).
- Selecciona el objeto Label.
- Arrastra la línea roja hacía el editor de líneas, y suelta el botón del ratón.

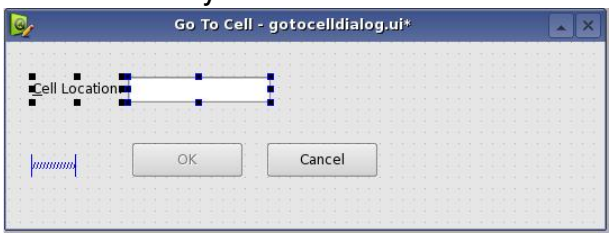
La etiqueta debería mostrar ahora Cell Location y tener al editor de líneas como su buddy.


Regresa de nuevo al *modo normal de edición de widgets*: **Menú Edit** → **Edit Widgets**.



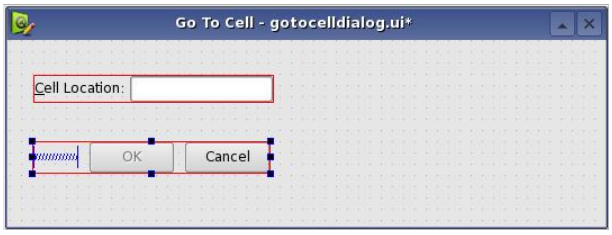

# 5. Colocando los widgets con gestores de posicionamiento

- Selecciona simultáneamente los widgets de la etiqueta y del editor de líneas. Para ello selecciona primero la etiqueta, pulsa y mantén pulsada la tecla `Shift` y selecciona el editor de líneas.




- Para colocar los dos widgets anteriores en un layout horizontal, seleccionamos **Menú Form** → **Lay Out Horinzontally**.

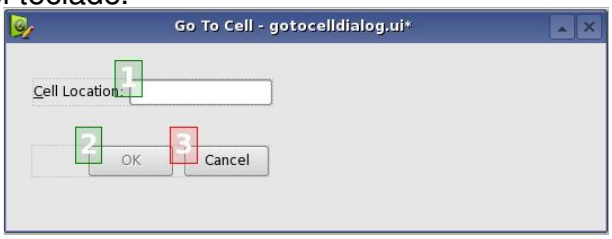

- Selecciona los tres componentes de la parte de abajo (spacer y los dos botones).
- Colócalos también en un layout horizontal de la misma forma que antes.




Las líneas rojas que aparecen en el form muestran los layouts que hemos creado. No aparecerán cuando ejecutemos el programa.

Establezcamos ahora el orden (*tab*) de los widget dentro del form. Esto definirá el orden en que van seleccionándose sucesivamente los widgets con la tecla `Tab`.

- Selecciona **Menú Edit** → **Edit Tab Order**. Aparecerá un número dentro de un rectángulo junto a cada uno de los widgets que pueden aceptar el foco del teclado.
- Haz click sobre cada widget en el orden que quieras que obtengan el foco del teclado.




- Selecciona **Menú Edit** → **Edit Widgets** para volver al modo de edición de widgets.

Para previsualizar el aspecto que tendrá el diálogo:

- Selecciona **Menú Form** → **Preview**.

- Comprueba que el orden tab es el deseado, pulsando sucesivamente la tecla `Tab`.
- Cierra el diálogo usando el botón *close* (la cruz) en su barra de títulos.

# 6. Compilación y ejecución del programa

- Salva el diálogo en un directorio llamado `gotocell` con el nombre `gotocelldialog.ui`.
- Usando un editor de textos, crea un fichero llamado `main.cpp` en el mismo directorio, para contener la función `main()` con el siguiente contenido:

```
#include <QApplication>
#include <QDialog>
#include "ui_gotocelldialog.h"
int main(int argc, char *argv[])
{
    QApplication app(argc, argv);
    Ui::GoToCellDialog ui;
    QDialog *dialog = new QDialog;
    ui.setupUi(dialog);
    dialog->show();
    return app.exec();
}
```

- Dentro del directorio `gotocell`, ejecuta la orden `qmake-qt4 -project` para obtener el fichero `.pro`.
- Ejecuta `qmake-qt4` para obtener el fichero `Makefile`.

