Réalisation d'interfaces graphiques multi-plateformes

QtCreator

Dernières modifications : janvier 2009

Philippe Lacomme (placomme@sp.isima.fr)

Contenu :

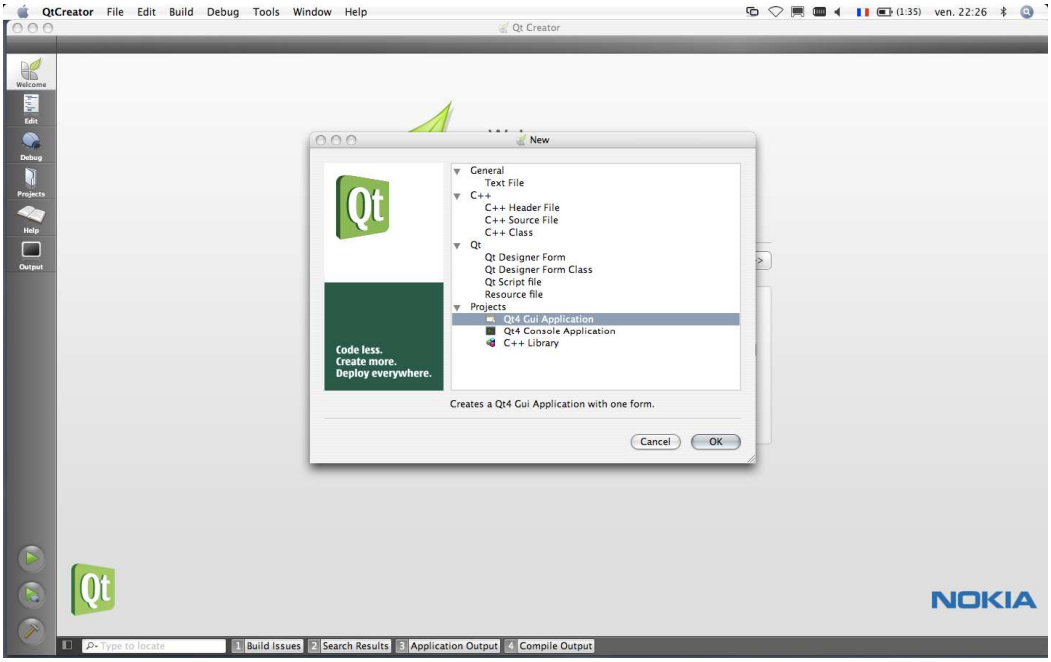
Ce document n'est pas un cours sur Qt ni un cours sur QtCreator.
C'est une introduction rapide permettant à un programmeur C++ n'ayant jamais manipulé Qt de faire en quelques minutes un programme avec une interface graphique.

Système: Mac OS X - 10.4.11

Utilisation de QT

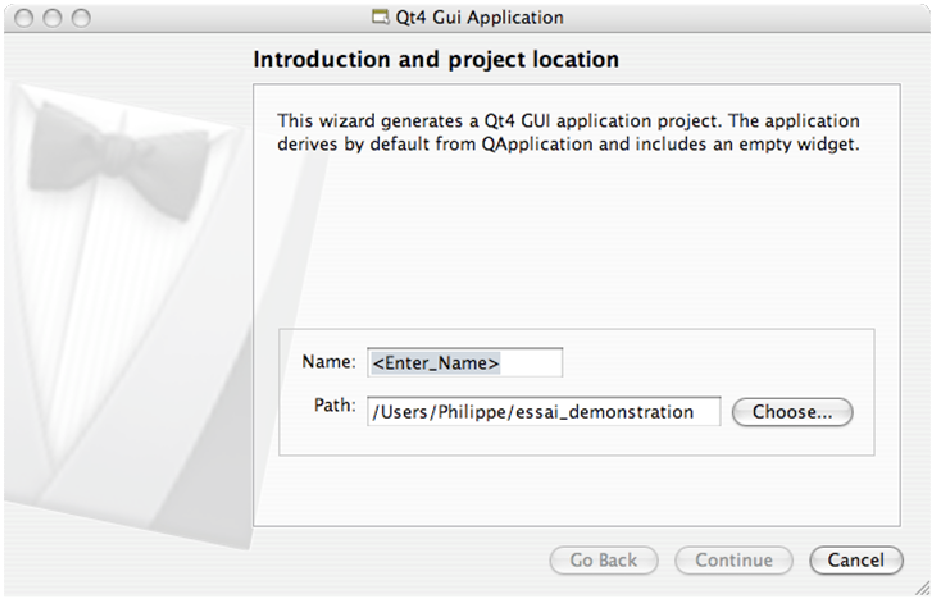
1) Création d'un projet

En utilisant l'icône QtCreator, lancez l'application et créez un projet de type GUI.

