# Ateliers Python+Qt : Premiers pas : Prise en main de l'interface de création d'applications graphiques Qt-Designer.

par X. HINAULT

www.mon-club-elec.fr




Truc d'utilisation : visualiser ce document en mode diaporama dans le visionneur PDF. Navigation avec les flèches HAUT / BAS ou la souris.

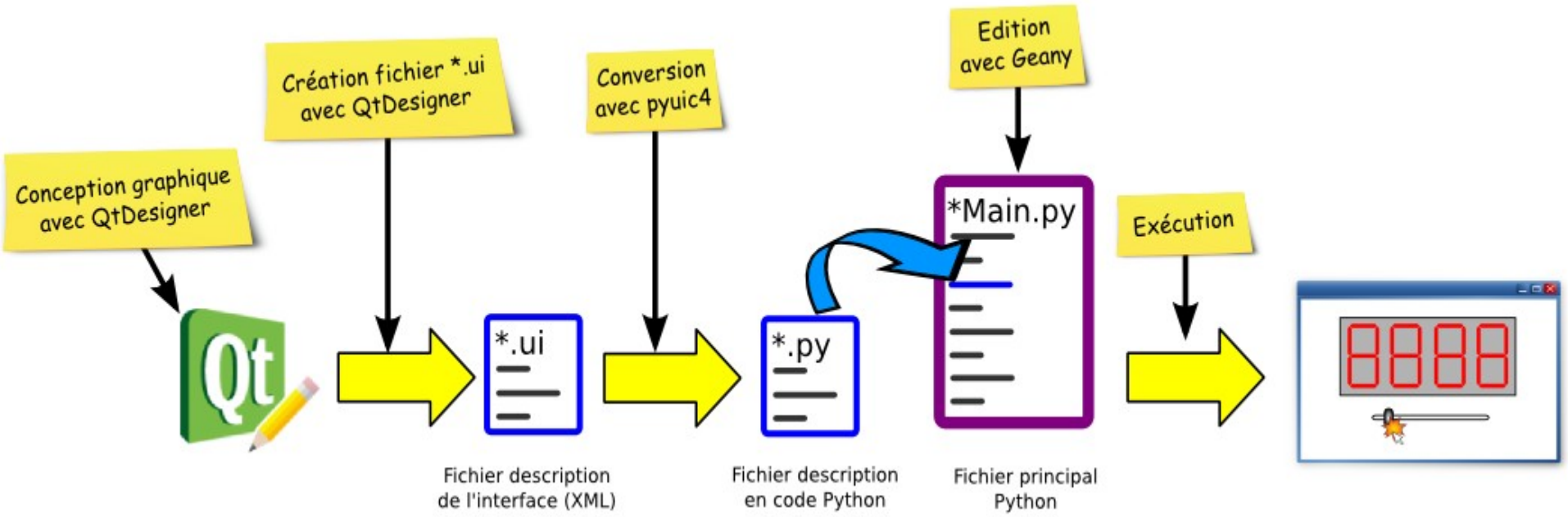
En mode fenêtre, activer le panneau latéral vous facilitera la navigation dans le document. Bonne lecture !

## Ce que l'on va faire ici

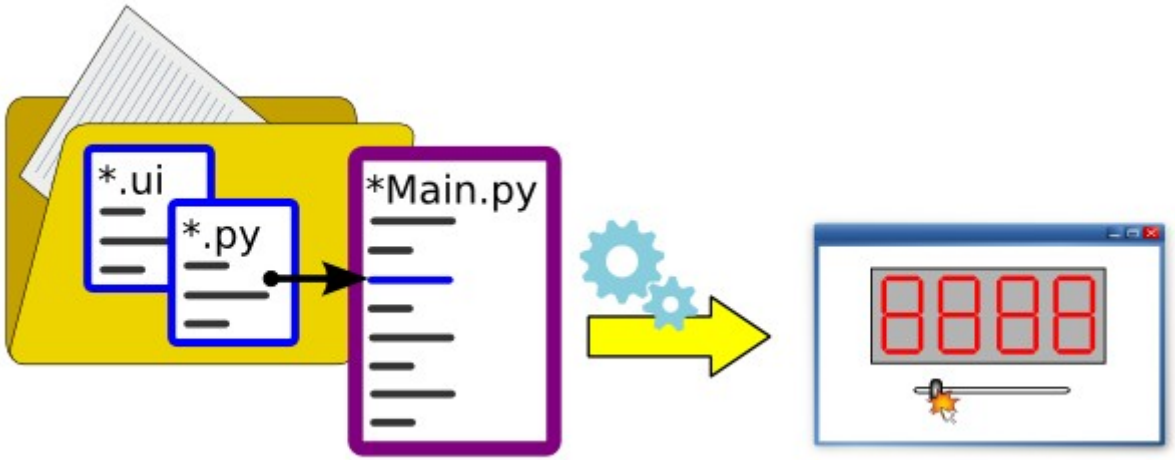
- Dans ce tutoriel, je vais vous présenter logiciel de conception graphique qui permet de concevoir des interfaces graphiques Qt en quelques clics. Je suppose ici que vous avez installé Qt-Designer sur votre système comme indiqué dans le tuto d'installation.

