Modélisez ses fenêtres avec Qt Designer

À force d'écrire le code de vos fenêtres, vous devez peut-être commencer à trouver cela long et répétitif. C'est amusant au début mais, au bout d'un moment on en a un peu marre d'écrire des constructeurs de 3 kilomètres de long juste pour placer les widgets sur la fenêtre.

C'est là que Qt Designer vient vous sauver la vie. Il s'agit d'un programme livré avec Qt (vous l'avez donc déjà installé) qui permet de concevoir vos fenêtres visuellement. Mais plus encore, Qt Designer vous permet aussi de modifier les propriétés des widgets, d'utiliser des layouts et d'effectuer la connexion entre signaux et slots.

Nous commencerons par apprendre à manipuler Qt Designer lui-même. Vous verrez que c'est un outil complexe mais qu'on s'y fait vite car il est assez intuitif.
Ensuite, nous apprendrons à utiliser les fenêtres générées avec Qt Designer dans notre code source.

C'est parti !

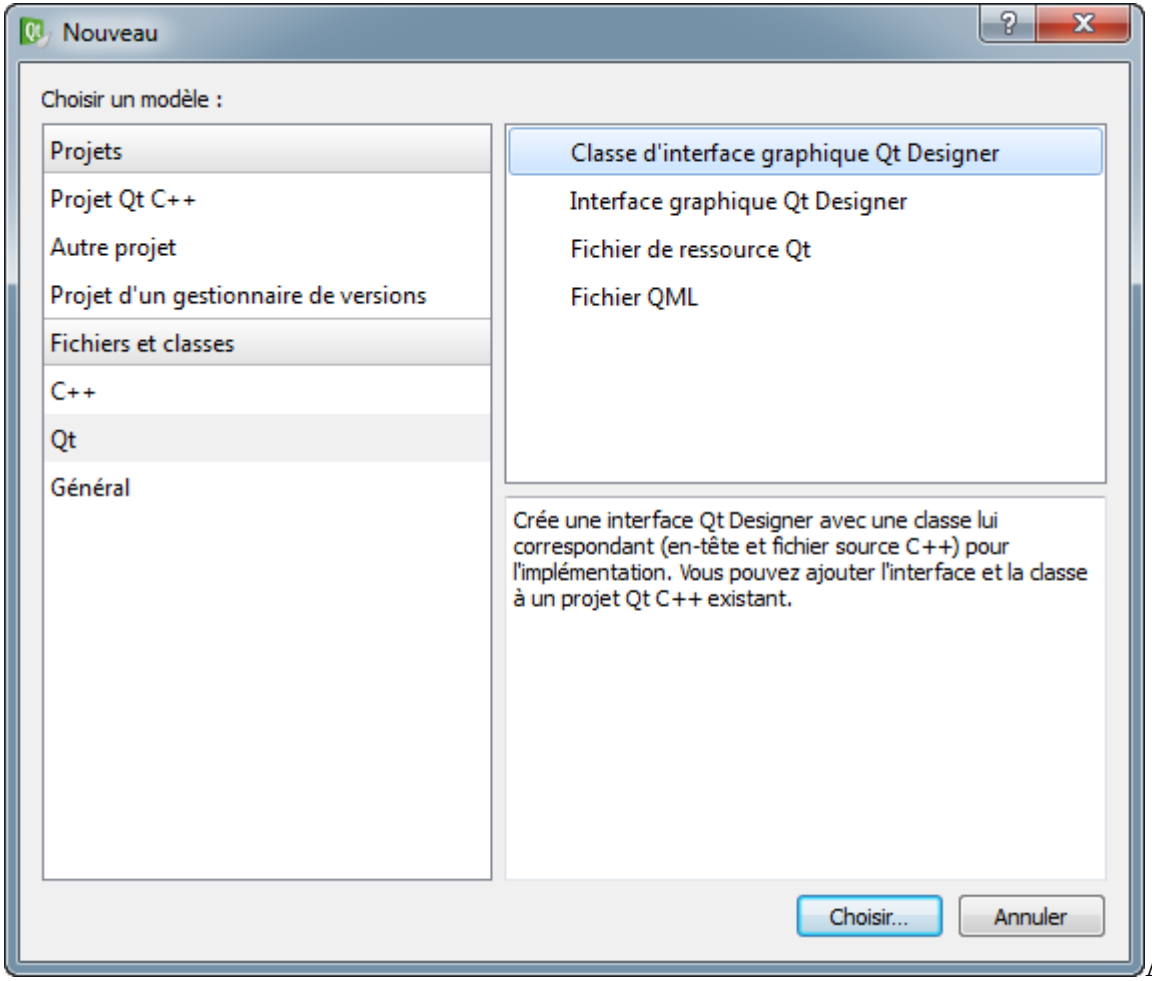
Qt Designer n'est pas un programme magique qui réfléchit à votre place. Il vous permet simplement de gagner du temps et d'éviter les tâches répétitives d'écriture du code de génération de la fenêtre.
N'utilisez PAS Qt Designer et ne lisez PAS ce chapitre si vous ne savez pas coder vos fenêtres à la main.

Présentation de Qt Designer

Qt Designer existe sous forme de programme indépendant mais il est aussi intégré au sein de Qt Creator, dans la section `Design`. Il est plus simple de travailler directement à l'intérieur de Qt Creator et cela ne change strictement rien aux possibilités qui vous sont offertes. En effet, Qt Designer est complètement intégré *dans* Qt Creator !

Comme c'est le plus simple et que cette solution n'a que des avantages, nous allons donc travailler directement dans Qt Creator.

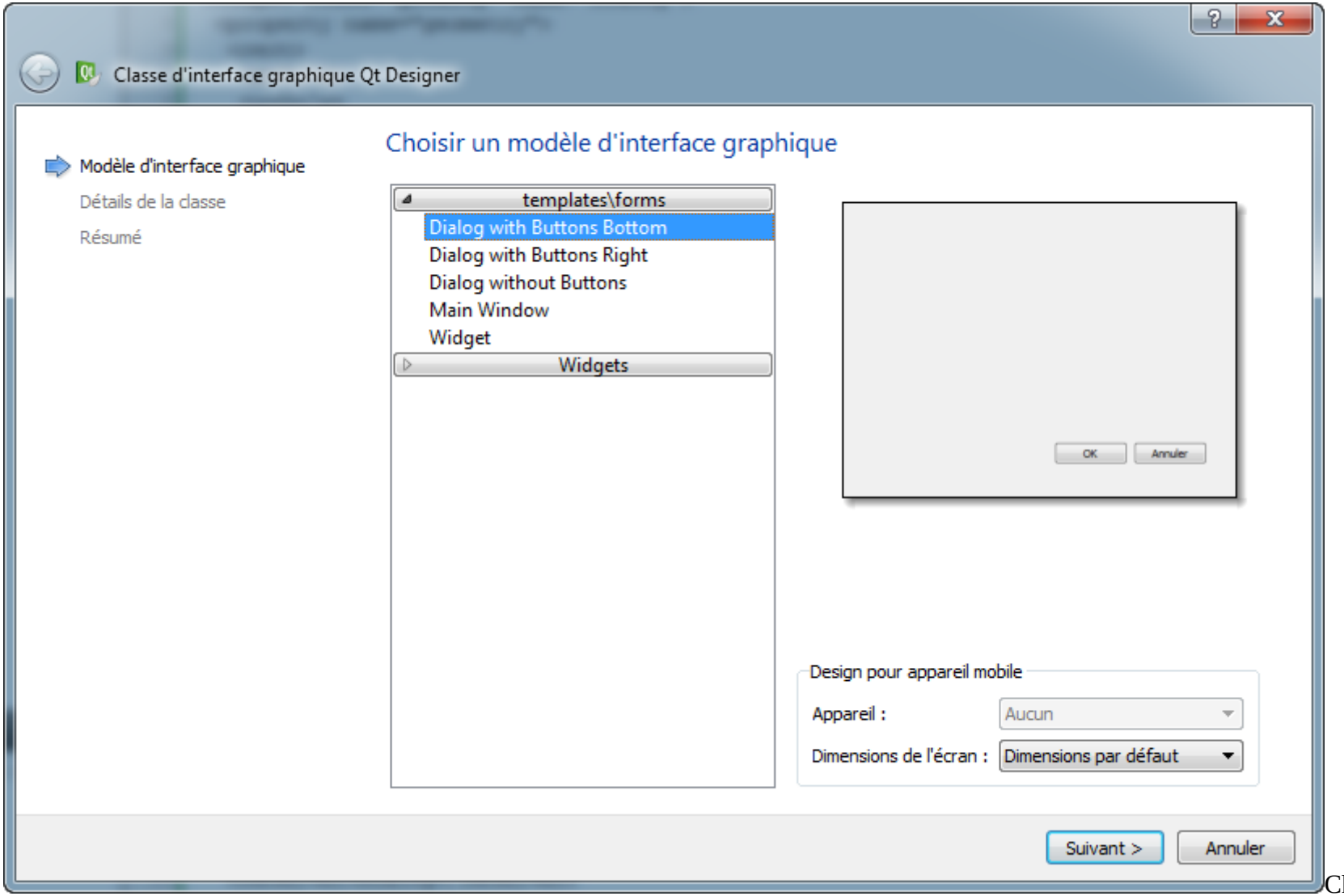
Je vais supposer que vous avez déjà créé un projet dans Qt Creator. Pour ajouter une fenêtre de Qt Designer, allez dans le menu `Fichier > Nouveau fichier ou projet` puis sélectionnez `Qt > Classe d'interface graphique Qt Designer`