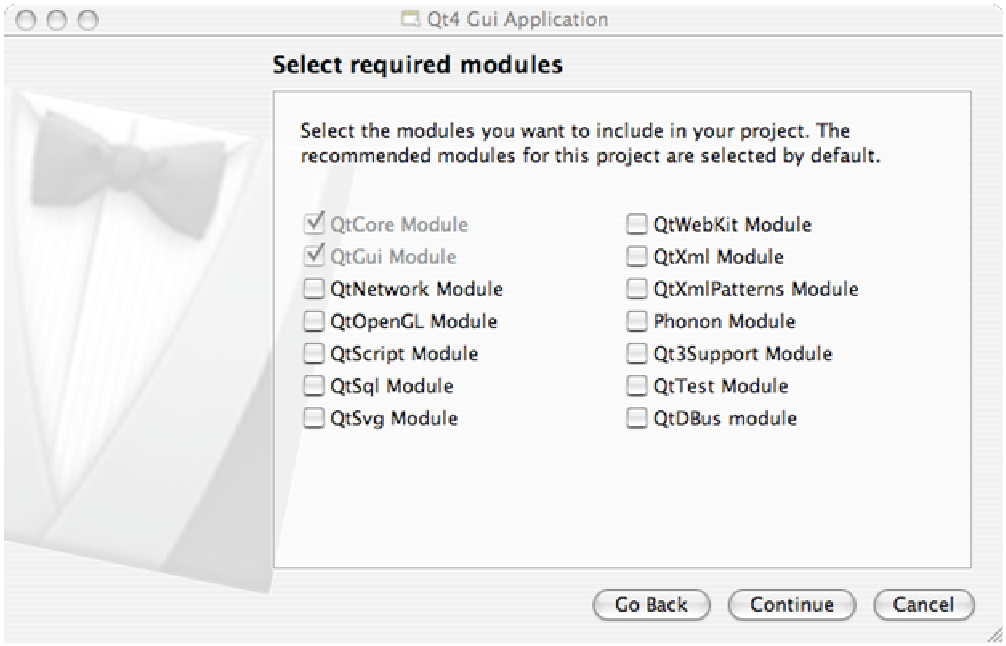


Une succession d'écrans vous permettant alors de créer un projet.

Dans un premier temps, il suffit de choisir un nom de projet et de choisir un répertoire de travail. Attention, ne mettez qu'un seul projet par répertoire car un projet se compose d'un ensemble de fichiers et il est important d'éviter les conflits.



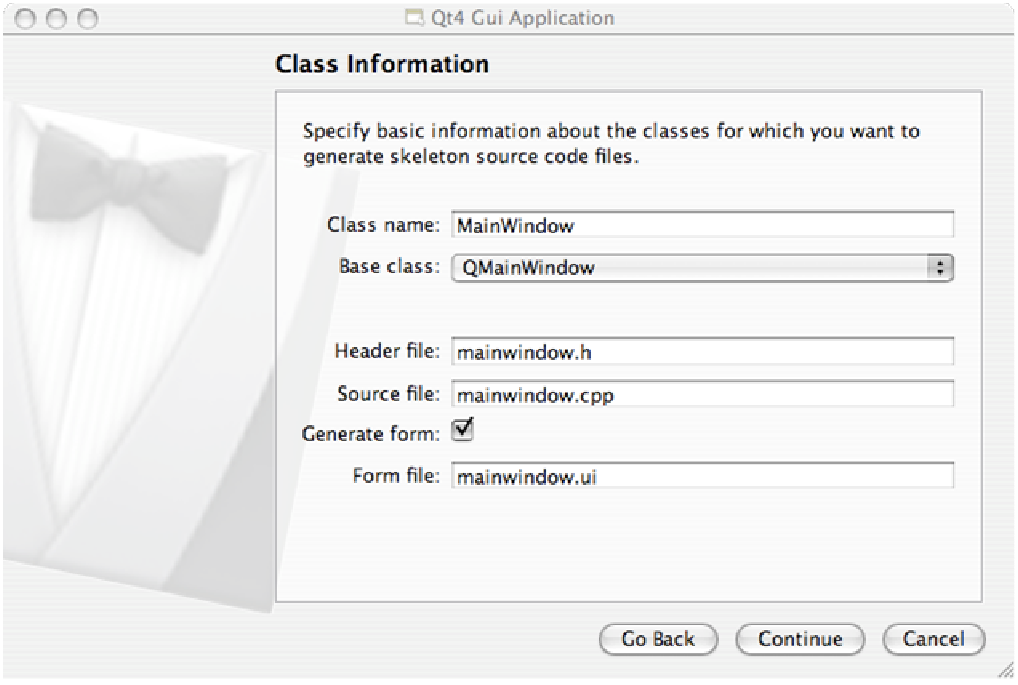
Il est important ensuite de choisir les modèles Qt à inclure dans votre projet. Par défaut seul les modules minimaux sont inclus et en particulier le module permettant de gérer les interfaces graphiques.



Le dernier écran vous permet de modifier d'une part le nom de la classe principale et par conséquent les noms des fichiers C++ correspondant. Par défaut :

- mainwindow.cpp
- mainwindow.h

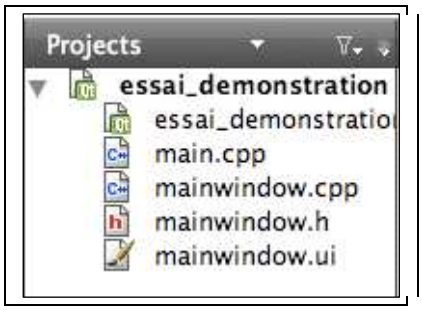
Notons la présence d'un fichier **mainwindow.ui** qui correspond au fichier de l'interface graphique c'est-à-dire permettant au générateur d'interface de fonctionner.s



2) Structure d'un projet

Le projet se compose de 5 fichiers :

- les 2 fichiers C++ dont les noms correspondent au projet
- un fichier main.cpp qui correspond au programme principal.
- Le fichier mainwindow.ui qui correspond à l'interface graphique.
- Le fichier projet Qt nommé ici **essai_demonstration**.