## Pour comprendre

- La conception d'une interface graphique en langage Python avec Qt à partir de Qt-Designer va répondre aux étapes suivantes :
  - 1. conception graphique de l'interface dans Qt Designer
  - 2. enregistrement de l'interface ainsi conçue dans un fichier de description de l'interface au format *.ui (équivalent XML)
  - 3. conversion avec un utilitaire du fichier de description *.ui en un fichier Python qui va générer l'interface identique
  - 4. création d'un fichier Python principal dans lequel sera inclut le fichier précédemment créé et édition du code actif voulu au sein de ce fichier Python principal
  - 5. exécution du fichier Python principal, soit depuis l'éditeur, soit en ligne de commande, soit par double-clic




- **De prime abord, cette séquence pourra paraître complexe, mais en fait non,** car la première étape est graphique, les étapes 2 et 3 sont « automatiques » et seule l'étape 4 est l'étape « active » en terme de codage. C'est cette étape qui est l'objet des tutos que je vous propose ici.
- Au final, on obtient un répertoire d'application qui comporte 3 fichiers :
  - le **fichier *.ui (XML) de description de l'interface** obtenu avec QtDesigner
  - le **fichier *.py (Python) de description de l'interface** obtenu avec l'utilitaire de conversion pyuic4
  - le **fichier *Main.py (Python) qui contient le code actif proprement dit** : c'est ce fichier qui sera édité puis exécuté.



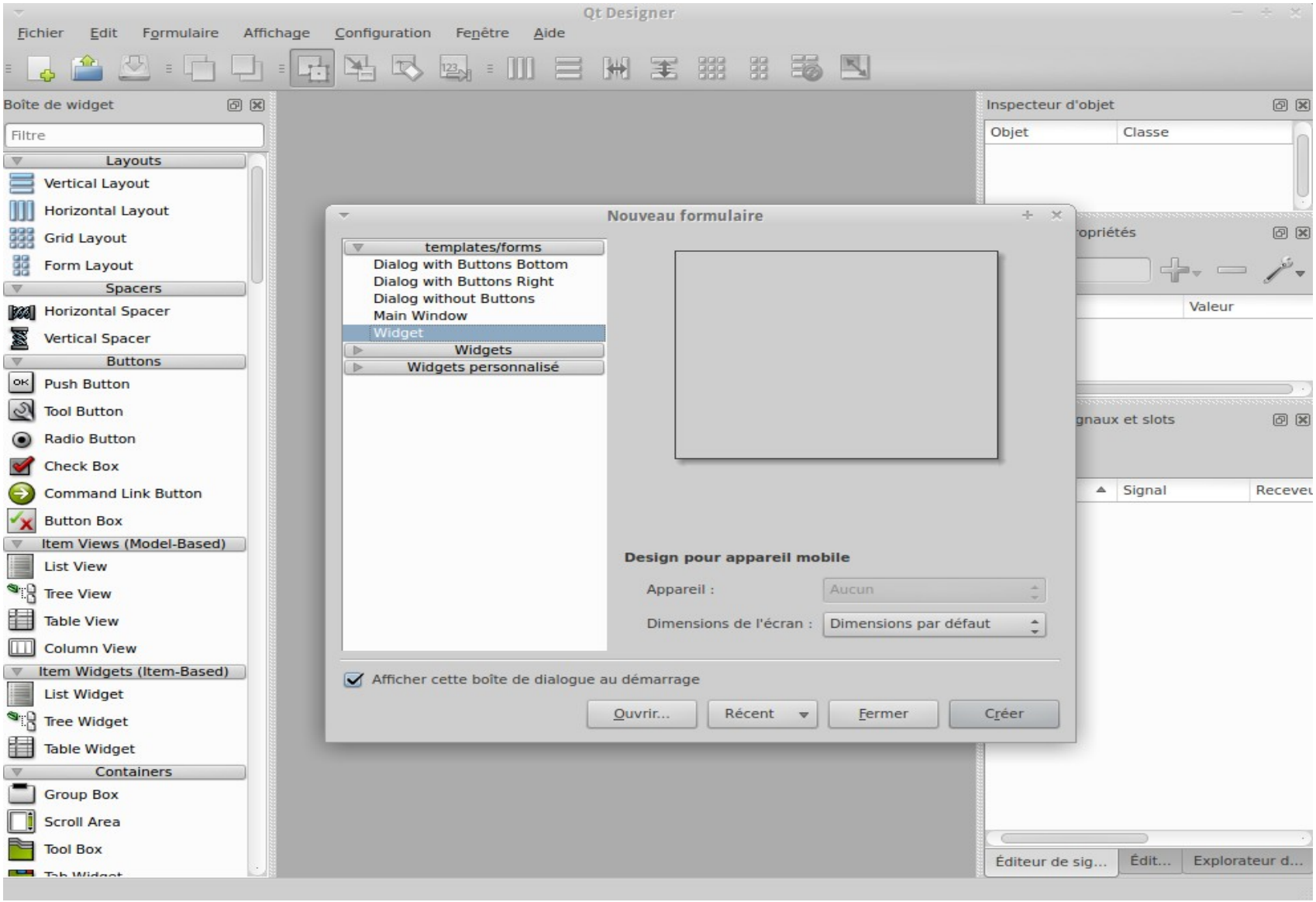

- En fait, le fichier *.ui n'est même pas obligatoire pour l'exécution (mais autant le laisser dans le répertoire puisqu'on l'a créé). Et les 2 autres fichiers sont de simples fichiers texte.
- L'utilisation de l'application ainsi créée sera très souple en fait :
  - il sera possible de copier/coller ce répertoire où on le souhaite et exécuter par un simple double-clic sur le fichier python principal.
  - de la même façon, on pourra créer une nouvelle application à partir d'une existante par simple copier/coller du répertoire et il suffira de modifier les noms de fichier avant de réaliser les adaptation voulues. Souplesse et simplicité maximale au rendez-vous !

