Tutoriel Qt Quick

par Traducteur : Louis du Verdier (Site Web) Qt Developer Network

Date de publication : 24/04/11

Dernière mise à jour : 08/06/2011

Qt Quick est sorti avec Qt 4.7. Les entreprises l'utilisent pour créer des interfaces utilisateur (IU) pour des *set-top boxes*, tablettes, systèmes informatifs de voiture, *e-readers* et téléphones mobiles. Bien d'autres entreprises évaluent sérieusement Qt Quick. Un tel intérêt est assez surprenant pour une technologie nouvelle n'ayant donc pas encore fait ses preuves. Ainsi, Qt Quick doit taper sur un nerf - ou, en fait, un grand nombre de nerfs. Qt Quick fournit un langage déclaratif, QML (*Qt Meta-object Language*, langage de méta-objets de Qt), pour la conception et l'implémentation d'interfaces utilisateur. QML est basé sur **CSS** et sur **JavaScript**. De grandes parties du code QML correspondent uniquement à des paires nom-valeur de propriétés dans le style CSS (par exemple, color: "red"). Le reste, les parties typiquement comportementales, comme la réponse à un clic de la souris, est écrit en JavaScript. Grâce à sa similarité avec CSS et JavaScript ainsi qu'avec sa simplicité, QML est facilement pris par les développeurs Web et Flash. De plus, il est une évidence pour les développeurs Qt/C++, C et Java. Grâce à sa simplicité, Qt Quick facilite l'écriture d'interfaces utilisateur fluides - rendues populaires par l'iPhone d'Apple.

I - L'article original

Le Qt Developer Network est un réseau de développeurs utilisant Qt afin de partager leur savoir sur ce framework. **Vous pouvez le consulter en anglais.**

Nokia, Qt, Qt Quarterly et leurs logos sont des marques déposées de *Nokia Corporation* en Finlande et/ou dans les autres pays. Les autres marques déposées sont détenues par leurs propriétaires respectifs.

Cet article est la traduction de **Qt Quick Tutorial**.

II - Introduction

Les designers d'interfaces utilisateur obtiennent un accès à Qt Quick à travers le Quick Designer, qui est un outil de conception visuelle pour les applications QML. Comparons cela avec le processus normal de développement. Les designers viennent avec un fantastique design d'interface utilisateur fait avec Photoshop et en Flash. Alors, les développeurs réimplémentent ce design en C++ ou en Java. Après un certain nombre de jours, voire même de semaines, les développeurs montrent fièrement leur résultat - qui ne ressemble même pas de loin au design original (voir le **fameux tree swing cartoon** pour illustration). Avec Qt Quick, c'est différent, parce que les designers et les développeurs vont toujours travailler sur le même artefact, qui est également identique au produit final. Cela permet une interaction serrée entre les designers et les développeurs et en résulte en quelque sorte des nouvelles idées ou de nouvelles fonctionnalités.

Comme si ce n'était pas suffisant, Qt Quick a accès à toute la puissance de Qt. Tous les propriétés, signaux et slots d'une classe dérivée de QObject sont instantanément accessibles depuis QML. Actuellement, tous les composants visuels de QML sont implémentés en tant que QGraphicObject et sont exposés à QML par le biais du système de métaobjets de Qt. On étend QML avec ses propres composants d'une manière similaire.

Jusqu'ici, on a fait un grand nombre de promesses. Qt Quick est à la fois simple d'utilisation et d'apprentissage, il fait le pont au-dessus du gouffre séparant les designers des développeurs. Il est puissant et il permet d'écrire des interfaces utilisateur fluides. Maintenant, on doit tenir ces fameuses promesses et c'est exactement l'objectif de ce tutoriel Qt Quick. On veut montrer comment il est simple de concevoir des interfaces utilisateur surprenantes avec Qt Quick et on veut apprendre comment fabriquer ces interfaces. Lorsque vous aurez marqué votre chemin à travers ce tutoriel, vous devrez avoir votre propre opinion du fait que Qt Quick soit ou non la technologie idéale pour votre prochain projet d'interface utilisateur et, si c'est le cas, vous devriez avoir les moyens techniques de réaliser ce projet.

III - Objectif

Maintenant, il est temps de donner des aperçus des sujets abordés par ce tutoriel. Les neuf modules qui suivent proviennent de la formation QML que nous (Nokia) avons donnée à nos clients. Ils sont déjà disponibles en tant que code QML lourdement commenté. Bien sûr, il y a bien d'autres sujets sur QML : les données *flippables*, les vues Web, l'internationalisation, la création d'objets dynamiques, les optimisations des performances et ainsi de suite. Nous espérons ajouter ces sujets dans les temps à venir. Vos suggestions à propos de nouveaux sujets et vos retours d'informations en général sont bien entendu les bienvenus.

IV - Contenu

IV-A - Comment suivre

Le tutoriel

On décrit dans cette section où on peut télécharger le code source du tutoriel, de quels outils on a besoin et comment lancer les exemples.