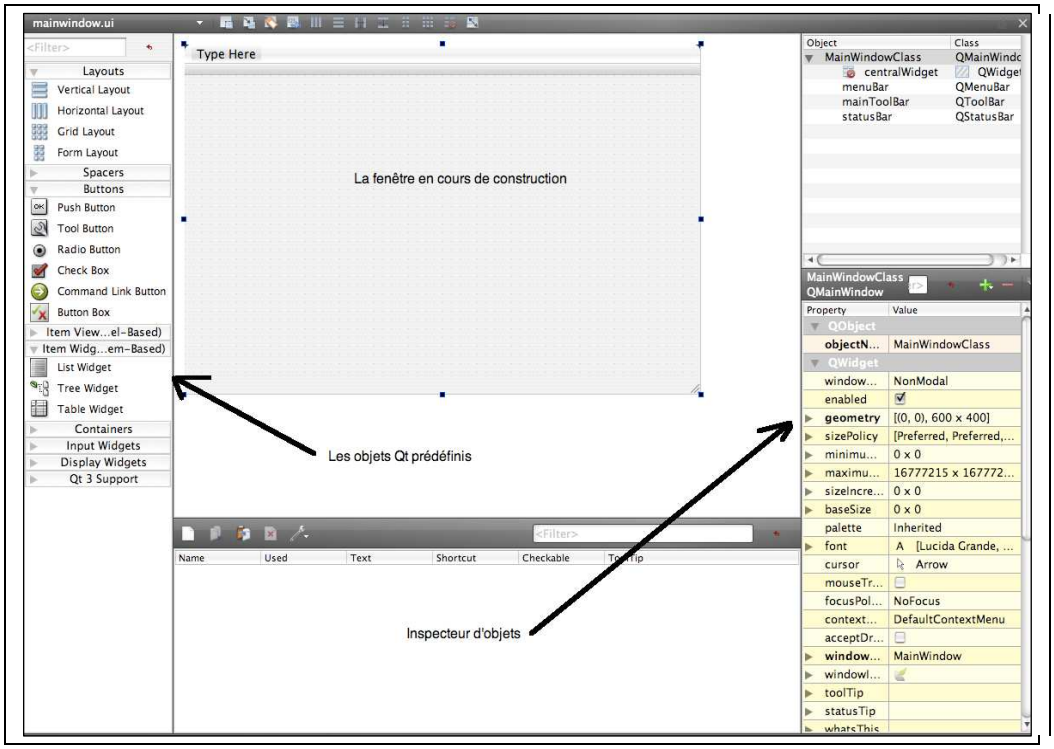


Le fichier main.cpp
On n'a vraiment besoin de le modifier que si on souhaite modifier la fenêtre de démarrage ou bien inclure de nouvelles librairies Qt.

Les fichiers mainwindow.cpp et mainwindow.h
Ces fichiers devront être souvent modifiés pendant la conception de l'interface graphique pour ajouter de nouveau slot ou signaux et ainsi faire communiquer les objets de l'interface.

3) Le générateur d'interface

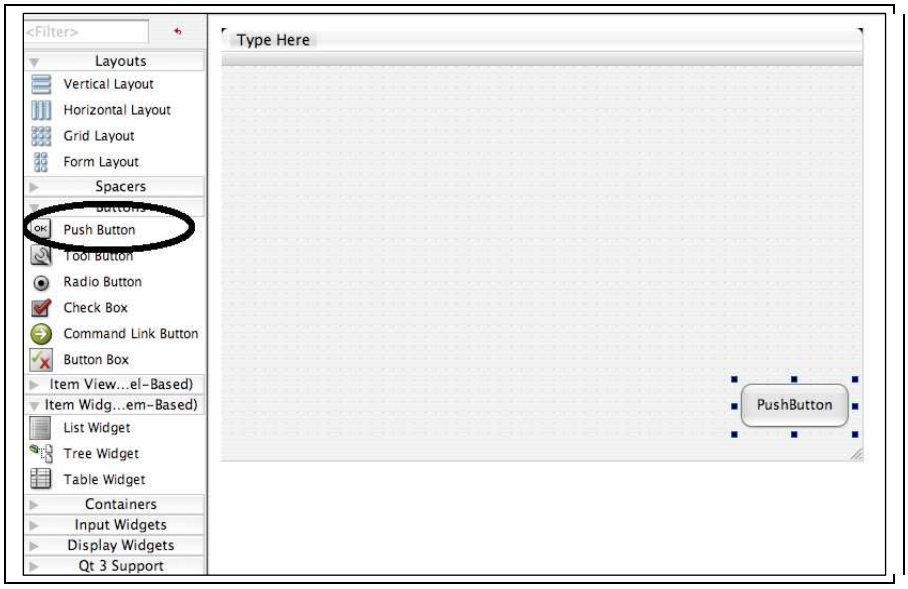
Un double click sur **mainwindow.gui** lance automatique le générateur d'interface.