## Premier lancement de Qt Designer

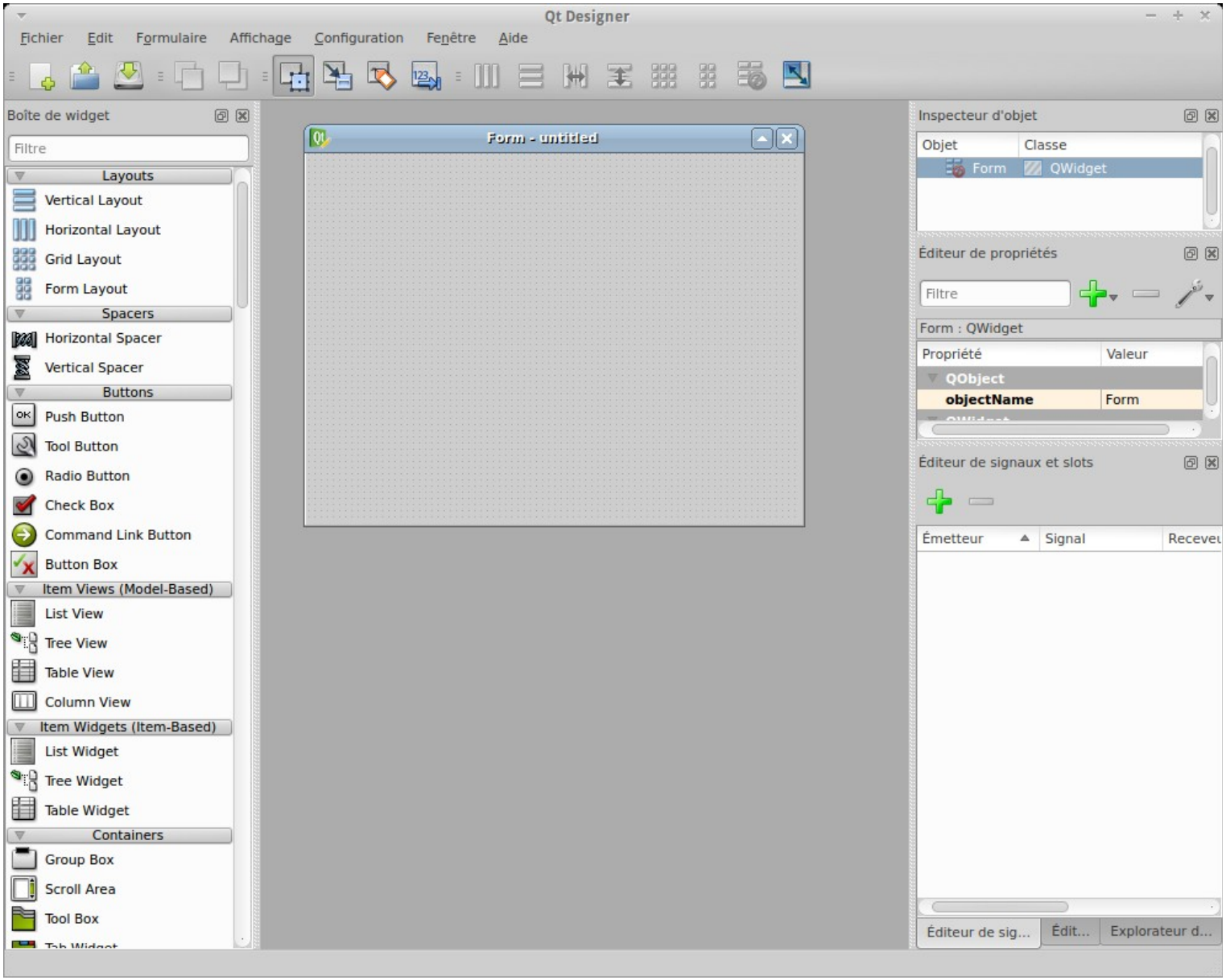
- Découvrons à présent l'interface qui va nous permettre de créer graphiquement nos interfaces graphiques : j'ai nommé Qt Designer, un outil libre et gratuit fournit avec l'espace de développement Qt.
- Pour lancer Qt Designer :
  - soit en ligne de commande avec la commande :

```
$ designer-qt4
```

  - soit depuis le menu graphique de votre distribution : Applications >Programmation/Developpement> qtdesigner
  - On obtient :