Ajout d'une fenêtre

Choix du type de fenêtre à créer

Lorsque vous demandez à créer une fenêtre, on vous propose de choisir le type de fenêtre :

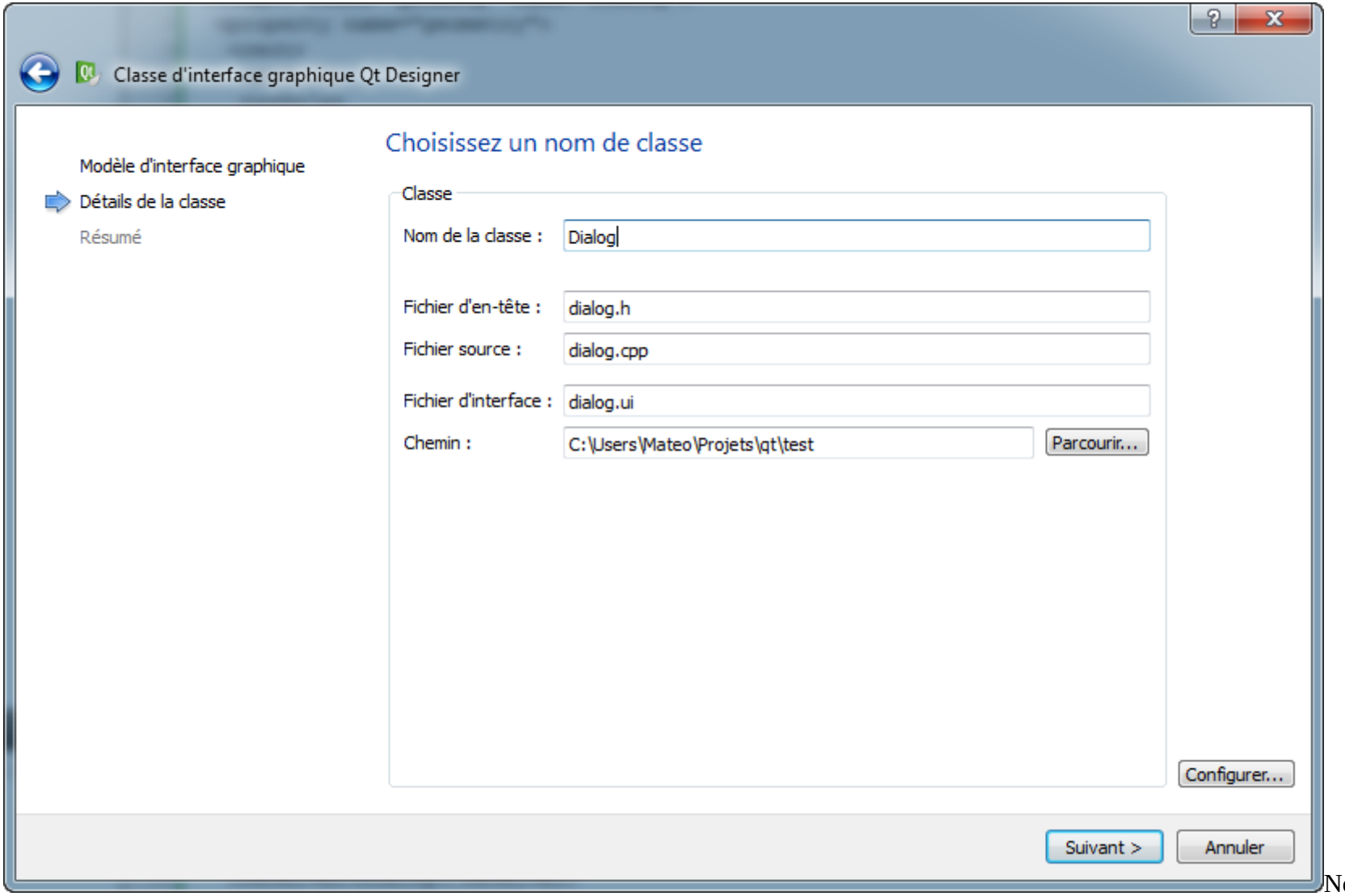


Choix du type de fenêtre

Les 3 premiers choix correspondent à des `QDialog`.
Vous pouvez aussi créer une `QMainWindow`si vous avez besoin de gérer des menus et des barres d'outils.
Enfin, le dernier choix correspond à une simple fenêtre de type `QWidget`.

Pour tester Qt Designer, peu importe le choix que vous ferez ici. On peut partir sur une `QDialog`si vous voulez (premier choix par exemple).

Dans la fenêtre suivante, on vous demande le nom des fichiers à créer (figure suivante). Pour le moment vous pouvez laisser les valeurs par défaut.