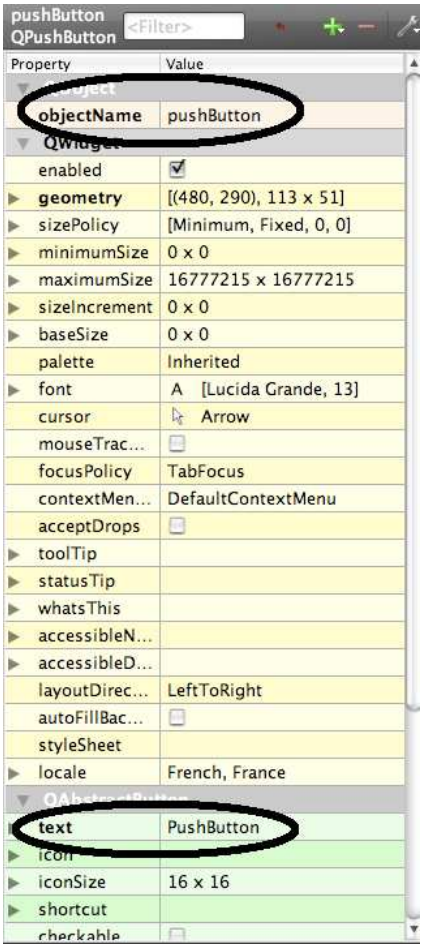
3.1. Création d'un bouton « Quitter »

Etape 1. Création d'un bouton dans la fenêtre.

Dans la section Buttons, par click à la souris, insérer un bouton sur la fenêtre, par exemple à bas à droite.



L'inspecteur d'objet affiche alors les caractéristiques de l'objet bouton qui vient d'être crée.



Deux propriétés sont importantes :

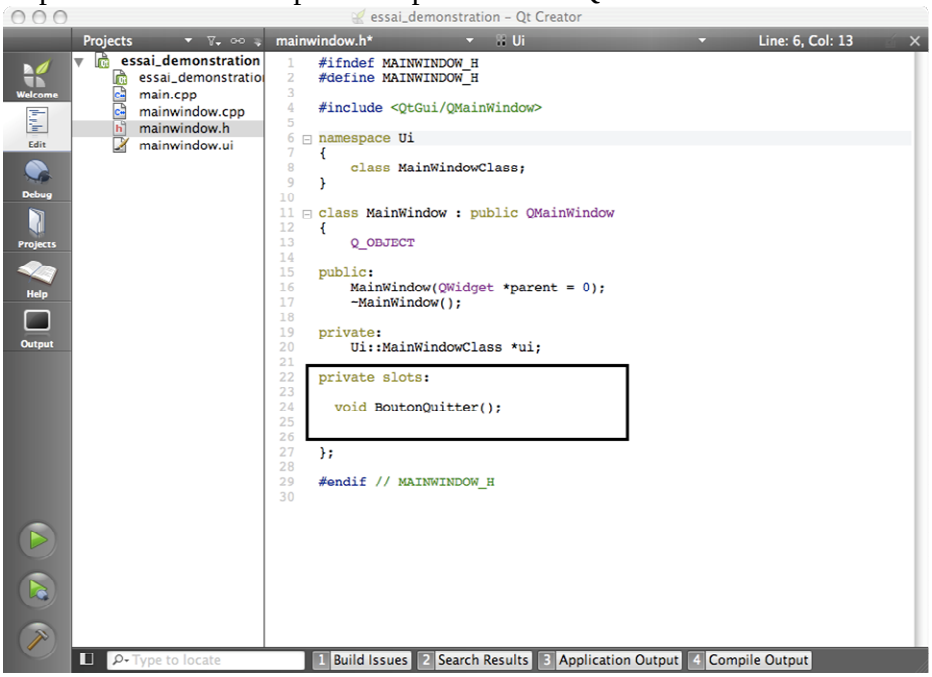
La propriété objectName correspond au nom C++ de l'instance bouton qui vient d'être crée. Cette propriété est dans la section QObject.

La propriété Text dans la section QAbstractButton qui correspond au texte affiché sur le bouton.

Modifions cette propriété en « Quitter ». Le bouton ressemble alors à ce qui suit :

Etape 2. Attacher du code sur l'événement click sur le bouton.

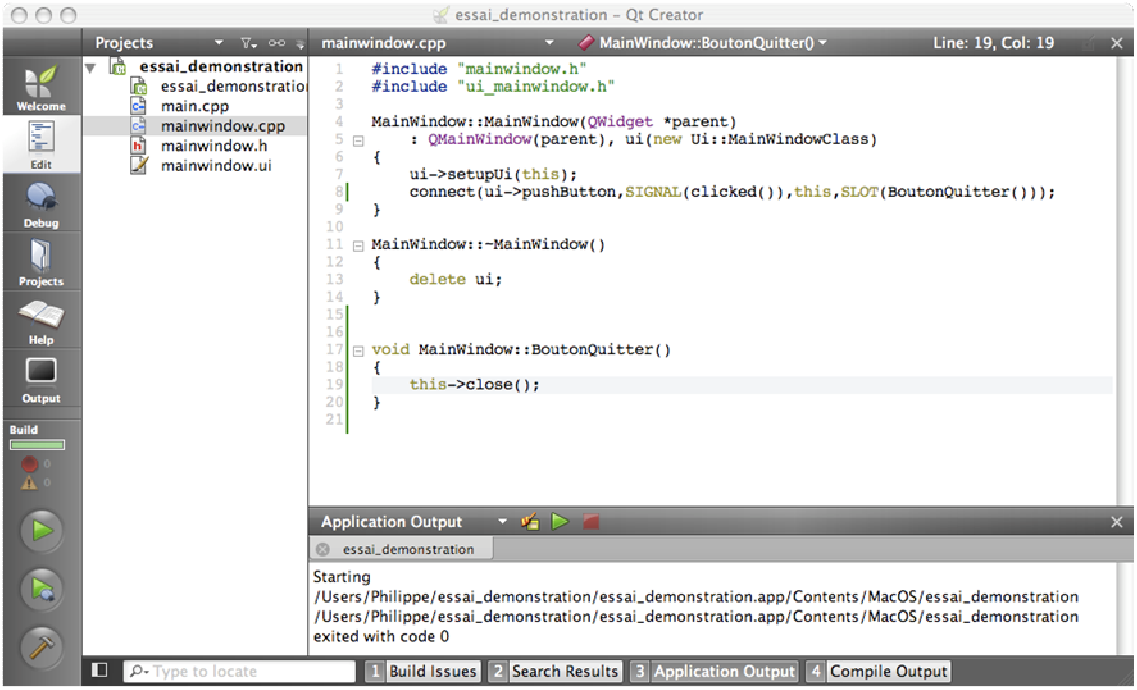
Pour cela dans le fichier **mainwindow.h** créer une nouvelle section nommée « Private Slots » et définir une procédure nommée par exemple « BoutonQuitter ».