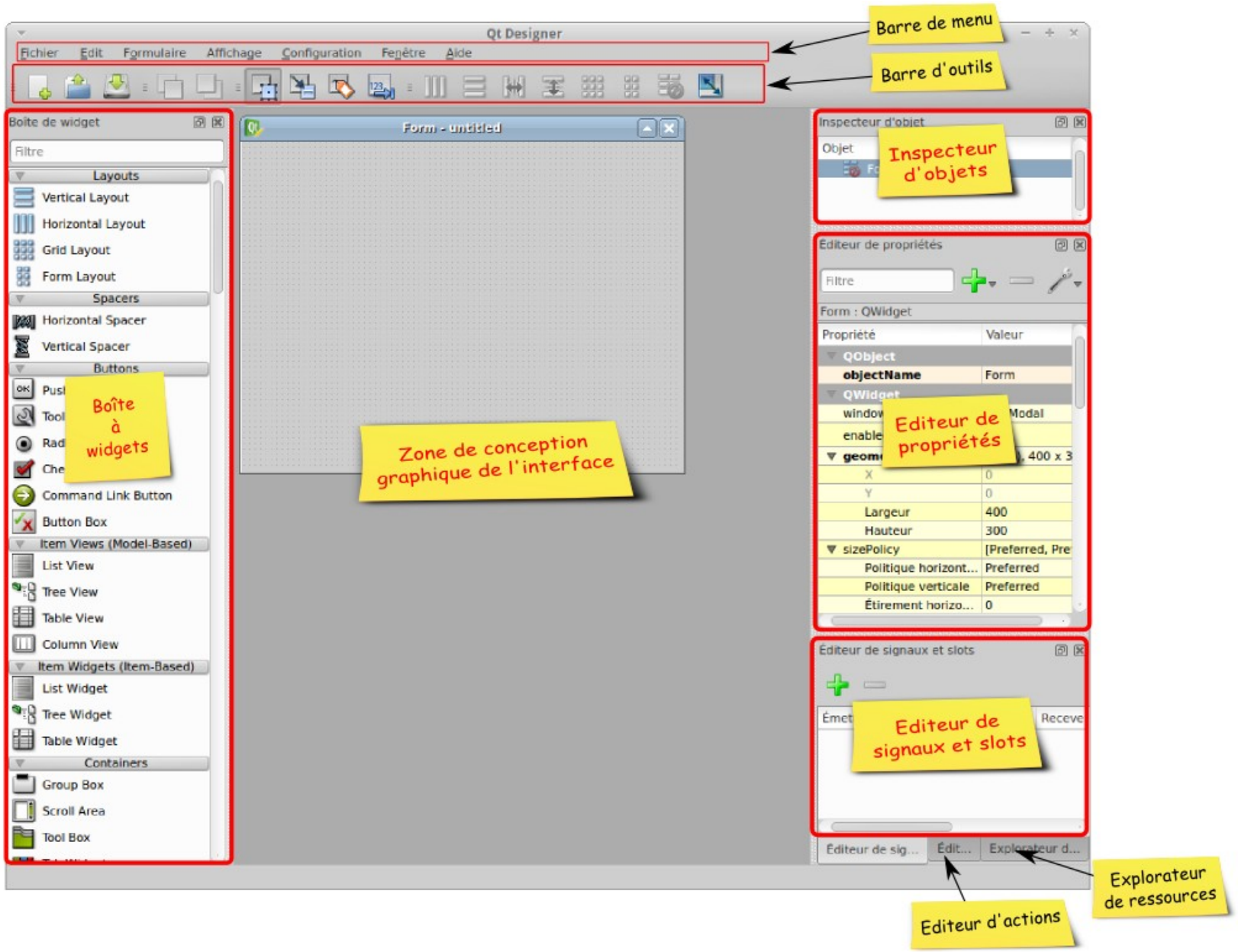

- Dans la fenêtre de dialogue qui s'ouvre, choisir Widget et cliquer sur créer. On obtient alors l'interface Qt avec un widget Form vide :




- En langage Qt, un widget est un objet graphique. « Tout est widget » pourrait-on dire. Le widget correspondant à une fenêtre est le QWidget.