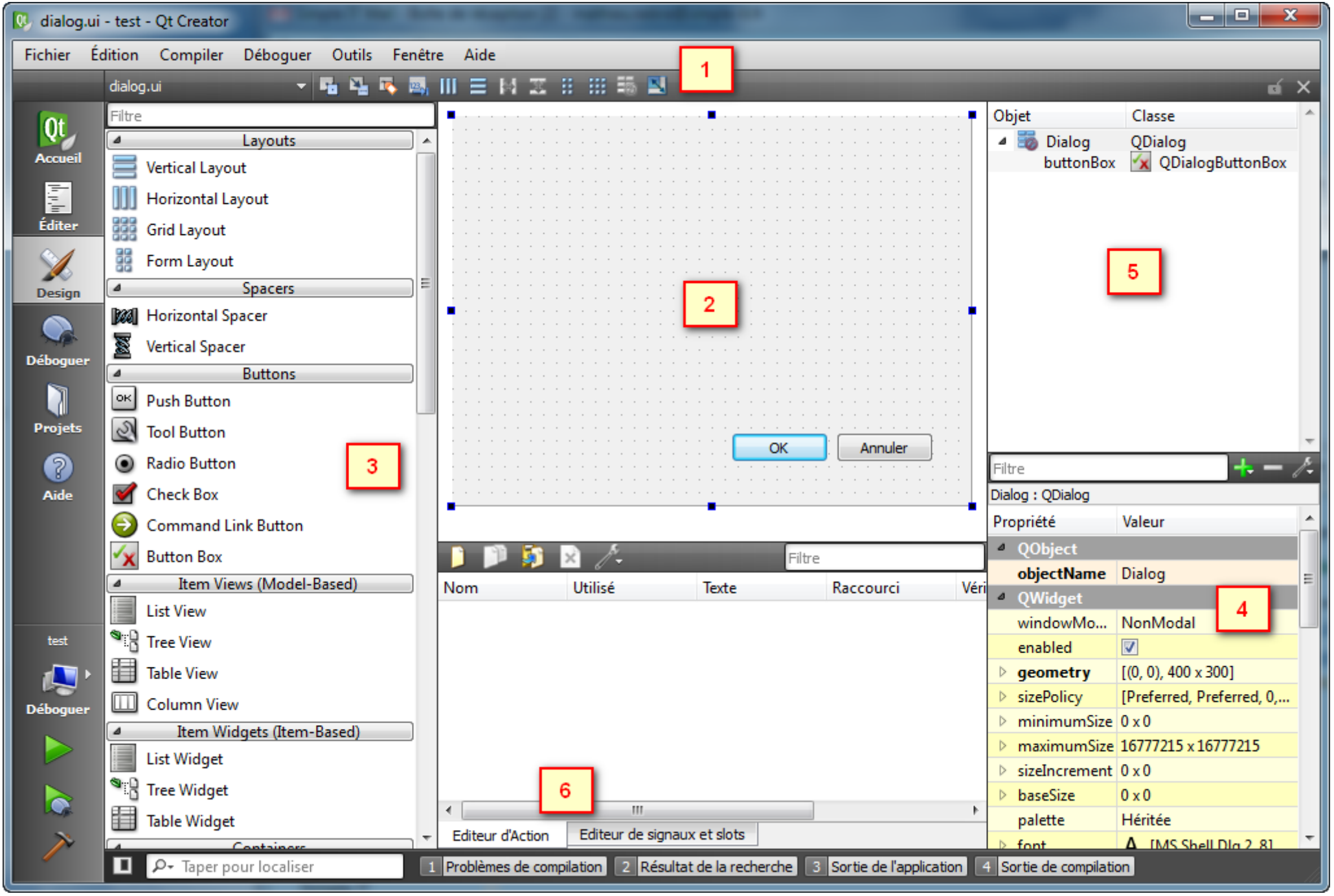
Noms des fichiers

Trois fichiers seront créés :

- `dialog.ui`: c'est le fichier qui contiendra l'interface graphique (de type XML). C'est ce fichier que nous modifierons avec l'éditeur Qt Designer ;
- `dialog.h`: permet de charger le fichier `.ui` dans votre projet C++ (en-tête de classe) ;
- `dialog.cpp`: permet de charger le fichier `.ui` dans votre projet C++ (code source de classe).

Analyse de la fenêtre de Qt Designer

Lorsque vous avez créé votre fenêtre, Qt Designer s'ouvre au sein de Qt Creator (voir figure suivante).


dialog.ui - test - Qt Creator
Fichier
Édition
Compiler
Déboguer
Outils
Fenêtre
Aide
1
dialog.ui
Accueil
Éditer
Design
Déboguer
Projets
Aide
test
Déboguer
Filtre
Layouts
Vertical Layout
Horizontal Layout
Grid Layout
Form Layout
Spacers
Horizontal Spacer
Vertical Spacer
Buttons
Push Button
Tool Button
Radio Button
Check Box
Command Link Button
Button Box
Item Views (Model-Based)
List View
Tree View
Table View
Column View
Item Widgets (Item-Based)
List Widget
Tree Widget
Table Widget
3
2
OK
Annuler
Objet
Classe
Dialog
QDialog
buttonBox
QDialogButtonBox
5
Filtre
Dialog : QDialog
Propriété
Valeur
QObject
objectName
Dialog
QWidget
windowMo...
NonModal
enabled
geometry
[(0, 0), 400 x 300]
sizePolicy
[Preferred, Preferred, 0,...
minimumSize
0 x 0
maximumSize
16777215 x 16777215
sizeIncrement
0 x 0
baseSize
0 x 0
palette
Héritée
4
Filtre
Nom
Utilisé
Texte
Raccourci
6
Editeur d'Action
Editeur de signaux et slots
Taper pour localiser
1 Problèmes de compilation
2 Résultat de la recherche
3 Sortie de l'application
4 Sortie de compilation

Qt Designer dans Qt Creator

Notez que, d'après le ruban de gauche, nous sommes dans la section Design de Qt Creator. Vous pouvez retrouver les fichiers de votre projet en cliquant sur`Éditer`.

Wow ! Mais comment je vais faire pour m'y retrouver avec tous ces boutons ?

En y allant méthodiquement.
Notez que la position des fenêtres peut être un peu différente chez vous, ne soyez pas surpris.
Détaillons chacune des zones importantes dans l'ordre :

1.
    1. Sur la **barre d'outils** de Qt Designer, au moins quatre boutons méritent votre attention. Ce sont les quatre boutons situés sous la marque « **(1)** » rouge que j'ai placée sur la capture d'écran. Ils permettent de passer d'un mode d'édition à un autre. Qt Designer propose quatre modes d'édition :
        - **Éditer les widgets** : le mode par défaut, que vous utiliserez le plus souvent. Il permet d'insérer des widgets sur la fenêtre et de modifier leurs propriétés.
        - **Éditer signaux/slots** : permet de créer des connexions entre les signaux et les slots de vos widgets.
        - **Éditer les copains** : permet d'associer des`QLabel`avec leurs champs respectifs. Lorsque vous faites un layout de type`QFormLayout`, ces associations sont automatiquement créées.
        - **Éditer l'ordre des onglets** : permet de modifier l'ordre de tabulation entre les champs de la fenêtre, pour ceux qui naviguent au clavier et passent d'un champ à l'autre en appuyant sur la touche `Tab`.

Nous ne verrons dans ce chapitre que les deux premiers modes (Éditer les widgets et Éditer signaux/slots). Les autres modes sont peu importants et je vous laisse les découvrir par vous-mêmes.

1.
    1. Au **centre** de Qt Designer, vous avez la fenêtre que vous êtes en train de dessiner. Pour le moment, celle-ci est vide.

Si vous créez une`QMainWindow`, vous aurez en plus une barre de menus et une barre d'outils. Leur édition se fait à la souris, c'est très intuitif. Si vous créez une `QDialog`, vous aurez probablement des boutons « OK » et « Annuler » déjà disposés.

1.
    1. **Widget Box** : ce dock vous donne la possibilité de sélectionner un widget à placer sur la fenêtre. Vous pouvez constater qu'il y a un assez large choix ! Heureusement, ceux-ci sont organisés par groupes pour y voir plus clair.

Pour placer un de ces widgets sur la fenêtre, il suffit de faire un glisser-déplacer. Simple et intuitif.

1.
    1. **Property Editor** : lorsqu'un widget est sélectionné sur la fenêtre principale, vous pouvez éditer ses propriétés. Vous noterez que les widgets possèdent en général beaucoup de propriétés, et que celles-ci sont organisées en fonction de la classe dans laquelle elles ont été définies. On peut ainsi modifier toutes les propriétés dont un widget hérite, en plus des propriétés qui lui sont propres.

Si aucun widget n'est sélectionné, ce sont les propriétés de la fenêtre que vous éditerez. Vous pourrez donc par exemple modifier son titre avec la propriété `windowTitle`, son icône avec `windowIcon`, etc.

1.
    1. **Object Inspector** : affiche la liste des widgets placés sur la fenêtre, en fonction de leur relation de parenté, sous forme d'arbre. Cela peut être pratique si vous avez une fenêtre complexe et que vous commencez à vous perdre dedans.
    2. **Éditeur de signaux/slots et éditeur d'action** : ils sont séparés par des onglets. L'éditeur de signaux/slots est utile si vous avez associé des signaux et des slots, les connexions du widget sélectionné apparaissant ici. Nous verrons tout à l'heure comment réaliser des connexions dans Qt Designer.

