Interface Graphique en Python avec PyQt5

Dr. ABOUABID Hamza

Académie Polytechnique de Tanger

21 mars 2024

What is PyQt ?

PyQt is the fusion of Python, a programming language known for its ease of learning, and Qt, a comprehensive framework originally written in C++ for building graphical interfaces. PyQt acts as a bridge between these two technologies, bringing Qt into the Python environment.

Python and Qt

- Python : Popular programming language, easy to learn.
- Qt : Multiplatform library, primarily for graphical user interfaces.
- PyQt : Connects Python with the capabilities of Qt.

Features of Qt

- GUI development capabilities.
- Modules for SQL database access.
- Integrated web browser.
- Multimedia functionalities.

Advanced Features and Rich Environment

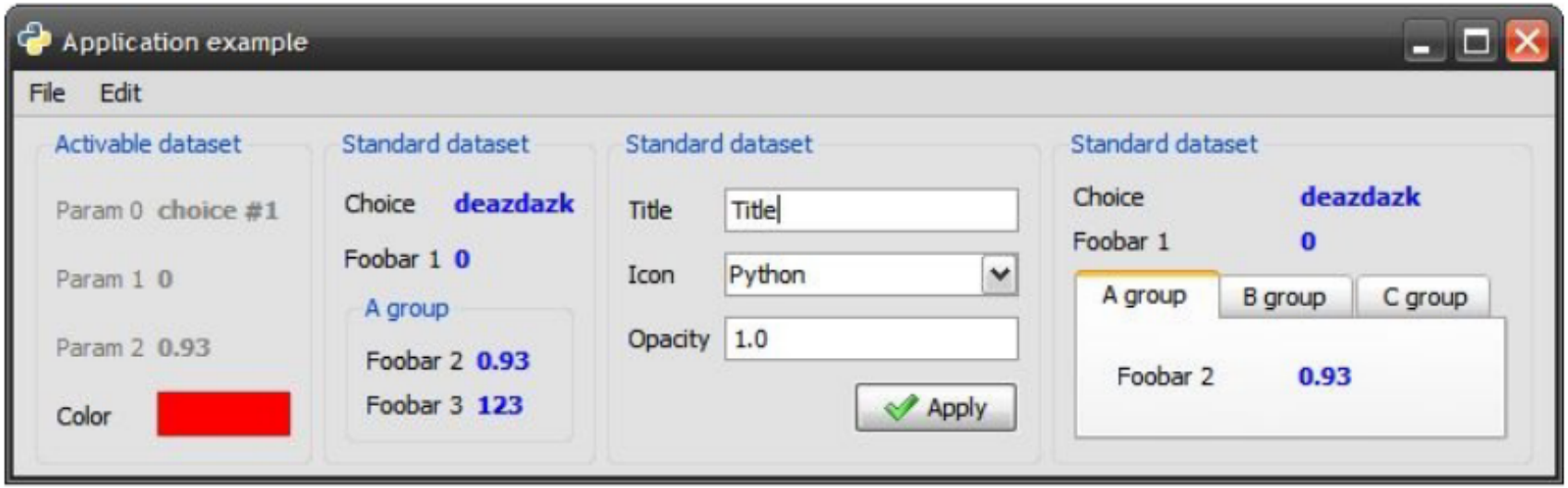
Qt extends its functionality to mapping and localization tools, wireless communication (NFC, Bluetooth), data visualization, and graphics. It also boasts a rich environment with many additional extension libraries.

Adaptability and Qt Quick

- Adaptability : Runs on mobile platforms like Android, iOS, and Windows Phone. Hybridization with web applications.
- Qt Quick : Declarative interface development technology, simplifying GUI creation.

Table des matières

Figure 1.1 : *Exemple d'interface réalisée avec GuiData*



PyQt et les autres bibliothèques de développement d'interfaces graphiques

- **PyQt vs Other Libraries :**
  - Comparaison avec Tkinter et wxPython.
  - Avantages de PyQt : Approche déclarative, documentation de qualité, Qt Designer.
- **Considérations de Licence :**
  - Tkinter sous licence Python, wxPython sous licence wxWidgets.
  - PyQt sous GPL ou licence commerciale.
  - PySide comme alternative sous LGPL.
- **Développement et Communauté :**
  - Les trois bibliothèques sont matures et disposent d'une grande communauté.
  - PyQt préféré pour son environnement et ses fonctionnalités techniques.

Choix d'une Interface dans PyQt

- Choix entre différentes approches : Qt Widgets, Qt Quick, Graphics View.
- Utilisation de méthodologies adaptées au type d'application.
- Complémentarité possible entre ces approches.

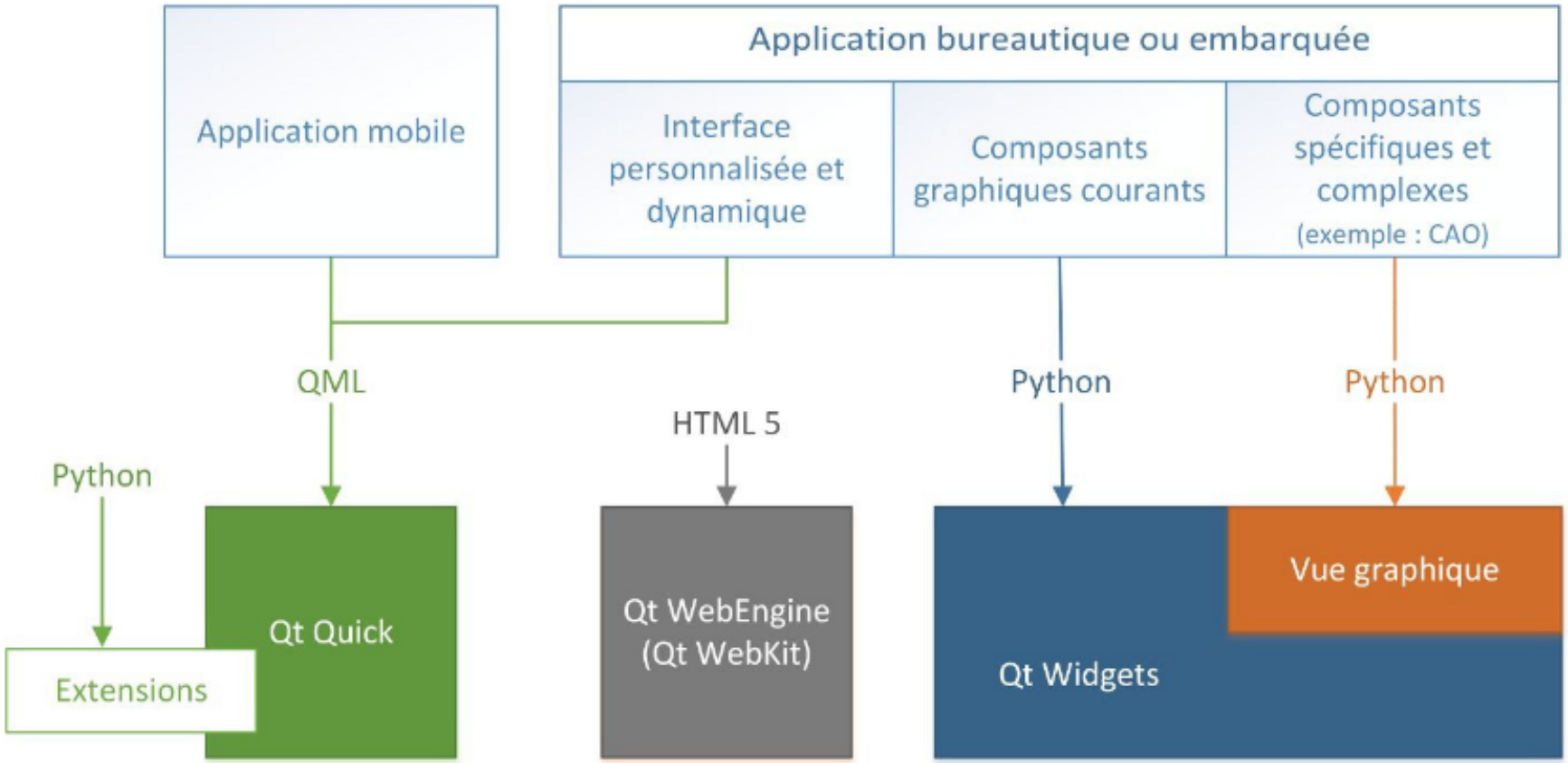
Quand Utiliser Quelle Interface

- **Qt Widgets** : Pour interfaces graphiques classiques sur PC, avec clavier et souris.
- **Qt Quick** : Pour téléphones, tablettes, et écrans tactiles ; nécessite l'apprentissage de QML.
- **Graphics View** : Pour graphiques complexes et interactifs.
- **Qt WebEngine** : Pour applications hybrides, web et mobile.