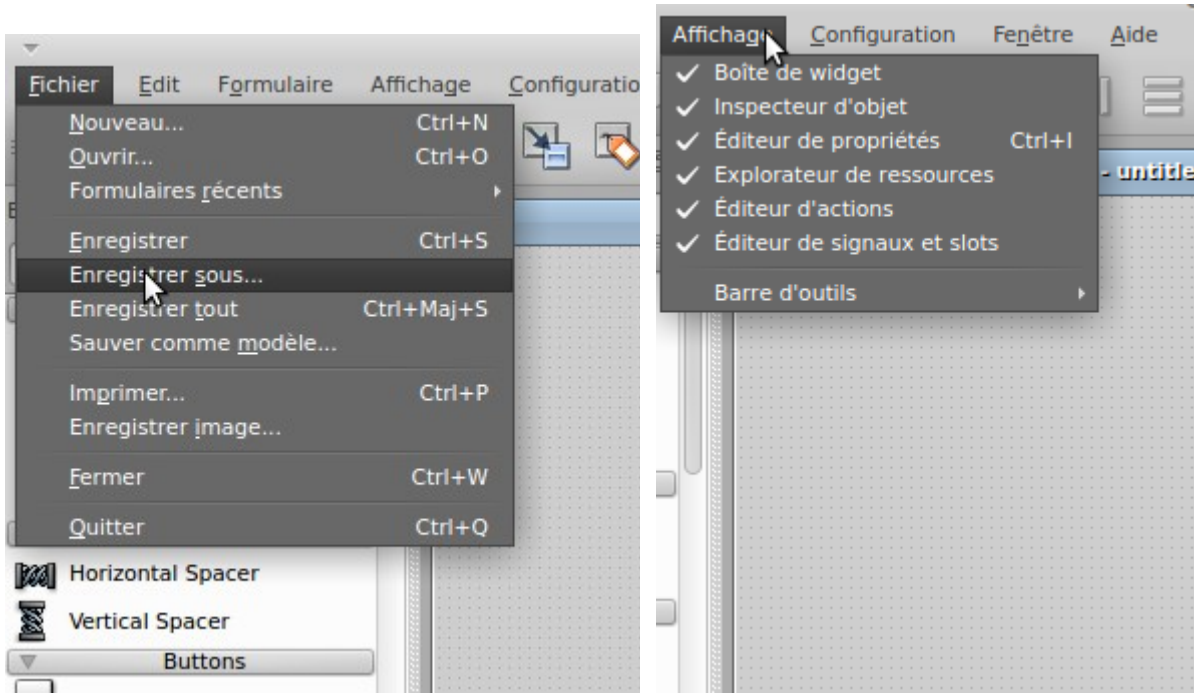
## Découvrir l'interface Qt-Designer

- L'interface Qt Designer peut sembler complexe de prime abord, mais en fait, les choses sont assez simples et logiques. On distingue :
  - dans le haut :
    - la classique barre de menu.
    - et une barre d'outil
  - sur la gauche la « Boîte de widget » qui contient tous les éléments graphiques utilisables pour vos interfaces (appelés « widgets » ) : il y en beaucoup (tant mieux!) mais peu vont servir au départ et on pourra se familiariser progressivement au fil des tutos avec les différents widgets.
  - Sur la droite, on a :
    - une fenêtre « inspecteur d'objet » : cette fenêtre va contenir la liste des éléments utilisés par l'interface en cours de conception, leur nom, leur classe.
    - Une fenêtre « Editeur de propriétés » qui donne accès aux propriétés de l'objet courant. Là encore, « beaucoup de monde »... mais rien de bien sorcier. On pourra se contenter des paramètres par défaut dans une première approche et les propriétés intéressantes seront signalées dans les différents tutos.
    - Une fenêtre à onglets contenant « l'éditeur de signaux », un « éditeur d'action » et un « explorateur de ressources ». Seul l'éditeur de signaux nous sera utile, mais c'est pas pour tout de suite.
- Noter que le menu « affichage » permet de paramétrer les différents éléments visibles ou non.
- Enfin, et surtout, la zone centrale dans laquelle seront réalisées les opérations de conception de l'interface en mode graphique comme nous le verrons. La grande force de Qt Designer réside à ce niveau : mettez les widgets en place dans votre interface, enregistrez, codez quelques lignes... et le tour est joué !
- En reprenant tout dans un beau schéma, çà nous donne :


Qt Designer
Fichier Edit Formulaire Affichage Configuration Fenêtre Aide
Barre de menu
Barre d'outils
Boîte de widget
Filtre
Layouts
Vertical Layout
Horizontal Layout
Grid Layout
Form Layout
Spacers
Horizontal Spacer
Vertical Spacer
Buttons
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
Containers
Group Box
Scroll Area
Tool Box
Boîte à widgets
Form - untitled
Zone de conception graphique de l'interface
Inspecteur d'objet
Objet
Inspecteur d'objets
Éditeur de propriétés
Filtre
Form : QWidget
Propriété
Valeur
QObject
objectName
Form
QWidget
Largeur
400
Hauteur
300
Editeur de propriétés
Éditeur de signaux et slots
Editeur de signaux et slots
Éditeur de sig...
Édit...
Explorateur d...
Editeur d'actions
Explorateur de ressources

## La barre de menu

- Je ne vais pas détailler ici toutes les options que vous aurez le loisir de découvrir au fur et à mesure. En fait, en pratique, le menu sert peu. On l'utilise essentiellement pour enregistrer le fichier obtenu. L'autre élément utile est le menu « Affichage » qui permet de paramétrer les éléments visibles (laisser en l'état par défaut) :

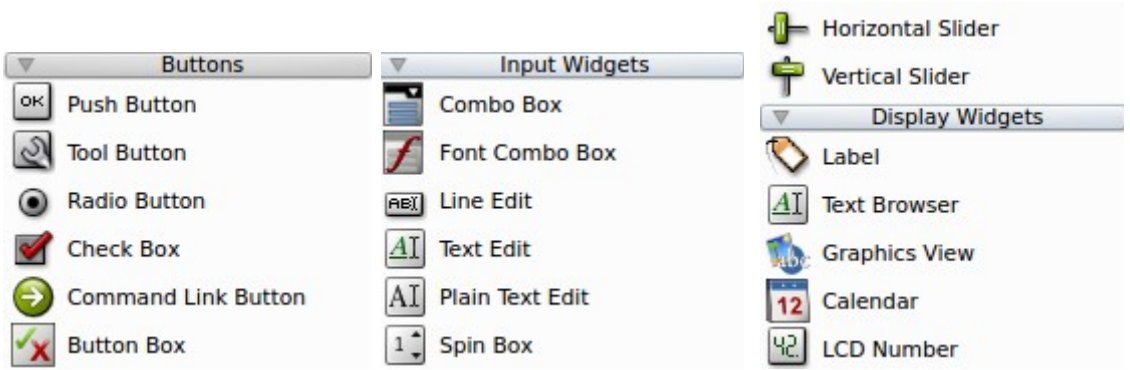


## La barre d'outils

- Au niveau de la barre d'outils, remarquer essentiellement les icônes de sélection du mode actif de l'interface Qt-Designer. Par défaut, on est en mode conception graphique. Pour éditer les signaux, on utilisera l'icône d'activation « Edition des signaux » comme nous le verrons dans un tuto dédié.