L'éditeur d'actions permet de créer des `QAction`. C'est donc utile lorsque vous créez une `QMainWindow`avec des menus et une barre d'outils.

Comme toutes les classes héritent de `QObject`, vous aurez toujours la propriété `objectName`. C'est le nom de l'objet qui sera créé. N'hésitez pas à le personnaliser, afin d'y voir plus clair tout à l'heure dans votre code source (sinon vous aurez par exemple des boutons appelés `pushButton`,`pushButton_2`,`pushButton_3`, ce qui n'est pas très clair).

Voilà qui devrait suffire pour une présentation générale de Qt Designer. Maintenant, pratiquons un peu.

Placez des widgets sur la fenêtre

Placer des widgets sur la fenêtre est en fait très simple : vous prenez le widget que vous voulez dans la liste à gauche et vous le faites glisser où vous voulez sur la fenêtre.

Ce qui est très important à savoir, c'est qu'on peut placer ses widgets de deux manières différentes :

- **De manière absolue** : vos widgets seront disposés au pixel près sur la fenêtre. C'est la méthode par défaut, la plus précise, mais la moins flexible aussi. Je vous avais parlé de ses défauts dans le chapitre sur les layouts.
- **Avec des layouts** (recommandé pour les fenêtres complexes) : vous pouvez utiliser tous les layouts que vous connaissez. Verticaux, horizontaux, en grille, en formulaire… Grâce à cette technique, les widgets s'adapteront automatiquement à la taille de votre fenêtre.

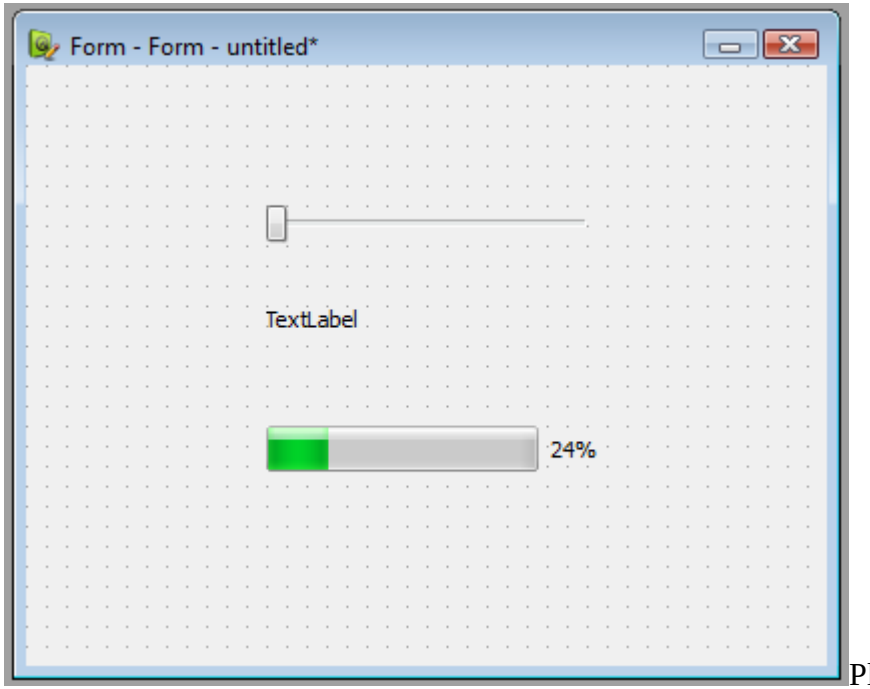
Commençons par les placer de manière absolue, puis nous verrons comment utiliser les layouts dans Qt Designer.

Placez les widgets de manière absolue

Je vous propose, pour vous entraîner, de faire une petite fenêtre simple composée de 3 widgets :

- `QSlider`;
- `QLabel`;
- `QProgressBar`.

Votre fenêtre devrait maintenant ressembler à peu près à la figure suivante.



Placement de widgets sur la fenêtre

Vous pouvez déplacer ces widgets comme bon vous semble sur la fenêtre.
Vous pouvez les agrandir ou les rétrécir.

Quelques raccourcis à connaître :

-
  - En maintenant la touche `Ctrl` appuyée, vous pouvez sélectionner plusieurs widgets en même temps.
  - Faites `Suppr` pour supprimer les widgets sélectionnés.
  - Si vous maintenez la touche `Ctrl` enfoncée lorsque vous déplacez un widget, celui-ci sera copié.
  - Vous pouvez faire un double-clic sur un widget pour modifier son nom (il vaut mieux donner un nom personnalisé plutôt que laisser le nom par défaut).

Sur certains widgets complexes, comme la`QComboBox`(liste déroulante), le double-clic vous permet d'éditer la liste des éléments contenus dans la liste déroulante.

- Pensez aussi à faire un clic droit sur les widgets pour modifier certaines propriétés, comme la bulle d'aide (`toolTip`).

Utilisez les layouts

Pour le moment, nous n'utilisons aucun layout. Si vous essayez de redimensionner la fenêtre, vous verrez que les widgets ne s'adaptent pas à la nouvelle taille et qu'ils peuvent même disparaître si on réduit trop la taille de la fenêtre !

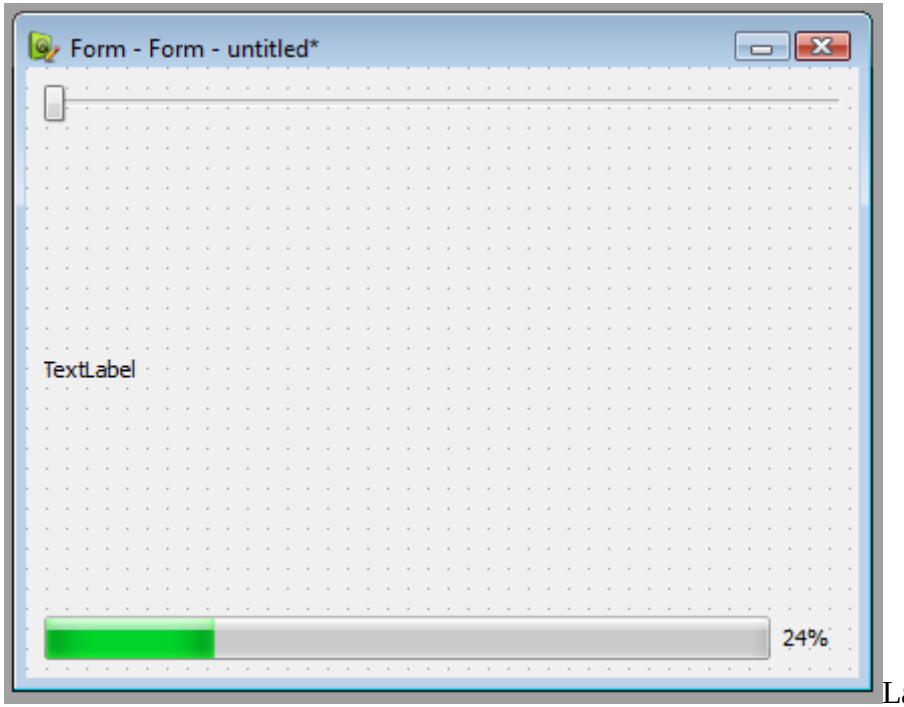
Il y a deux façons d'utiliser des layouts :

- utiliser la barre d'outils en haut ;
- glisser-déplacer des layouts depuis le dock de sélection de widgets (« Widget Box »).

Pour une fenêtre simple comme celle-là, nous n'aurons besoin que d'un layout principal.
Pour définir ce layout principal, le mieux est de passer par la barre d'outils (figure suivante).