Editer ensuite le fichier mainwindow.cpp.

Il faut inclure dans ce fichier :

- Le corps de la méthode BoutonQuitter
- Une connexion entre le signal ou événement « click » et la méthode BoutonQuitter ;



La méthode « BoutonQuitter » contient un simple appel à la méthode « close » de la fenêtre.

```
void MainWindow::BoutonQuitter()
{
this->close();
}
```

Le lien entre l'événement « Click » et la méthode se fait par la méthode Connect :

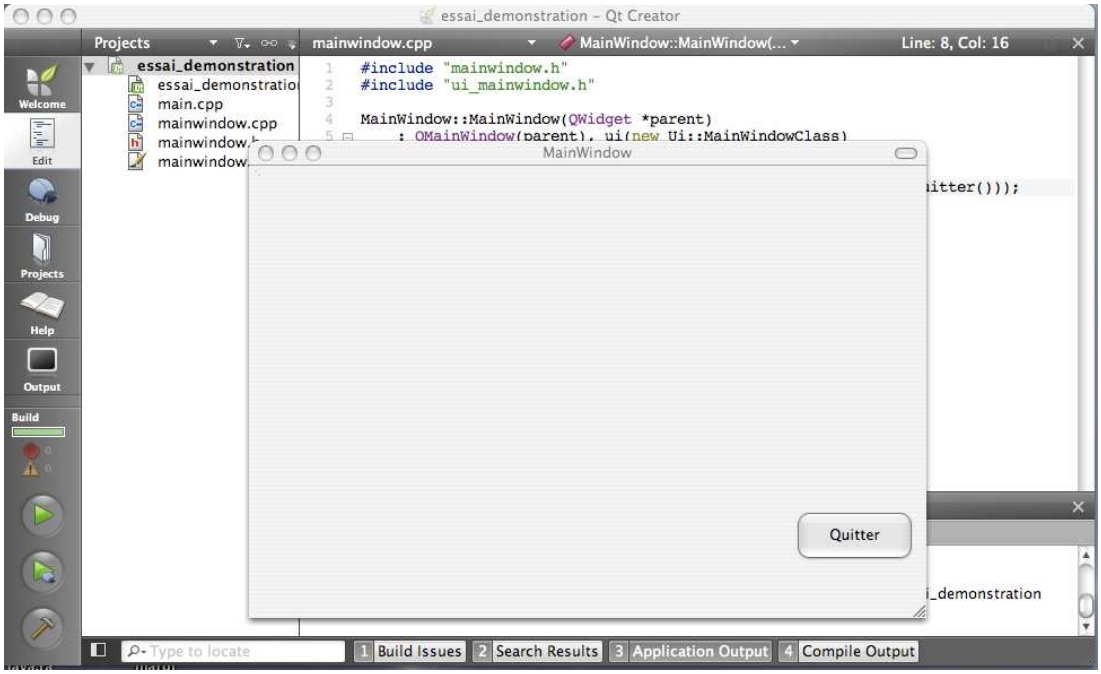
```
connect(ui->pushButton,SIGNAL(clicked()),this,SLOT(BoutonQuitter()));
```

Celle-ci ce compose de 3 parties principales :

- Le premier paramètre (ui->pushButton) fait référence à l'object graphique ;
- Le deuxième SIGNAL(clicked()) fait référence à l'événement concerné (ici « click ») ;
- Le quatrième crée la connexion entre l'objet graphique et la procédure (en Qt le lien s'appelle un slot).

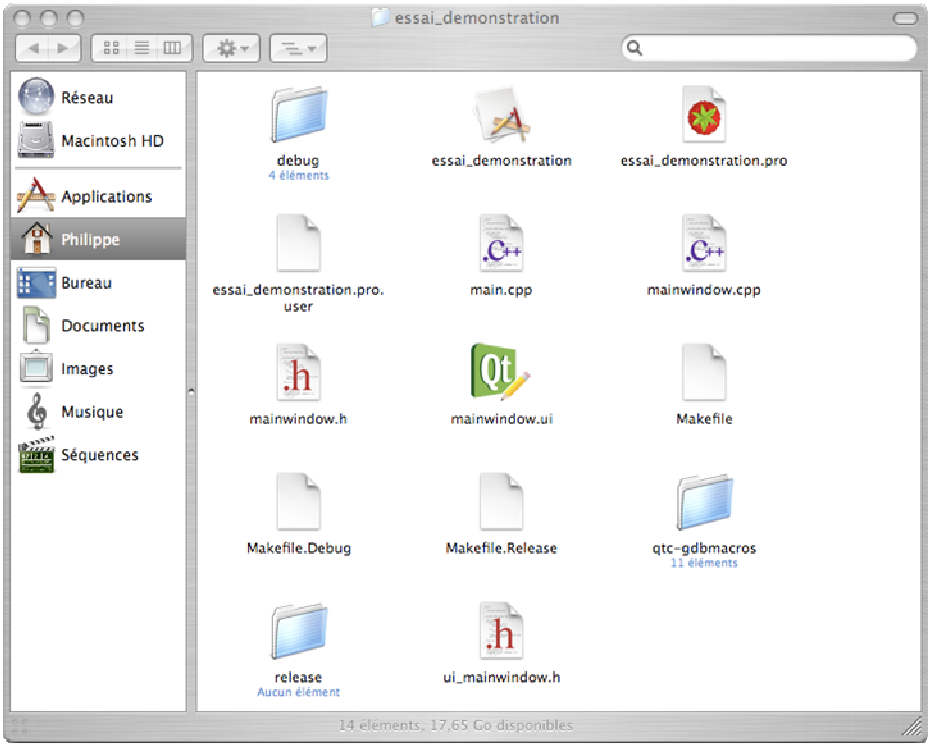
Etape 3. Vérification.

Après compilation et en lançant directement le code à partir de QtCreator on obtient le résultat suivant avec la fermeture de la fenêtre sur l'événement click du bouton.



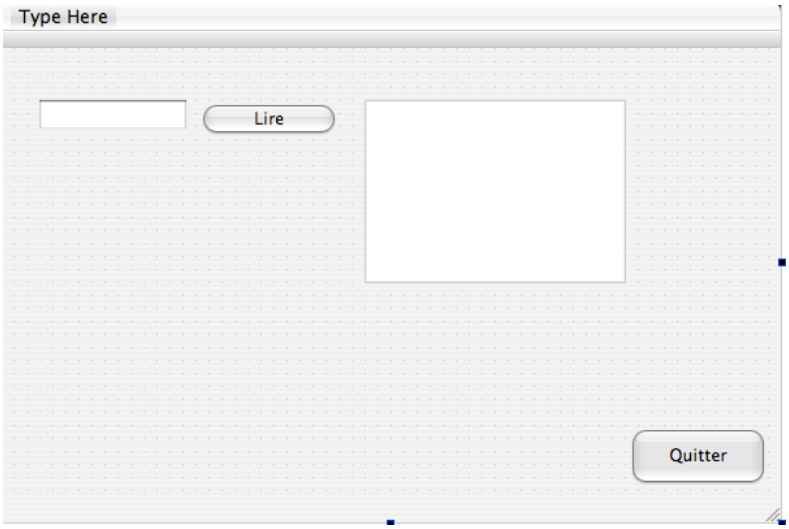
En examinant le dossier de travail, on retrouve les différents fichier du projet ainsi qu'un fichier nommé « essai_demonstration.app » sous MacIntosh ou « essai_demonstration.exe » sous Windows.

Le double click sur le fichier lance automatique l'application. Attention sous windows, il est nécessaire que votre variable d'environnement PATH contienne un lien vers les fichiers .dll de Qt. Si ce n'est pas le cas, penser à cliquer sur le poste de travail et à modifier vos variables d'environnement.



3.2. Afficher des messages sur la fenêtre.

Créez une fenêtre avec un « Lineedit » et un « textEdit » séparé par un bouton intitulé « Lire » comme indiqué sur la copie d'écran ci-dessous.



Ces deux éléments se trouvent dans la partie :

Comme précédemment, déclarer une nouvelle procedure dans le fichier mainwindow.h :

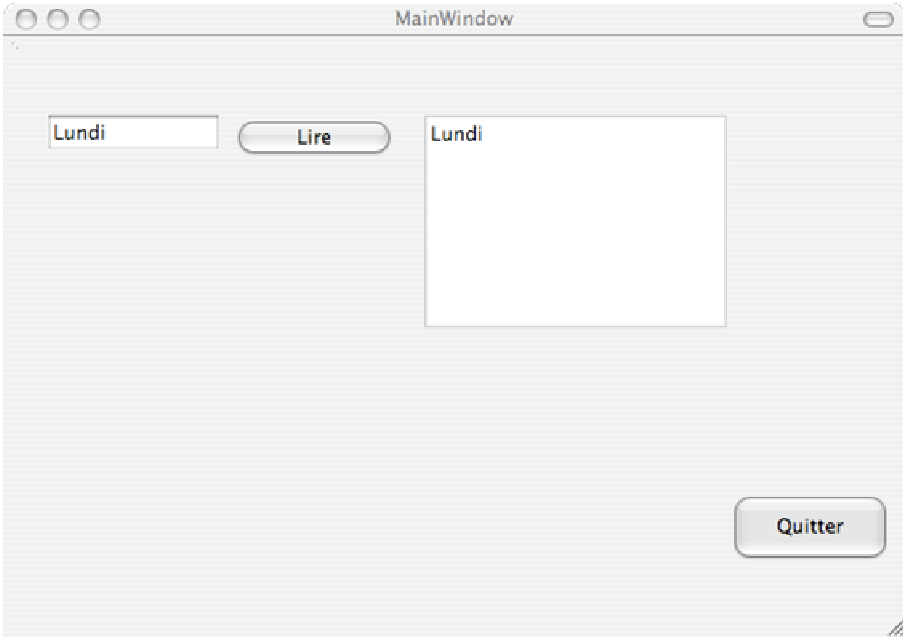
```
private slots:

  void BoutonQuitter();
  void BoutonLire();
```

Le fichier mainwindow.cpp doit être modifié comme suit :

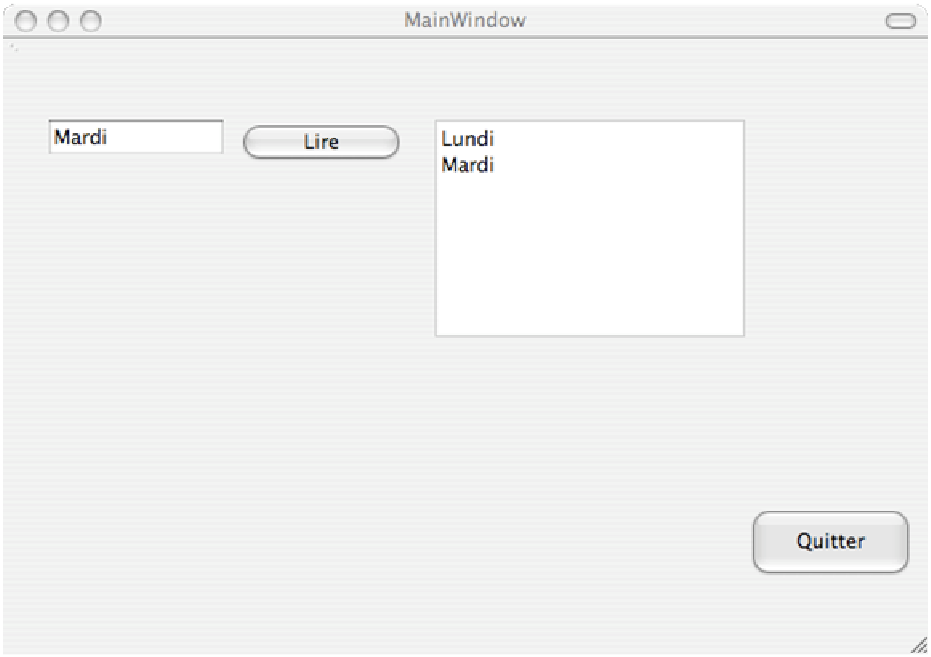
```cpp
#include "mainwindow.h"
#include "ui_mainwindow.h"
#include <qstring.h>

MainWindow::MainWindow(QWidget *parent)
    : QMainWindow(parent), ui(new Ui::MainWindowClass)
{
    ui->setupUi(this);
    connect(ui->pushButton,SIGNAL(clicked()),this,SLOT(BoutonQuitter()));
    connect(ui->pushButton_2,SIGNAL(clicked()),this,SLOT(BoutonLire()));
}

MainWindow::~MainWindow()
{
    delete ui;
}


void MainWindow::BoutonQuitter()
{
    this->close();
}

void MainWindow::BoutonLire()
{
   QString chaine;
   chaine = ui->lineEdit->text();
   ui->textEdit->append(chaine);
}

```

En venant du C++ standard, il faut prendre soin d'utiliser les types spéciaux Qt dont en particulier le type QString et non pas le type string classique.

Le résultat à l'exécution est le suivant :



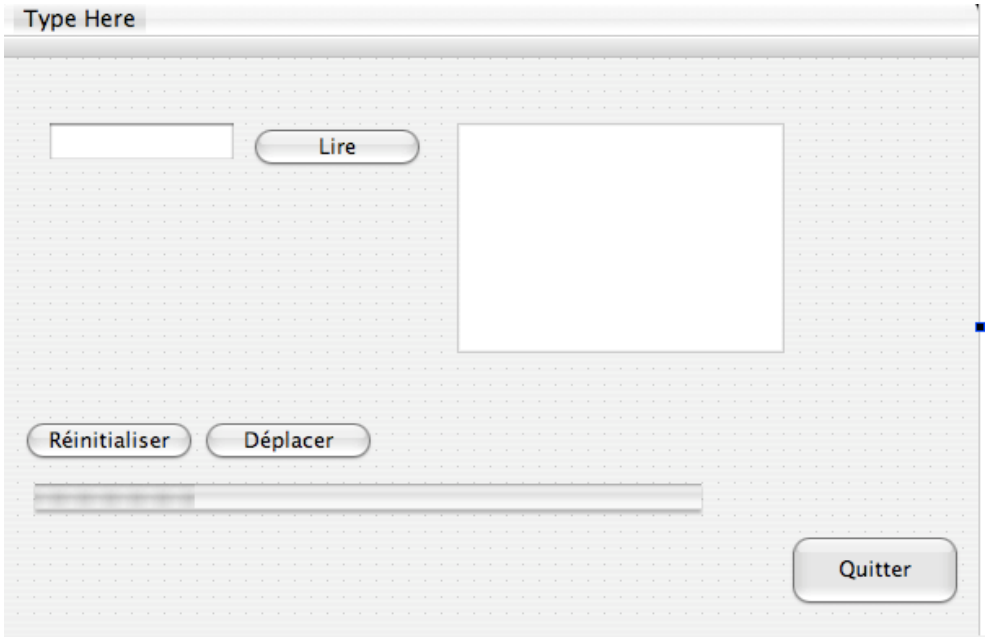
On peut constater que le texte du LineEdit est bien ajouté au fur et à mesure dans la zone de type TextEdit. Nous avons sur cet exemple mis en évidence la différence entre une zone de type LineEdit et une zone de type TextEdit.