IV-B - Module 1 : les bases

Le tutoriel

On introduit les concepts classiques de QML : composants et propriétés. On commence en montrant comment organiser des composants à partir de simples composants, comment utiliser des expressions JavaScript dans les valeurs des propriétés et comment accéder aux propriétés d'autres composants. On illustre ces concepts avec de simples exemples. On débute avec un rectangle bleu et le fait évoluer en une rangée de trois photos avec des titres. Cette simple visionneuse de photo va nous accompagner en tant qu'exemple à travers les différents modules.

Mots-clés : composants, organisation de composants, propriétés intégrées et personnalisées.

IV-C - Module 2 : les composants

Le tutoriel

On apprend à écrire nos propres composants personnalisés et comment les utiliser en code client. Dans la visionneuse de photos, on introduit un composant Photo et l'utilise dans le code de la visionneuse.

Mots-clés : composants personnalisés, alias de propriétés, id (identificateurs d'instance) de propriétés.

IV-D - Module 3 : les modules

Les bibliothèques de composants QML réutilisables sont appelées des modules. Dans ce module, on va montrer comment organiser les composants QML sous forme de modules et comment accéder à des composants en provenance de modules en code client. Les composants sont stockés en modules ou répertoires et sont rendus disponibles en code client à travers des déclarations d'importation. On explique en détail comment les déclarations d'importation comme *import Qt 4.7* ou *import "../dir"* fonctionnent.

IV-E - Module 4 : les layouts

Jusqu'ici, on a exposé des composants QML en spécifiant des positions x et y absolues. C'est lourd et compliqué à maintenir. Une meilleure solution consiste en les layouts, effectuant automatiquement le positionnement. QML fournit des layouts de lignes, colonnes, grilles et ancres. En utilisant notre exemple de visionneuse de photos, on explique les différentes possibilités de layouts.

IV-F - Module 5 : les entrées de souris et clavier

Notre visionneuse photo n'est pas tout à fait adaptée pour les utilisateurs pour le moment. Ils pourraient devoir changer le code QML pour entrer un titre, une description ou une date pour une photo. Par conséquent, la visionneuse photo devrait autoriser l'utilisateur à sélectionner une photo avec la souris ou avec les touches du clavier puis le laisser entrer le titre, la description et la date par le biais d'un formulaire. On découvre dans ce module comment gérer l'entrée clavier et souris dans QML.

IV-G - Module 6 : les vues de listes

Les vues de listes donnent une première vision des interfaces utilisateur fluides dans QML. Par défaut, les vues de listes sont basculantes et peuvent être poussées sur la gauche ou la droite par un mouvement du doigt ou par le curseur de la souris. Les vues de listes utilisent l'architecture modèle-vue de Qt et sont faites d'une vue, d'un représentant normal, d'un représentant surligneur et d'un modèle. On implémente ici un *cover flow* très simple pour notre visionneuse de photos.

IV-H - Module 7 : les états et les transitions

L'interaction entre l'utilisateur et le système et la transition d'un écran à l'autre sont habituellement modelées avec les machines à états. Chaque composant QML fournit des propriétés pour l'implémentation des machines à états. On utilise l'exemple de la visionneuse de photos pour illustrer les états et les transitions dans QML. La visionneuse de photos a trois écrans. Le premier écran montre la photo dans une liste. Dans cet écran, l'utilisateur peut choisir de voir ou bien d'éditer la photo sélectionnée en cliquant sur le bouton respectif de la photo elle-même. Cela mène l'utilisateur au deuxième écran, l'écran utilisé par une unique photo, ou bien au troisième, pour éditer l'information additionnelle de la photo. En cliquant sur un bouton de retour, l'utilisateur retourne sur le premier écran. Si on considère les écrans et les états, la description effectuée définit une machine à états avec des états et des transitions.

IV-I - Module 8 : les animations

Avec les animations, on va respirer encore plus de fluidité et de dynamisme dans les applications QML. On peut joindre les animations aux transitions des machines à états. Cela permet par exemple des effets de disparition lors de la transition d'un écran de photo à un autre. Les animations peuvent également être liées aux propriétés. Par exemple, la photo sélectionnée d'une liste de photos peut être graduellement rétrécie lorsqu'une autre photo est sélectionnée.

IV-J - Module 9 : l'écriture de nouveaux composants QML en Qt/C++

En utilisant l'exemple d'un simple lecteur de musique, on explique comment QML et Qt/C++ communiquent entre eux. On montre comment un composant QML peut accéder aux propriétés d'un objet Qt ou comment il peut appeler les slots de cet objet. Ce lecteur de musique peut jouer des sons depuis une liste de lecture. Il peut passer à la piste suivante ou précédente. Il peut mettre en pause ou relancer le son. Il peut en changer l'ordre dans la liste de lecture, répéter un son ou tous les sons de cette dernière. Ce lecteur de musique montre une barre dans une défilement où en est le son. Il montre également des informations à propos du son, comme l'artiste, le titre, l'album et l'année.

V - Remerciements

Merci à **Thibaut Cuvelier** et à **Claude Leloup** pour leur relecture !