Cliquez sur une **zone vide** de la fenêtre (en clair, c'est la fenêtre qui doit être sélectionnée et non un de ses widgets). Vous devriez alors voir les boutons de la barre d'outils des layouts s'activer.

Cliquez sur le bouton correspondant au layout vertical (le second) pour organiser automatiquement la fenêtre selon un layout vertical.
Vous devriez alors voir vos widgets s'organiser comme sur la figure suivante :



Layout vertical

C'est le layout vertical qui les place ainsi afin qu'ils occupent toute la taille de la fenêtre. Bien sûr, vous pouvez réduire la taille de la fenêtre si vous le désirez.
Vous pouvez aussi demander à ce que la fenêtre soit réduite à la taille minimale acceptable, en cliquant sur le bouton tout à droite de la barre d'outils, intitulé « Adjust Size ».

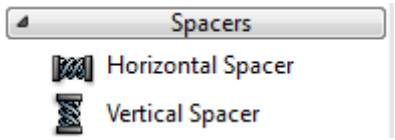
Maintenant que vous avez défini le layout principal de la fenêtre, sachez que vous pouvez insérer un sous-layout en plaçant par exemple un des layouts proposés dans la Widget Box.

Insérez des spacers

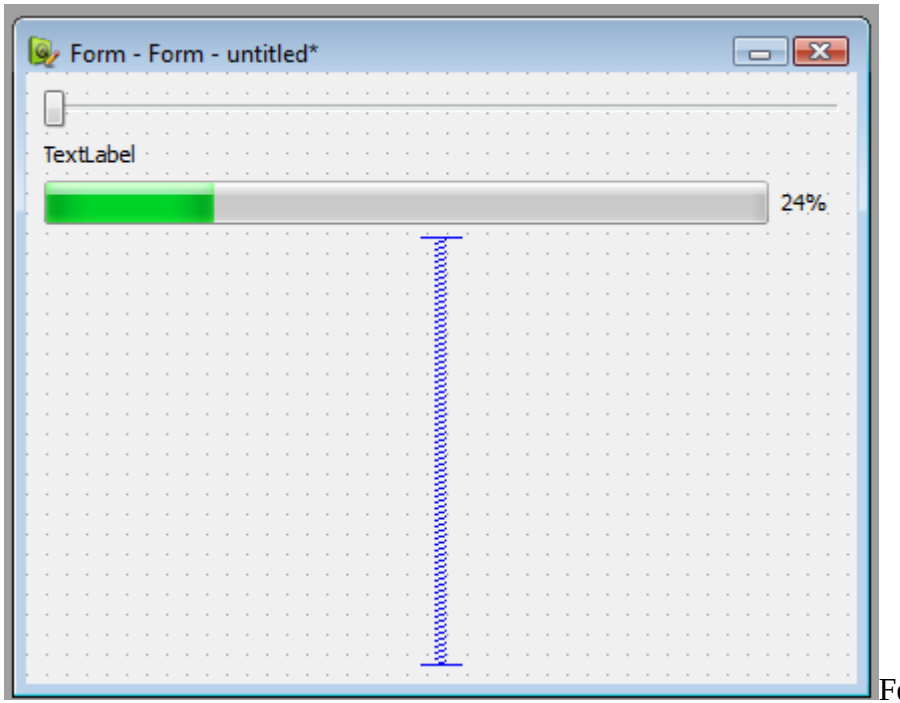
Vous trouvez que la fenêtre est un peu moche si on l'agrandit trop ?
Moi aussi. Les widgets sont trop espacés, cela ne me convient pas.

Pour changer la position des widgets tout en conservant le layout, on peut insérer un spacer. Il s'agit d'un widget invisible qui sert à créer de l'espace sur la fenêtre.

Le mieux est encore d'essayer pour comprendre ce que cela fait. Dans la Widget Box, vous devriez avoir une section « Spacers » :



Prenez un « Vertical Spacer », et insérez-le tout en bas de la fenêtre :



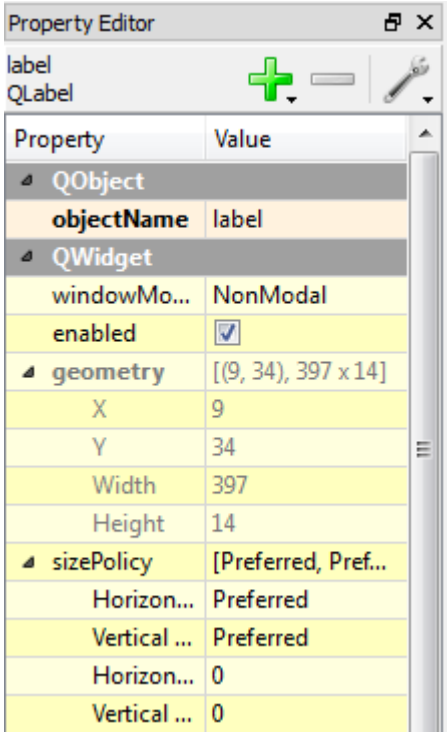
Fenêtre avec spacer

Le spacer force les autres widgets à se coller tout en haut. Ils sont toujours organisés selon un layout, mais au moins, maintenant, nos widgets sont plus rapprochés les uns des autres.
Essayez de déplacer le spacer sur la fenêtre pour voir. Placez-le entre le libellé et la barre de progression. Vous devriez voir que la barre de progression se colle maintenant tout en bas.

Le comportement du spacer est assez logique mais il faut l'essayer pour bien le comprendre.

Éditez les propriétés des widgets

Il nous reste une chose très importante à voir : l'édition des propriétés des widgets.
Sélectionnez par exemple le libellé (`QLabel`). Regardez le dock intitulé « Property Editor ». Il affiche maintenant les propriétés du`QLabel`(figure suivante).



Éditeur de propriétés

Ces propriétés sont organisées en fonction de la classe dans laquelle elles ont été définies et c'est une bonne chose. Je m'explique.
Vous savez peut-être qu'un`QLabel`hérite de`QFrame`, qui hérite de`QWidget`, qui hérite lui-même de`QObject`?

Chacune de ces classes définit des propriétés.`QLabel`hérite donc des propriétés de`QFrame`,`QWidget`et`QObject`, mais a aussi des propriétés qui lui sont propres.

Sur ma capture d'écran, on peut voir une propriété de`QObject`:`objectName`. C'est le nom de l'objet qui sera créé dans le code. Je vous conseille de le personnaliser pour vous y retrouver dans le code source par la suite.

La plupart du temps, on peut éditer le nom d'un widget en faisant un double-clic dessus, sur la fenêtre.

Si vous descendez un peu plus bas dans la liste, vous devriez vous rendre compte qu'un grand nombre de propriétés sont proposées par `QWidget` (notamment la police, le style de curseur de la souris, etc.). Descendez encore plus bas. Vous devriez arriver sur les propriétés héritées de `QFrame`, puis celles propres à `QLabel`.

Vous devriez modifier la propriété `text`, pour changer le texte affiché dans le `QLabel`. Mettez par exemple « 0 ». Amusez-vous à changer la police (propriété `font` issue de `QWidget`) ou encore à mettre une bordure (propriété `frameShape` issue de `QFrame`).