3.3. Utilisation des barres de progression.

Ajoutez sur la fenêtre une barre de progression qui se trouve dans la section « Display Widgets ».

Ajouter 3 boutons afin d'obtenir une fenêtre comme celle-ci :



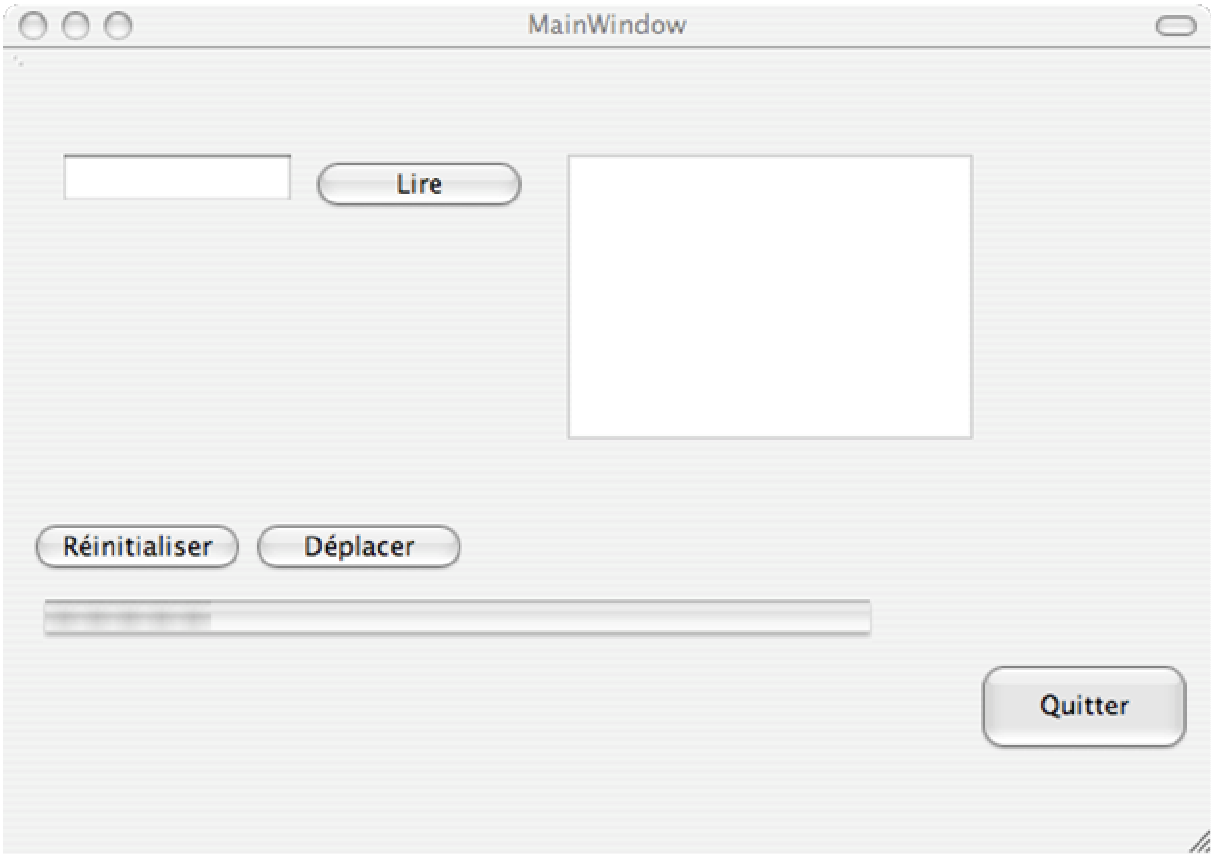
Comme précédemment, modifiez le fichier mainwindow.h :

```
void BoutonQuitter();
void BoutonLire();
void BoutonReinitialiser();
void BoutonAugmenter();
```

Le code du fichier mainwindow.cpp est modifié comme suit :

```
#include "mainwindow.h"
#include "ui_mainwindow.h"
#include <qstring.h>

MainWindow::MainWindow(QWidget *parent)
    : QMainWindow(parent), ui(new Ui::MainWindowClass)
{
    ui->setupUi(this);
    connect(ui->pushButton,SIGNAL(clicked()),this,SLOT(BoutonQuitter()));
    connect(ui->pushButton_2,SIGNAL(clicked()),this,SLOT(BoutonLire()));

    connect(ui->pushButton_3,SIGNAL(clicked()),this,SLOT(BoutonReinitialiser()));
    connect(ui->pushButton_4,SIGNAL(clicked()),this,SLOT(BoutonAugmenter()));

}

MainWindow::~MainWindow()
{
    delete ui;
}


void BoutonReinitialiser()
   {
      ui->progressBar->setValue(0);
   }

void BoutonAugmenter()
   {
    int i = ui->progressBar->value();
    i=i+10;
    ui->progressBar->setValue(i);
   }

```

Ceci qui donne finallement :



4) Conclusion

Voilà une rapide introduction à QtCreator terminée. Consultez le site de : http://www.qtfr.org/ pour obtenir une aide complète sur les librairies Qt.