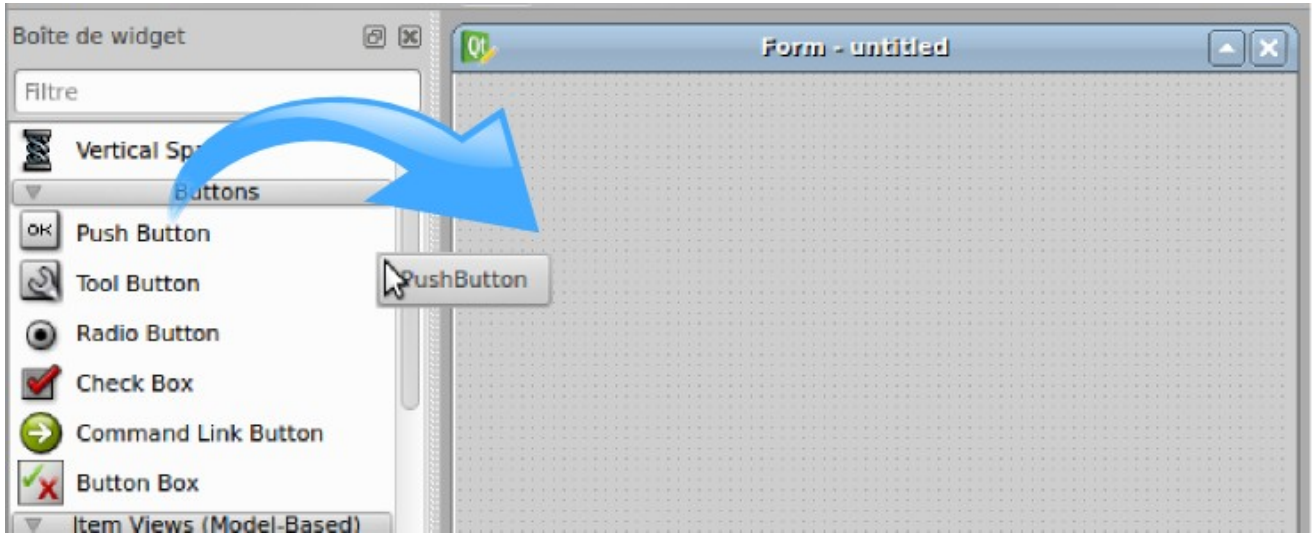

## Boîte de widgets

- Cette fenêtre contient tous les éléments graphiques disponibles pour l'édition. Le choix est important... Les widgets sont classés par catégories. Seuls quelques widgets de base sont vraiment utiles au départ. Repérer notamment le Label (étiquette texte simple), le LineEdit (champ texte), le pushButton, le Slider (réglage horizontal ou vertical), etc... Nous découvrirons ces différents widgets au fur et à mesure des tutos.

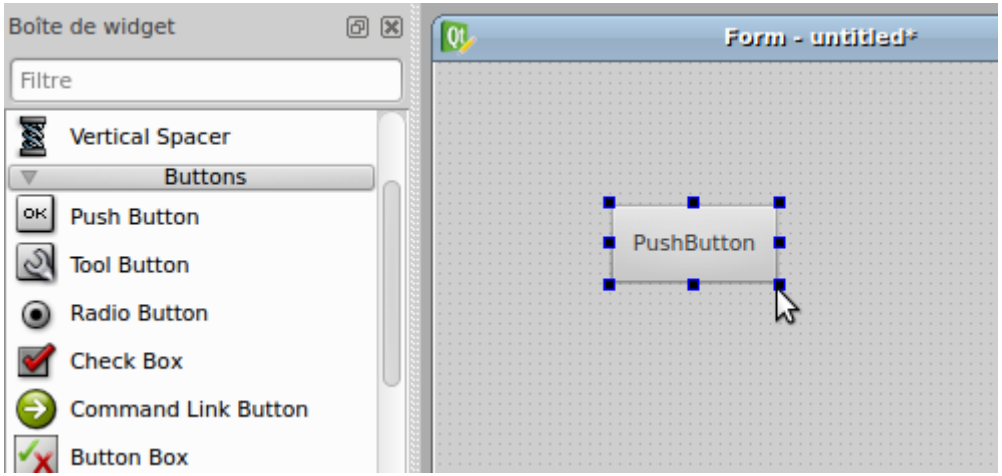



- Le principe d'utilisation est le suivant :
  - cliquer sur le widget voulu et garder le bouton gauche de souris enfoncé

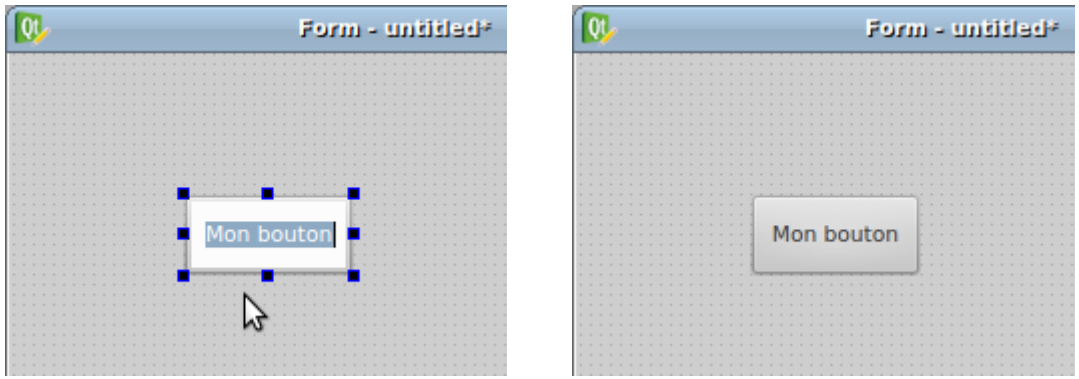

- tout en maintenant le bouton gauche de souris enfoncé, déplacer le curseur vers le widget « Form » (fenêtre de base) présent dans la zone d'édition




- puis relâcher le bouton de souris : le widget a été ajouté à l'interface. Il ne reste plus qu'à effectuer les réglages de taille, à définir les propriétés.