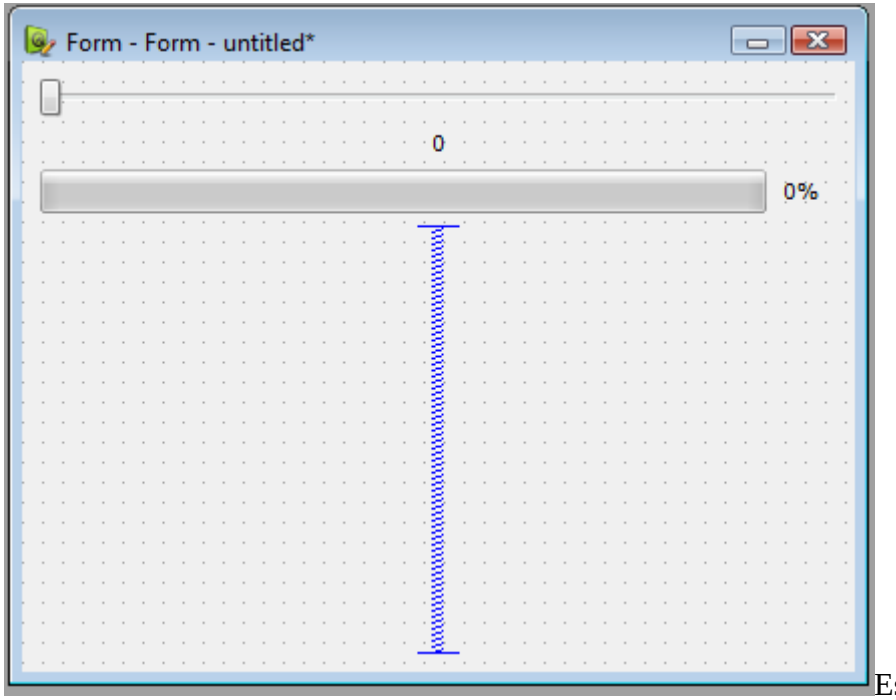
Vous remarquerez que, lorsque vous éditez une propriété, son nom s'affiche en gras pour être mis en valeur. Cela vous permet par la suite de repérer du premier coup d'œil les propriétés que vous avez modifiées.

Modifiez aussi les propriétés de la `QProgressBar` pour qu'elle affiche 0% par défaut (propriété `value`).

Vous pouvez aussi modifier les propriétés de la fenêtre. Cliquez sur une zone vide de la fenêtre afin qu'aucun widget ne soit sélectionné. Le dock « Property Editor » vous affichera alors les propriétés de la fenêtre (ici, notre fenêtre est un `QWidget`, donc vous aurez juste les propriétés de `QWidget`).

Astuce : si vous ne comprenez pas à quoi sert une propriété, cliquez dessus puis appuyez sur la touche `F1`. Qt Designer lancera automatiquement Qt Assistant pour afficher l'aide sur la propriété sélectionnée.

Essayez d'avoir une fenêtre qui, au final, ressemble *grosso modo* à la mienne :



Essayez de construire une fenêtre comme celle-ci !

Le libellé et la barre de progression doivent afficher 0 par défaut.

Bravo, vous savez maintenant insérer des widgets, les organiser selon un layout et personnaliser leurs propriétés dans Qt Designer ! Nous n'avons utilisé pour le moment que le mode « Edit Widgets ». Il nous reste à étudier le mode « Edit Signals/Slots »…

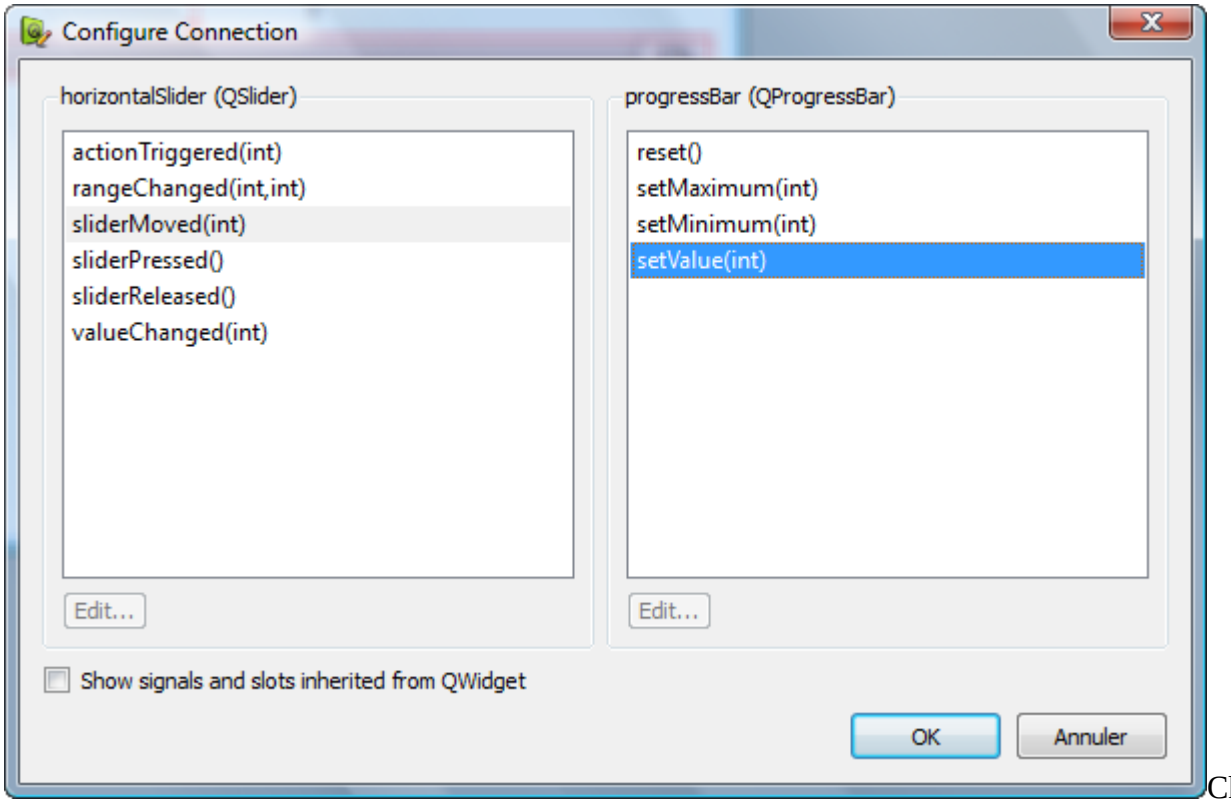
Configurez les signaux et les slots

Passez en mode « Edit Signals/Slots » en cliquant sur le second bouton de la barre d'outils (figure suivante).

Dans ce mode, on ne peut pas ajouter, modifier, supprimer, ni déplacer de widgets. Par contre, si vous pointez sur les widgets de votre fenêtre, vous devriez les voir s'encadrer de rouge.

Vous pouvez, de manière très intuitive, associer les widgets entre eux pour créer des connexions simples entre leurs signaux et slots. Je vous propose par exemple d'associer le`QSlider`avec notre`QProgressBar`.

Pour cela, cliquez sur le`QSlider`et maintenez enfoncé le bouton gauche de la souris. Pointez sur la`QProgressBar`et relâchez le bouton. Une fenêtre apparaît alors pour que vous puissiez choisir le signal et le slot à connecter :

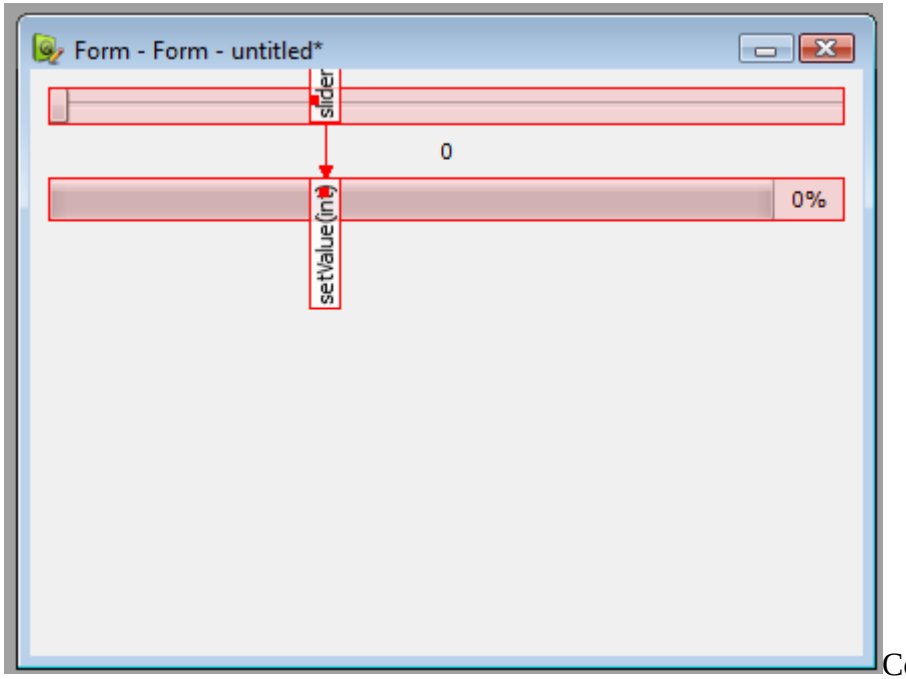


Choix des signaux et slots

- À gauche : les signaux disponibles dans le `QSlider`.
- À droite : les slots *compatibles* disponibles dans la `QProgressBar`.