  El programa `qmake-qt4` es capaz de encontrar el fichero de interfaz `gotocelldialog.ui` y generar la reglas adecuadas en el fichero makefile para llamar al programa `uic` (el compilador del interfaz de usuario de Qt).
  La utilidad `uic` convierte `gotocelldialog.ui` en código C++ y guarda el resultado en `ui_gotocelldialog.h`.

- Ejecuta make para obtener el fichero ejecutable `gotocell`.

El fichero `ui_gotocelldialog.h` contiene la definición de la clase `Ui::GoToCellDialog`, que representa el código C++ equivalente al fichero `gotocelldialog.ui`. La clase declara variables miembro que guardan los widgets hijos del form y los layouts, y una función `setupUi()` que inicializa el form. La clase generada tiene el siguiente aspecto:

```
class Ui::GoToCellDialog
{
public:
    QLabel *label;
    QLineEdit *lineEdit;
```

```
    QSpacerItem *spacerItem;
    QPushButton *okButton;
    QPushButton *cancelButton;
    ...
    void setupUi(QWidget *widget) {
        ...
    }
};
```

La clase generada no hereda de ninguna otra clase Qt. Cuando usamos el form en `main.cpp`, se crea un `QDialog` y lo pasamos a la función `setupUi()`.

Podemos ya ejecutar el programa, aunque no funciona todavía según deseamos:

- El botón **OK** está siempre desactivado.
- El botón **Cancel** no hace nada.
- El editor de líneas acepta cualquier texto, en lugar de aceptar solamente nombres de celda válidos.

# 7. Heredando de la clase generada por Qt Designer

Para solucionar los problemas que hemos descrito al final de la sección anterior, lo más adecuado es crear una nueva clase que herede de las clases `QDialog` y `Ui::GoToCellDialog` y que implemente la funcionalidad que le faltaba al anterior diálogo. El convenio de nomenclatura que usaremos es darle a la nueva clase el mismo nombre que la clase generada por Qt-Designer pero quitándole el prefijo `Ui::`.

Usando cualquier editor de textos crea el fichero `gotocelldialog.h` con el siguiente contenido:

```
#ifndef GOTOCELLDIALOG_H
#define GOTOCELLDIALOG_H
#include <QDialog>
#include "ui_gotocelldialog.h"
class GoToCellDialog : public QDialog, public Ui::GoToCellDialog
{
    Q_OBJECT
public:
    GoToCellDialog(QWidget *parent = 0);
private slots:
    void on_lineEdit_textChanged();
};
#endif
```

Crea también el fichero `gotocelldialog.cpp` con el siguiente contenido:

```
#include <QtGui>
#include "gotocelldialog.h"
GoToCellDialog::GoToCellDialog(QWidget *parent)
    : QDialog(parent)
{
    setupUi(this);
    QRegExp regExp("[A-Za-z][1-9][0-9]{0,2}");
    lineEdit->setValidator(new QRegExpValidator(regExp, this));
    connect(okButton, SIGNAL(clicked()), this, SLOT(accept()));
    connect(cancelButton, SIGNAL(clicked()), this, SLOT(reject()));
}
void GoToCellDialog::on_lineEdit_textChanged()
{
    okButton->setEnabled(lineEdit->hasAcceptableInput());
}
```

- El constructor hace la llamada a la función `setupUi(this)` para inicializar el form.
- Gracias a la herencia múltiple, podemos acceder directamente a los miembros de la clase `Ui::GoToCellDialog` desde la clase `GoToCellDialog`.
- La función `setupUi()` conecta también automáticamente cualquier slot de `objectName`, cuyo nombre sea `on_objectName_signalName()`, a la señal `signalName()`. En nuestro caso esto significa que `setupUi()` establecerá automáticamente la siguiente conexión:

  ```
  connect(lineEdit, SIGNAL(textChanged(const QString &)),
          this, SLOT(on_lineEdit_textChanged()));
  ```

- En el constructor, también se inicializa un validador para restringir el rango de la entrada. Qt proporciona tres clases validadoras: `QIntValidator`, `QDoubleValidator` y `QRegExpValidator`. En este caso se una la clase `QRegExpValidator` con la expresión regular `"[A-Za-z][1-9][0-9]{0,2}"` que significa *Aceptar una letra minúscula o mayúscula, seguida de un dígito en el rango 1-9, seguido de cero, uno o dos dígitos en el rango 0-9*.