- Simple non ? Ensuite, pour modifier le texte du widget, double cliquer dessus, ce qui le rend éditable. Modifier et valider :

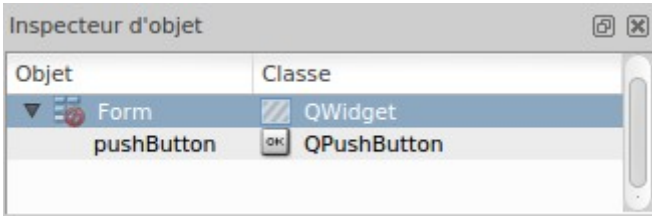



- Il suffira de répéter l'opération pour enrichir l'interface, positionner les widgets, etc... **Un outil très puissant qui vous fera gagner beaucoup de temps !**

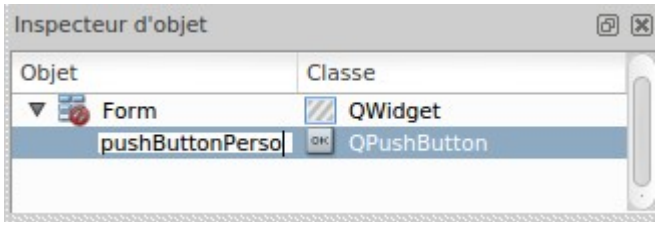
## L'inspecteur d'objet

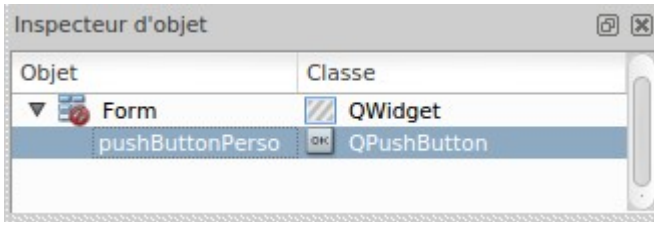
- L'inspecteur d'objet contient le listing des éléments graphiques de l'interface en cours de conception. Pour chaque élément, on dispose :
  - du nom de l'objet,
  - du nom de la classe de cet objet (son « genre », ou son type si vous voulez...)
- Le widget actif est surligné.

- Dans notre cas, l'interface à laquelle nous avons ajouté un pushButton contient les éléments suivants :




- Remarquer que les objets se voient attribuer automatiquement un nom. On pourra se contenter d'utiliser ce nom par défaut... mais il sera vite pratique de pouvoir nommer les éléments, ce qui se fait simplement en double-cliquant sur le nom de l'objet ce qui le rend éditable. Modifier puis valider :

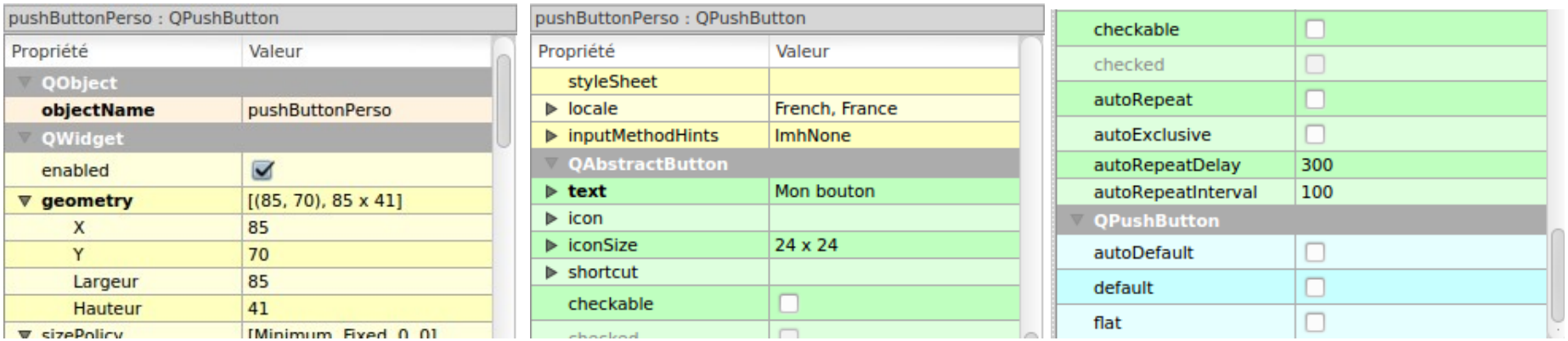



- Deux remarques importantes :
  - ne pas confondre le nom de l'objet (celui dans l'inspecteur d'objet) et son « label » (celui qui est écrit sur l'objet lui-même) : modifier le label ne modifie pas le nom de l'objet...
  - il sera important de bien noter les noms donnés aux objets que vous utiliserez (d'où l'intérêt d'utiliser les noms par défaut au début) : ces noms seront utilisés dans le code actif pour appeler ou utiliser les objets voulus. Si vous les renommez, il est donc important d'utiliser un nom explicite par rapport à la fonction de l'objet dans votre programme. Nous verrons cela au fur et à mesure des tutos...
- **Un truc à connaître :** à tout moment, vous pouvez obtenir une prévisualisation de votre interface à l'aide du raccourci clavier CONTROL + R. Il suffit de refermer la fenêtre de visualisation pour revenir à l'interface d'édition.