Sélectionnez un signal à gauche, par exemple `sliderMoved(int)`. Ce signal est envoyé dès que l'on déplace un peu le slider.
Vous verrez que la liste des slots compatibles apparaît à droite.

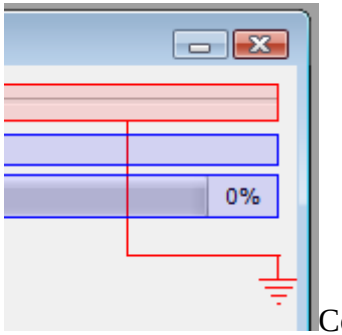
Nous allons connecter `sliderMoved(int)` du `QSlider` avec `setValue(int)` de la `QProgressBar`.
Faites OK pour valider une fois le signal et le slot choisis. C'est bon, la connexion est créée !



Connexion établie !

Faites de même pour associer `sliderMoved(int)` du QSlider à `setNum(int)` du `QLabel`.

Notez que vous pouvez aussi connecter un widget à la fenêtre. Dans ce cas, visez une zone vide de la fenêtre. La flèche devrait se transformer en symbole de masse (bien connu par ceux qui font de l'électricité ou de l'électronique).



Connexion à la masse

Cela vous permet d'associer un signal du widget à un slot de la fenêtre, ce qui peut vous être utile si vous voulez créer un bouton « Fermer la fenêtre » par exemple.

Attention : si dans la fenêtre du choix du signal et du slot vous ne voyez aucun slot s'afficher pour la fenêtre, c'est normal. Qt les masque par défaut car ils sont nombreux. Si on les affichait pour chaque connexion entre 2 widgets, on en aurait beaucoup trop (puisque tous les widgets héritent de`QWidget`).
Pour afficher quand même les signaux et slots issus de`QWidget`, cochez la case « Show signals and slots inherited from QWidget ».

Pour des connexions simples entre les signaux et les slots des widgets, Qt Designer est donc très intuitif et convient parfaitement.

Eh, mais si je veux créer un slot personnalisé pour faire des manipulations un peu plus complexes, comment je fais ?

Qt Designer ne peut pas vous aider pour cela. Si vous voulez créer un signal ou un slot personnalisé, il faudra le faire tout à l'heure dans le code source (en modifiant les fichiers`.h`et`.cpp`qui ont été créés en même temps que le`.ui`).
Comme vous pourrez le voir, néanmoins, c'est très simple à faire.

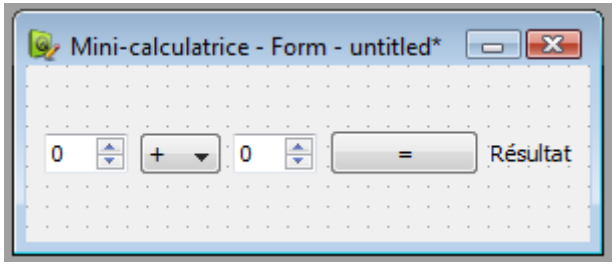
En y réfléchissant bien, c'est même la seule chose que vous aurez à coder ! En effet, tout le reste est automatiquement géré par Qt Designer. Vous n'avez plus qu'à vous concentrer sur la partie « réflexion » de votre code source.
Qt Designer vous permet donc de gagner du temps en vous épargnant les tâches répétitives et basiques qu'on fait à chaque fois que l'on crée une fenêtre.

Utilisez la fenêtre dans votre application

Il reste une dernière étape et non des moindres : apprendre à utiliser la fenêtre ainsi créée dans votre application.

Notre nouvel exemple

Je vous propose de créer une nouvelle fenêtre. On va créer une mini-calculatrice (figure suivante).



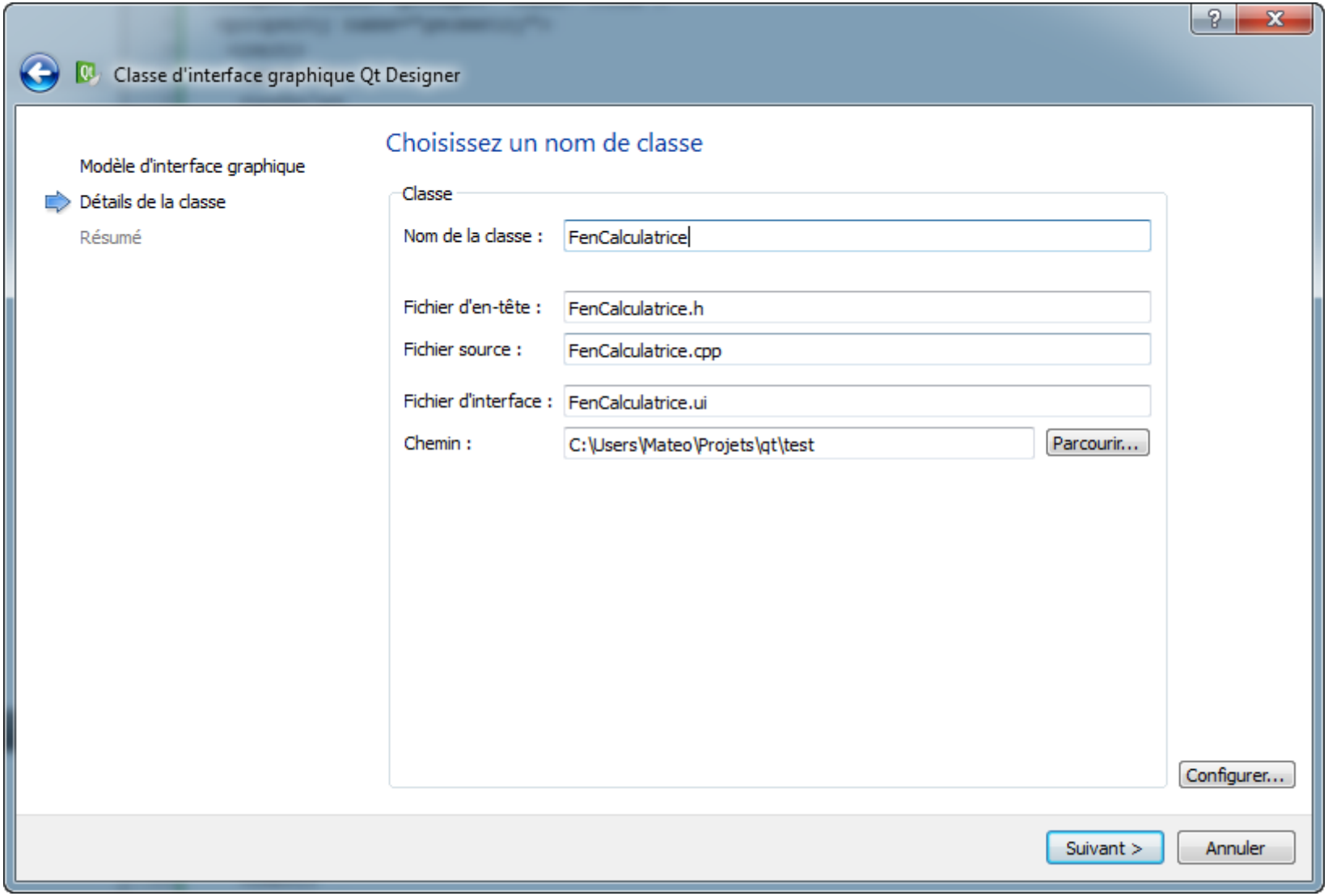
Essayez de reproduire à peu près la même fenêtre que moi, de type Widget.
Un layout principal horizontal suffira à organiser les widgets.