  Al pasar *this* al constructor de `QRegExpValidator`, lo hacemos hijo del objeto `GoToCellDialog`, con lo cual se destruirá automáticamente cuando su padre se destruya.
- Al final del constructor se conecta el botón **OK** al slot `accept()` y el botón **Cancel** al slot `reject()`. Ambos slots cierran el diálogo, aunque `accept()` hace que el valor devuelto por el diálogo sea `QDialog::Accepted` (que es igual a 1), y `reject()` hace que sea `QDialog::Rejected` (que es igual a 0). Esto permitirá saber si el usuario pulsó el botón **OK** o **Cancel** en los programas que usen este diálogo.

- El slot `on_lineEdit_textChanged()` activa o desactiva el botón **OK** dependiendo de si el editor de líneas contiene una celda válida o no. `QLineEdit::hasAcceptableInput()` usa el validador que se creó en el constructor.

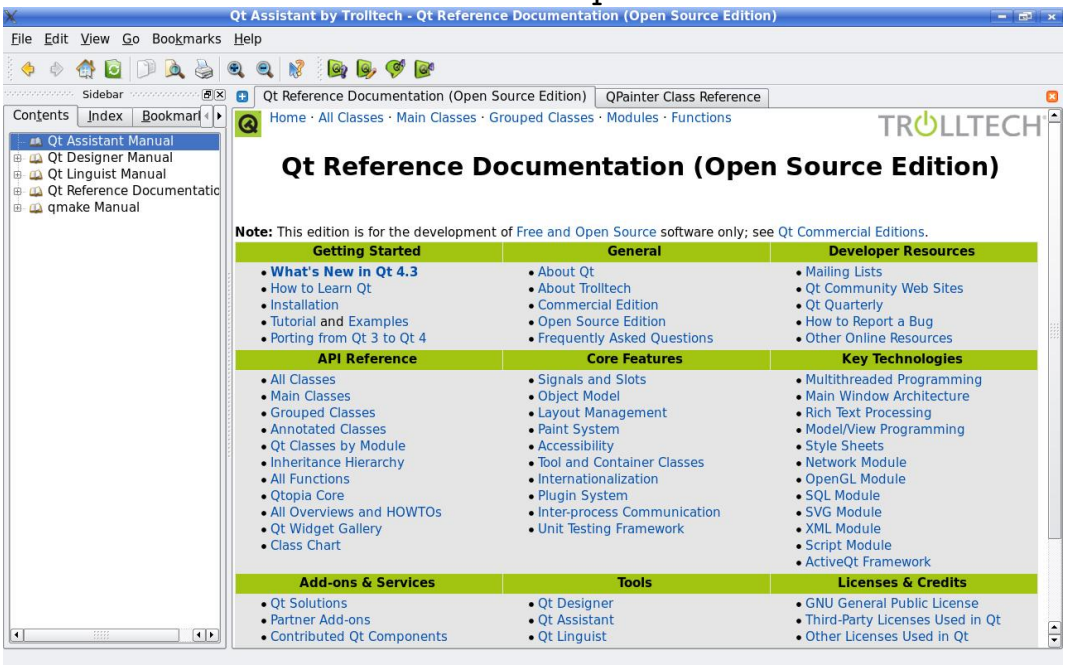
Con esto tenemos el diálogo terminado. Ahora reescribe el fichero `main.cpp` para usar el nuevo diálogo:

```
#include <QApplication>
#include "gotocelldialog.h"
int main(int argc, char *argv[])
{
    QApplication app(argc, argv);
    GoToCellDialog *dialog = new GoToCellDialog;
    dialog->show();
    return app.exec();
}
```

Construye de nuevo la aplicación (`qmake-qt4 -project; qmake-qt4; make`) y ejecútala de nuevo. Ahora puedes comprobar que el botón **OK** se activa al introducir un nombre válido de celda, por ejemplo **A12**.

# 8. Otras utilidades de Qt

Además de Qt-Designer, Qt proporciona otras herramientas muy útiles al desarrollar una aplicación. La primera de ellas es **Qt Assistant**, que nos permite navegar por la documentación on-line de Qt. Entre otras cosas podemos consultar la documentación de cada una de las clases, así como ejemplos de muchas de ellas. Además contiene manuales de las otras utilidades de Qt (Qt Designer, Qt Linguist, qmake). Podemos ejecutar Qt Assistant con el comando `assistant-qt4`:



Especialmente interesantes son los tutoriales sobre Qt Designer de Qt Assistant. Lee ahora el manual *Creating Main Windows in Qt Designer* que mostrará cómo crear la ventana principal de la aplicación, definir la barra de menús con sus menús correspondientes, las barras de utilidades, acciones, etc.