## L'éditeur de propriétés

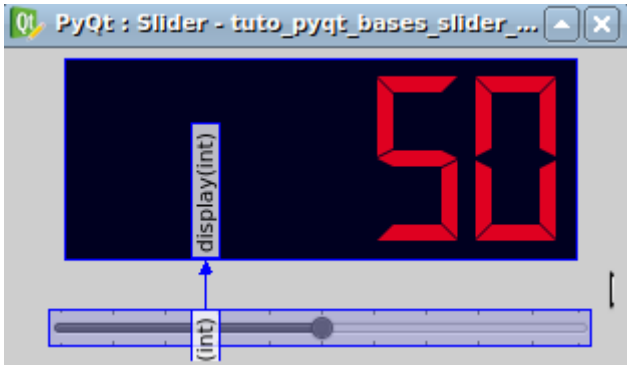
- Cette fenêtre est une des plus importantes lors de la conception d'une interface : elle permet de fixer graphiquement les propriétés de chaque objet. De prime abord, cela semble compliqué... mais on peut se contenter dans une première approche de laisser les propriétés par défaut inchangées. Les propriétés utiles seront abordées pour chaque type d'objet dans les différents tutos que je vous propose.
- Il faut savoir que chaque objet hérite de une ou plusieurs classes parentes. En bon français, cela veut dire que l'objet « baleine » appartient à la « classe » parente « mammifère marin » qui elle-même appartient à la classe parente des « mammifères » qui elle-même appartient à la classe parente des « animaux vertébrés », etc...
- De la même façon, un pushButton va hériter de la classe parente QabstractButton qui elle-même va hériter de la classe parente Qwidget qui elle-même va hériter de la classe parente Qobject.
- Il faut savoir que tout objet, en plus de ses propriétés propres, va hériter également des propriétés des classes parentes. Ceci abouti au fait qu'il peut-être compliqué de visualiser parmi les propriétés d'un objet celles qui lui sont propres, de celles dont il hérite...
- Je vous parle de çà, non pas pour vous effrayer, mais pour vous montrer comment Qt Designer est bien fait : l'interface va utiliser une couleur d'arrière plan différente pour les propriétés venant de classes parentes différentes, ce qui donne :

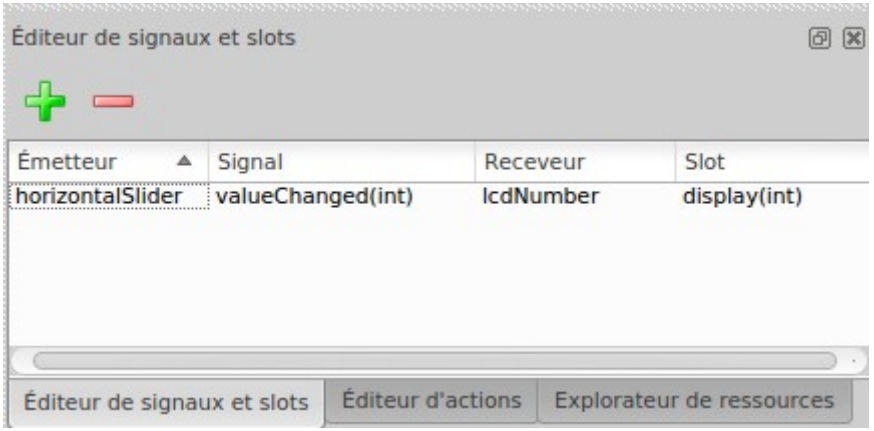



- Ceci a 2 conséquences très pratiques :
  - on visualise très facilement l'héritage ascendant de l'objet que l'on utilise, et donc on a conscience des classes parentes dont il hérite, ce qui facilite la compréhension de logique du code que vous écrirez ensuite (notamment l'articulation entre les classes d'objets),
  - on distingue très bien la hiérarchie des propriétés d'un objet, d'un seul coup d'oeil

## L'éditeur de signaux

- Je ne détaillerai pas ici l'utilisation des signaux, ce qui sera fait dans un tuto dédié, mais sachez que cette fenêtre vous permettra de visualiser les relations qui auront été créées graphiquement entre les objets.
- A titre d'exemple, voici ce que l'on obtiendra :




## Enregistrement du fichier *.ui

- Une fois la configuration graphique terminée, vous allez pouvoir enregistrer votre interface. Le fichier va être enregistré au format *.ui.
- On obtient un fichier XML qui est enregistré dans le répertoire voulu. Ce fichier décrit l'interface et l'ensemble des paramètres utilisés. C'est ce fichier qui servira de base pour générer la trame du code de votre application graphique.
- Ce fichier est un simple fichier texte au format « XML ». Editer ce fichier n'a rien d'obligatoire et vous n'aurez pas à le modifier vous même. Cependant, si vous êtes curieux, vous pouvez l'éditer notamment avec l'éditeur Geany que vous avez dû déjà installer. A titre indicatif, voici le

fichier obtenu dans l'exemple précédent d'un simple bouton poussoir ajouter à la fenêtre de base (objet Form) :

```
<?xml version="1.0" encoding="UTF-8"?>
<ui version="4.0">
 <class>Form</class>
 <widget class="QWidget" name="Form">
  <property name="geometry">
   <rect>
    <x>0</x>
    <y>0</y>
    <width>270</width>
    <height>163</height>
   </rect>
  </property>
  <property name="windowTitle">
   <string>Form</string>
  </property>
  <widget class="QPushButton" name="pushButtonPerso">
   <property name="geometry">
    <rect>
     <x>85</x>
     <y>60</y>
     <width>85</width>
     <height>41</height>
    </rect>
   </property>
   <property name="text">
    <string>Mon bouton</string>
   </property>
  </widget>
 </widget>
 <resources/>
 <connections/>
</ui>
```

- Cela vous donne une idée de « tout le boulot » que Qt Designer fait pour vous... et que par conséquent vous n'aurez pas à faire vous-même : vous allez pouvoir ainsi vous concentrez, comme nous le verrons, sur la partie « active » de votre code.

## Conclusion

- Voilà, vous disposez des bases pour utiliser l'interface Qt Designer : comme pour tout nouveau logiciel, il vous faudra un petit temps de rodage. Une fois passée la phase de « découverte », vous verrez que Qt Designer est très agréable à utiliser. Vous vous familiariserez pas à pas avec Qt Designer, du plus simple au plus complexe, à l'aide des autres tutos que je vous propose ici. Suivez le guide !