Vue Graphique vs Qt Quick

- **Historique et Évolution** : Vue graphique stable depuis 2006, Qt Quick évolué vers des performances optimisées avec OpenGL.
- **Choix Technique** : Vue graphique pour la détection de collisions et rendu d'éléments multiples, Qt Quick pour des interfaces modernes et performantes.
- **Implications pour le Développeur** : Vue graphique facile à étendre en Python, Qt Quick orienté vers l'extension en QML.
- **Conclusion** : Chaque technologie a ses avantages selon le type d'application ; le livre couvre les deux pour un choix éclairé.

Table des matières



Installation de Python et PyQt

- **Prérequis** : Installation de Python nécessaire.
- **Installation de PyQt** : Privilégier une version déjà compilée en raison de dépendances C++.
- **Utilisation de PyQt5** : Focus sur PyQt5 plutôt que PyQt4.
- **Installation sur Windows et macOS** :
  - Utiliser pip : `pip3 install pyqt5` (ou `pip` sur certains systèmes).
- **Installation sur Linux** :
  - Utiliser le gestionnaire de paquets : `sudo apt-get install python3-pyqt5` pour Ubuntu, `sudo yum install python3-qt5` pour Fedora.
- **Installation d'Extensions PyQt** :
  - Utiliser pip : par exemple, `pip3 install Camelot`, `pip3 install pyqtgraph`, etc.
- **Méthode Alternative** : Compilation de PyQt pour Python (uniquement si nécessaire).

Distribuer une Application PyQt - Concepts et Défis

- **Contraste avec les Environnements Compilés** : Simplicité de lancement en développement, complexité en distribution.
- **Distribution** : Comprendre les rôles des machines hôte et cible, et la possibilité de croisement de plateformes.
- **Considérations Clés** :
  - Installation de Python et PyQt sur la machine cible.
  - Gestion des multiplateformes et dépendances.
  - Droits administrateurs et profil de l'utilisateur final.
- **Isolation de Python** : Importance d'éviter les conflits avec les systèmes existants.

Méthodes de Distribution d'une Application PyQt

- **Distribution par Copie des Sources** :
  - Installation de l'environnement de développement sur la machine cible.
  - Avantages : Simplicité, légèreté, maintenance aisée.
  - Inconvénients : Complexité pour l'utilisateur, risques de conflits.
- **Utilisation de Machines Virtuelles** : Alternative pour limiter les inconvénients, mais avec des considérations de taille et de performance.
- **Distribution par Outils (Freezing)** :
  - Création d'un paquet autonome avec Python/PyQt inclus (ex : PyInstaller, pyqtdeploy).
  - Avantages : Indépendance de l'environnement cible, simplification de l'installation.
  - Choix d'outil selon les besoins (multiplateforme, support PyQt, etc.).

Vue Détaillée sur les Outils de Distribution (PyInstaller et pyqtdeploy)

- **PyInstaller** :
  - Supporte Python 2.7 et 3.3+ et automatiquement les modules nécessaires.
  - Multiplateforme mais pas de croisement de plateformes.
  - Options de distribution : fichier exécutable unique ou répertoire complet.
  - Installation via pip : `pip3 install pyinstaller`.
- **pyqtdeploy** :
  - Développé par Riverbank Computing, spécifique à PyQt.
  - Plus complexe mais permet le croisement de plateformes.
  - Idéal pour des situations nécessitant un contrôle plus poussé de l'environnement de déploiement.
- **Utilisation Pratique** :
  - Lancer PyInstaller depuis le répertoire contenant le programme principal.
  - Choix entre un fichier exécutable ou un répertoire selon les besoins.