

Formation Qt : développer des interfaces graphiques en C++ avec Qt Widgets et QML

Réf. SF33502

Concevez des applications graphiques C++ multiplateformes avec Qt, des écrans Qt Widgets classiques aux interfaces Qt Quick en QML branchées sur votre code C++.

4 j Durée	2 600 € HT Inter / participant	6 200 € HT Intra (≤12 pers.)	0 Sessions à venir
--------------	-----------------------------------	---------------------------------	-----------------------

Objectifs pédagogiques

- ✓ Configurer un projet Qt avec Qt Creator et CMake
- ✓ Implémenter la communication entre objets avec le modèle objet Qt et les signaux et slots
- ✓ Concevoir une interface Qt Widgets avec fenêtres, dialogues, menus, layouts et widgets personnalisés
- ✓ Afficher et éditer des données avec l'architecture Model/View
- ✓ Développer une interface Qt Quick en QML connectée à un backend C++
- ✓ Intégrer les API Qt courantes (fichiers, bases de données, réseau, threads, traduction) dans une application graphique

Public visé & prérequis

Public visé Développeurs

Prérequis- Maîtriser la programmation orientée objet en C++ (classes, héritage, polymorphisme, pointeurs et références) - Connaître les lambdas et les pointeurs intelligents du C++ moderne est un plus

Programme détaillé

Écosystème Qt et environnement de développement

- Framework Qt : modules principaux (Core, GUI, Widgets, Quick, SQL, Network) et plateformes cibles desktop, mobile et embarquées
- Licences Qt : éditions open source (LGPL, GPL) et commerciale, conséquences pour la diffusion d'une application
- Qt Widgets ou Qt Quick : critères de choix selon le type d'interface, et approche hybride
- Qt Creator : kits de compilation, exécution et débogage d'une application Qt
- Projet CMake : déclaration des modules Qt, de l'exécutable et des ressources ; qmake pour reprendre les projets existants

Activités pratiques :

- Installation de Qt et configuration d'un kit dans Qt Creator
- Création du projet CMake de l'application fil rouge (application de gestion simple)

Modèle objet Qt et signaux/slots

- QObject et hiérarchie parent/enfant : gestion automatique de la mémoire et durée de vie des objets
- Meta-Object Compiler (MOC) et macro Q_OBJECT : métadonnées et introspection à l'exécution
- Système de propriétés (Q_PROPERTY) : accesseurs, signal de notification, propriétés dynamiques
- Signaux et slots : syntaxe de connexion moderne avec pointeurs de fonction et lambdas, types de connexion
- Boucle d'événements et timers : fonctionnement de l'application Qt, QTimer
- Pièges courants : connexions multiples, objets détruits, déconnexion

Activités pratiques :

- Création d'une classe métier qui notifie ses changements par signaux
- Connexion de composants par lambdas et déclenchement d'actions périodiques avec un timer

Construire une interface avec Qt Widgets

- QMainWindow : zone centrale, menus, barres d'outils, barre d'état et actions (QAction)
- Dialogues : QDialog modal et non modal, boîtes de messages et dialogues standard (fichiers, couleurs)
- Composants courants : champs de saisie, listes, cases à cocher, onglets et validation de la saisie
- Layouts : disposition horizontale, verticale, en grille et en formulaire, adaptation au redimensionnement et aux écrans haute résolution
- Qt Designer : conception des fichiers .ui et intégration dans le code C++
- Système de ressources : icônes et images embarquées dans l'exécutable
- Zones de défilement et impression : afficher un contenu volumineux, imprimer un document

Activités pratiques :

- Écran principal du fil rouge avec menus, barre d'outils et barre d'état
- Formulaire de saisie conçu dans Qt Designer, avec validation et boîtes de dialogue

Événements et widgets personnalisés

- Gestionnaires d'événements : souris, clavier, redimensionnement, surcharge des méthodes d'événement
- Filtres d'événements : intercepter les événements d'un autre objet sans le dériver
- Widget personnalisé : hériter de QWidget, taille préférée, propriétés et signaux propres
- Dessin avec QPainter : formes, texte et images dans l'événement de peinture
- Intégration dans Qt Designer : promotion d'un widget personnalisé
- Apparence : feuilles de style Qt (QSS) et styles de la plateforme

Activités pratiques :

- Création d'un widget représentant un formulaire réutilisable, intégré à Qt Designer
- Dessin d'un indicateur graphique (jauge, voyant) réagissant à la souris
- Habillage de l'application avec une feuille de style

Afficher et manipuler des données avec Model/View

- Architecture Model/View : rôles respectifs du modèle, de la vue et du delegate
- Vues de liste, de tableau et d'arbre : différence avec les widgets à éléments intégrés
- QStandardItemModel : modèle prêt à l'emploi pour les cas simples
- Modèle sur mesure : dériver QAbstractTableModel, rôles de données, notification des changements
- Tri, filtrage et recherche : modèle proxy QSortFilterProxyModel
- Édition et delegates : éditeurs personnalisés et validation des valeurs

Activités pratiques :

- Tableau des éléments du fil rouge avec tri, filtre et recherche
- Édition des données avec un delegate personnalisé

Interfaces déclaratives avec Qt Quick et QML

- Qt Quick : architecture, moteur QML et rendu graphique accéléré
- Syntaxe QML : objets, propriétés, identifiants, expressions JavaScript
- Property binding : liaisons déclaratives et mise à jour automatique de l'interface
- Éléments de base : rectangles, textes, images, interaction avec la souris et le clavier
- Positionnement : ancres, positionneurs et layouts Qt Quick
- Qt Quick Controls : boutons, champs, menus, dialogues et styles prêts à l'emploi
- Composants réutilisables : fichiers QML, propriétés et signaux personnalisés
- États, transitions et animations : rendre l'interface vivante et fluide

Activités pratiques :

- Création en QML d'une interface simple avec champs de texte, menus et boîtes de dialogue
- Conception d'un composant QML réutilisable animé

Intégrer C++ et QML

- Architecture hybride : backend C++ et frontend QML, partage de la logique avec une interface Widgets
- Module QML dans CMake : déclarer les fichiers QML et les types C++ du projet
- Exposer des types C++ à QML : enregistrement des types, propriétés, méthodes invocables et signaux
- Objets singleton : partager un service C++ avec toute l'interface QML
- Modèles C++ dans les vues QML : dériver QAbstractListModel, afficher les données dans ListView et ses délégués
- Échanges dans les deux sens : appeler le C++ depuis QML, réagir en QML aux signaux C++

Activités pratiques :

- Écran QML prototype connecté au backend C++ de l'application fil rouge
- Affichage d'une liste de données C++ dans une vue QML

API Qt pour une application complète

- Conteneurs et chaînes Qt : QList, QHash, QString et interopérabilité avec la bibliothèque standard
- Fichiers et formats : lecture et écriture de fichiers, paramètres d'application (QSettings), JSON ; XML avec QDomStreamReader
- Bases de données avec Qt SQL : connexion SQLite, requêtes préparées, modèles SQL pour les vues
- Réseau avec Qt Network : requêtes HTTP asynchrones et traitement des réponses JSON, sockets TCP
- Multithreading : garder l'interface réactive, worker objects et QThread, QtConcurrent, communication entre threads par signaux
- Internationalisation : textes traduisibles, Qt Linguist et chargement des traductions
- Déploiement : outils de déploiement Qt et dépendances à livrer avec l'application

Activités pratiques :

- Ajout d'un accès à une base SQLite et d'un journal dans un fichier
- Envoi des données de l'application à un serveur distant en HTTP
- Exécution d'un traitement long en tâche de fond avec barre de progression
- Traduction de l'interface de l'application

Pourquoi choisir Sparks Formation ?

20 ans d'expertise IT, des formateurs experts et des sessions garanties contre l'annulation.

20 ans
d'expertise

1 068+
formations

11 178
apprenants formés/an

+67
Net Promoter Score

Avis des participants

Note moyenne des participants de la filière Langages & Écosystèmes Backend : 4,9/5 — 110 avis

Selim B. — Très bonne formation, très instructif même après quelques années de Python dans les bras. Une très petite équipe de stagiaires qui facilite les échanges. Je redoutais le mode...

Alexis C. — Anthony explique bien les concepts attendus et prend le temps de répondre aux questions et aux différents problèmes rencontrés. Le support présenté créé par lui-même est différent des...

Votre formateur

Albina S. — Formatrice Java & Python

Albina intervient sur des formations portant sur les langages et écosystèmes backend. Parmi les formations qui lui sont confiées figurent notamment « C# » et « C++ ». La conception...

Ils nous font confiance

Cette année notre filière Langages & Écosystèmes Backend a accompagné plus de 292 entreprises dont :

SOPRA STERIA RENAULT SAS ORANGE CAMPUS TECH ORANGE INNOVATION SCHOOL STMICROELECTRONICS GRANT THORNTON FRANCE
ORANGE BUSINESS RTE SII ILE DE FRANCE THALES GLOBAL SERVICES +282 autres

Financement & certifications

Cette formation est éligible aux financements OPCO et FNE-Formation. Nos équipes vous accompagnent dans le montage de votre dossier.



Réservez votre place dès maintenant

20 ans d'expertise · 1 000+ formations · NPS +67 · réponse sous 24h ouvrées

0 805 950 800

demande@sparks-formation.com