**Qt Designer Manual → Creating Main Windows in Qt Designer**

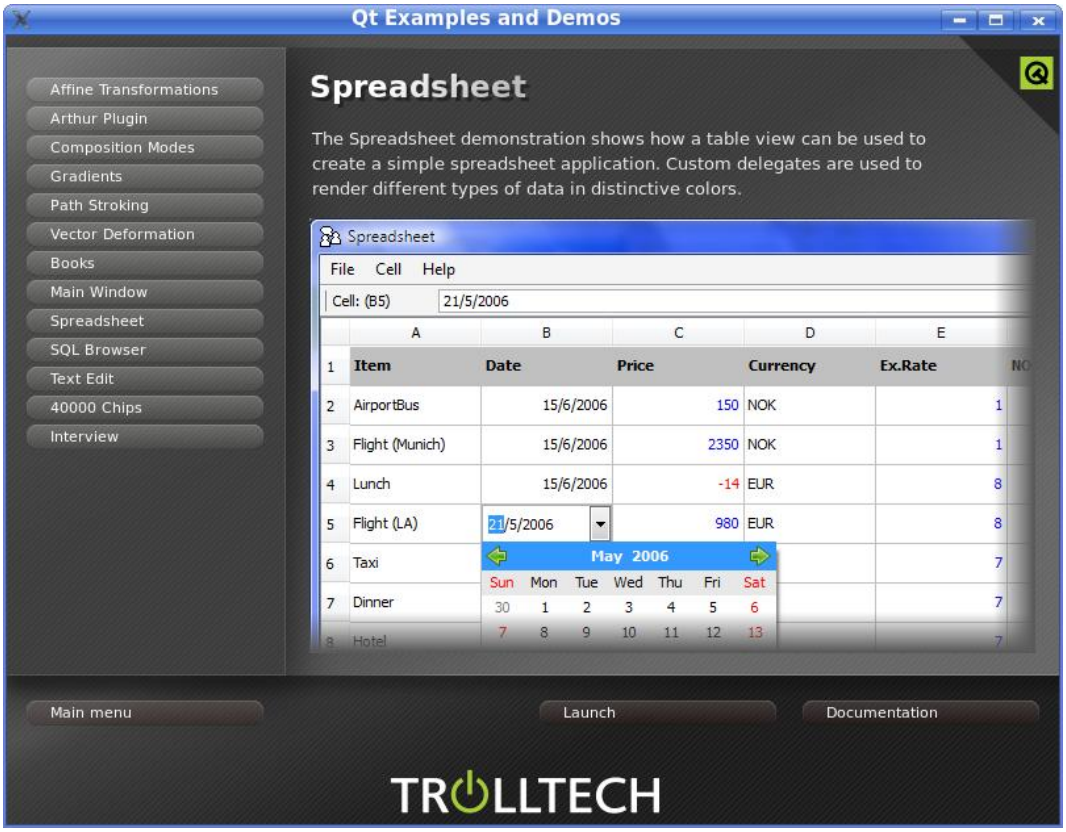
Lee también el manual **Qt Designer's Signals and Slots Editing Mode** que enseña cómo conectar las señales de un objeto con los slots de otro objeto del mismo form.

**Qt Designer Manual → Qt Designer's Signals and Slots Editing Mode**

Por último lee el manual **Using Custom Widgets with Qt Designer**, que nos enseña cómo colocar un *widget optimizado* (como por ejemplo el HexSpinBox que hay en las trasparencias) en un form, mediante Qt Designer.

**Qt Designer Manual → Using Custom Widgets with Qt Designer**

Otra utilidad interesante es **Qt Demo**, que nos permite ejecutar los ejemplos y demos que nos instala Qt. También podemos ver el código fuente de cada programa. Podemos ejecutar Qt Demo con el comando `qtdemo-qt4`:




Finalmente la utilidad **Qt Linguist** permite traducir nuestra aplicación a

otros idiomas. Podemos ejecutar Qt Assistant con el comando `linguist-qt4`: