Qt: Contexte, événements (signaux/slots) et propriétés

François Delobel

Master TechMed et Setsys, Université Clermont Auvergne

Plan

QObject et compilation

“Ça compile pas” ? N’avez vous pas...

- Ajouté des signaux, slots, ou propriétés dans une classe qui ne dérive pas de `QObject`.
- Fait un héritage *privé* en oubliant le mot clé `public` dans l’héritage.
- Oublié de mettre la macro `Q_OBJECT` au début de la classe.
- Quand on ajoute une macro `Q_OBJECT`, oublié de relancer un coup de `qmake`.
- Essayé de copier un `QObject` ou un de ses dérivés au lieu de passer (ou de retourner) des pointeurs.

QML

Votre composant ne s'affiche pas?

- Il a une taille nulle ?
- Il est de la même couleur que le fond?
- il est affiché *derrière* un autre élément ? Ils sont empilés dans l'ordre de déclaration.

Il n'a pas la bonne taille ou position ?

- Consultez la *sortie de l'application* pour vérifier les messages de QML.
- Ne donnez pas des instructions contradictoires (genre spécifier un `x` dans un `row`...).

Votre fenêtre ne s'affiche pas?

- Vous avez sans doute oublié de faire un `show()` dessus.

Communication C++ / QML

Erreurs fréquentes

- Essayer d'accéder à un *attribut* depuis QML, alors qu'on devrait en faire une *propriété*.
- Essayer d'appeler une *méthode* C++ sur un objet en ayant oublié le `Q_INVOKABLE` devant son prototype (les slots aussi sont appelables).
- Se compliquer la vie en construisant un modèle C++ alors qu'une liste de `QString` ou de `QObject` suffirait.
- Une propriété C++ vue en QML ne se met pas à jour quand le métier change? Êtes vous bien passé par le setter de la propriété qui envoie un signal de notification au lieu de taper l'attribut comme un goret?

Modèle C++

- Avez vous *vraiment* besoin d'un `AbstractListModel`? C'est inutile pour les listes qui ne varient pas.
- Vous ne voyez pas les rôles dans votre délégué? Avez vous bien correctement redéfini la méthode `roleNames()`?
- Votre insertion ne marche pas alors que vous avez redéfini `insertRows`? C'est sans doute que vous avez soit oublié d'appeler `beginInsertRow` et `endInsertRows` ou que vous n'avez pas bien calculé vos indices?
- Votre destruction ne marche pas alors que vous avez redéfini `removeRows`? C'est sans doute que vous avez soit oublié d'appeler `beginRemoveRow` et `endRemoveRows` ou que vous n'avez pas bien calculé vos indices?

Conception

- Votre métier contient des modèles? Ça sent le sapin... Votre métier doit contenir des conteneurs. Écrivez un modèle qui sert de façade au conteneur de votre métier (Par exemple en prenant un pointeur sur le conteneur) et chargez le avec les valeurs utiles le moment venu.
- Globalement, vous sous-utilisez les signaux et slots. On peut faire plein de choses avec une simple connexion!

C++

- `MaClass c();` ne déclare pas une instance de `MaClasse` mais une fonction qui ne prend rien en paramètre et renvoie une instance de `MaClasse`. Préférez `MaClass c;` ou la version C++14 des constructeurs homogènes: `MaClass c{}`.
- Le passage de paramètre par défaut, c'est par *copie* (et Qt n'aime pas les copieurs!). Les `QObject` ne sont pas copiables!

QML: attention aux dépendances

- Attention, QML gère des *dépendances* entre les *propriétés*: Quand une propriété change, QML recalcule tous les objets qui dépendent de cette propriété. QML ne fait cependant pas de magie et ne peut en particulier trouver des dépendances quand une fonction Javascript est appelée. Gardez donc vos expressions le plus simple possible (mais qui voudrait écrire `Label { text: myModel.getValue(index).name()}` au lieu de `Label { text: name }`?).