La fenêtre est constituée des widgets suivants, de gauche à droite :

Widget	Nom de l'objet
`QSpinBox`	`nombre1`
`QComboBox`	`Operation`
`QSpinBox`	`nombre2`
`QPushButton`	`boutonEgal`
`QLabel`	`Resultat`

Pensez à bien renommer les widgets afin que vous puissiez vous y retrouver dans votre code source par la suite.
Pour la liste déroulante du choix de l'opération, je l'ai déjà pré-remplie avec quatre valeurs : `+`, `-`, `*` et `/`.
Faites un double-clic sur la liste déroulante pour ajouter et supprimer des valeurs.

Il faudra donner un nom à la fenêtre lorsque vous la créerez dans Qt Creator (figure suivante). Je l'ai appelée `FenCalculatrice` (de même que les fichiers qui seront créés).



Utilisez la fenêtre dans notre application

Pour utiliser dans notre application la fenêtre créée à l'aide de Qt Designer, plusieurs méthodes s'offrent à nous. Le plus simple est encore de laisser Qt Creator nous guider !

Eh oui, souvenez-vous : Qt Creator a créé un fichier `.ui` mais aussi des fichiers `.cpp` et `.h` de classe ! Ce sont ces derniers fichiers qui vont appeler la fenêtre que nous avons créée.

En pratique, dans la déclaration de la classe générée par Qt Creator (fichier `FenCalculatrice.h`), on retrouve le code suivant :

```
#ifndef FENCALCULATRICE_H
#define FENCALCULATRICE_H

#include <QWidget>

namespace Ui {
    class FenCalculatrice;
}

class FenCalculatrice : public QWidget
{
    Q_OBJECT

public:
    explicit FenCalculatrice(QWidget *parent = 0);
     FenCalculatrice();

private:
    Ui::FenCalculatrice *ui;
};

#endif // FENCALCULATRICE_H
```

Le fichier `FenCalculatrice.cpp`, lui, contient le code suivant :

```
#include "FenCalculatrice.h"
#include "ui_FenCalculatrice.h"

FenCalculatrice::FenCalculatrice(QWidget *parent) :
```

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

FenCalculatrice:: FenCalculatrice()
{
    delete ui;
}
```

Comment marche tout ce bazar ?

Vous avez une classe`FenCalculatrice`qui a été créée automatiquement par Qt Creator (fichiers`FenCalculatrice.h`et`FenCalculatrice.cpp`). Lorsque vous créez une nouvelle instance de cette classe, la fenêtre que vous avez dessinée tout à l'heure s'affiche !

Pourquoi ? Le fichier de la classe est tout petit et ne fait pas grand chose pourtant ?

Si, regardez bien : un fichier automatiquement généré par Qt Designer a été automatiquement inclus dans le`.cpp`:`#include "ui_FenCalculatrice.h"`

Par ailleurs le constructeur charge l'interface définie dans ce fichier auto-généré grâce à`ui->setupUi(this);`. C'est cette ligne qui lance la construction de la fenêtre.

Bien sûr, la fenêtre est encore une coquille vide : elle ne fait rien. Utilisez la classe`FenCalculatrice`pour compléter ses fonctionnalités et la rendre intelligente. Par exemple, dans le constructeur, pour modifier un élément de la fenêtre, vous pouvez faire ceci :

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

    ui->boutonEgal->setText("Egal");
}
```

Le nom du bouton`boutonEgal`, nous l'avons défini dans Qt Designer tout à l'heure (propriété`objectName`de`QObject`). Retournez voir le petit tableau un peu plus haut pour vous souvenir de la liste des noms des widgets associés à la fenêtre.

Bon, en général, vous n'aurez pas besoin de personnaliser vos widgets vu que vous avez tout fait sous Qt Designer. Mais si vous avez besoin d'adapter leur contenu à l'exécution (pour afficher le nom de l'utilisateur par exemple), il faudra passer par là.

Maintenant ce qui est intéressant surtout, c'est d'effectuer une connexion :

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

    connect(ui->boutonEgal, SIGNAL(clicked()), this, SLOT(calculerOperation()));
}
```

N'oubliez pas à chaque fois de préfixer chaque nom de widget par`ui`!

Ce code nous permet de faire en sorte que le slot`calculerOperation()`de la fenêtre soit appelé à chaque fois que l'on clique sur le bouton. Bien sûr, c'est à vous d'écrire le slot`calculerOperation()`.

Il ne vous reste plus qu'à adapter votre`main`pour appeler la fenêtre comme une fenêtre classique :

```
#include <QApplication>
#include <QtWidgets>
#include "FenCalculatrice.h"

int main(int argc, char *argv[])
{
    QApplication app(argc, argv);

    FenCalculatrice fenetre;
    fenetre.show();

    return app.exec();
```

```
}
```

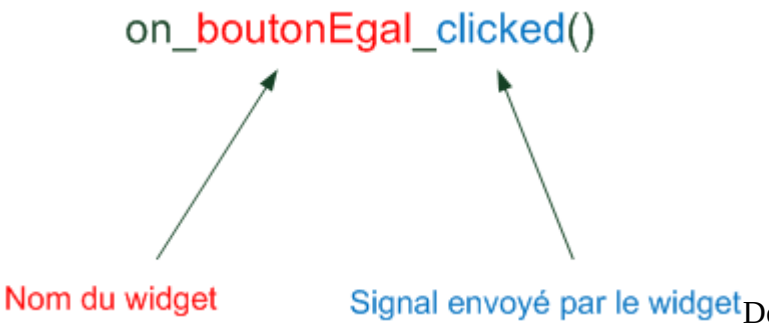
Personnalisez le code et utiliser les Auto-Connect

Les fenêtres créées avec Qt Designer bénéficient du système « Auto-Connect » de Qt. C'est un système qui crée les connexions tout seul.

Par quelle magie ?
Il vous suffit en fait de créer des slots en leur donnant un nom qui respecte une convention.

Prenons le widget `boutonEgal` et son signal `clicked()`. Si vous créez un slot appelé `on_boutonEgal_clicked()` dans votre fenêtre, ce slot sera automatiquement appelé lors d'un clic sur le bouton.

La convention à respecter est représentée sur le schéma suivante.



Donnez un nom précis à votre slot et la connexion sera automatique !

Exercice (ne me dites pas que vous ne l'avez pas vu venir) : complétez le code de la calculatrice pour effectuer l'opération correspondant à l'élément sélectionné dans la liste déroulante.

En résumé

- Qt Designer est un programme qui permet de construire rapidement ses fenêtres à la souris. Il nous évite l'écriture de nombreuses lignes de code.

- Qt Designer est intégré à Qt Creator mais existe aussi sous forme de programme externe. Il est conseillé de l'utiliser dans Qt Creator car le fonctionnement est plus simple.
- Qt Designer ne nous épargne pas une certaine réflexion : il est toujours nécessaire de rédiger des lignes de code pour faire fonctionner sa fenêtre après l'avoir dessinée.
- Qt Designer crée automatiquement un fichier de code qui place les widgets sur la fenêtre. On travaille ensuite sur une classe qui hérite de ce code pour adapter la fenêtre à nos besoins.