

Les pièges les plus courants de QThread

Par [Andy Shaw](#) - traducteur : [Guillaume Belz](#)  - Digia Qt

Date de publication : 30 mars 2012

Dernière mise à jour : 16 avril 2012

On va aborder les threads, dont on a tendance à abuser en général, ce qui peut conduire à des problèmes par la suite. Cela n'apportera peut-être pas d'informations pour beaucoup d'entre vous, si ce n'est la majorité, mais on rencontre encore régulièrement des personnes qui font des erreurs sur ces points.

N'hésitez pas à commenter cet article !

Commentez

I - L'article original.....	3
II - Problème n° 1 : ajouter des slots dans une classe dérivée de QThread et les appeler à partir d'objets depuis la fonction run() ?.....	4
III - Problème n° 2 : création d'un objet à l'intérieur de la fonction run() du thread et passer « this » comme parent.....	5
IV - Problème n° 3 : lancer une mise à jour d'un widget à l'intérieur de la fonction run() du thread.....	6
V - Conclusion.....	7
VI - Remerciements.....	8

I - L'article original

Cet article est une adaptation en langue française de **Qt Commercial Support Weekly #9: Common pitfalls with QThread**, d'Andy Shaw.

II - Problème n° 1 : ajouter des slots dans une classe dérivée de QThread et les appeler à partir d'objets depuis la fonction run() ?

Il s'agit d'une erreur facile à commettre, puisque l'on doit déjà dériver de **QThread** pour exécuter la fonction **run()**, qui est l'endroit où tout le code qui doit être exécuté dans le thread devrait aller. Cependant, si on ajoute des slots dans la classe dérivée de **QThread**, ils n'appartiennent pas au même thread que vous utilisez. Lorsque la fonction **run()** est invoquée, le code s'exécute à l'intérieur d'un autre thread et tous les objets créés dans le corps de cette fonction appartiennent à ce thread. Au contraire, l'objet **QThread** lui-même appartient au thread qui l'a effectivement créé, qui est généralement le thread principal.

Par conséquent, pour s'assurer que les slots sont invoqués dans le bon thread, on crée une classe dérivant de **QObject**, on y ajoute les slots en tant que membres de cette sous-classe, puis on crée une instance de la classe dérivée de **QObject** à l'intérieur de la fonction **run()**. Cela permet de garantir que les slots appartiennent au même thread.

III - Problème n° 2 : création d'un objet à l'intérieur de la fonction run() du thread et passer « this » comme parent

C'est une chose naturelle que de passer `this` comme parent à la création d'un widget ou d'un objet, de telle sorte qu'on ne doit pas s'inquiéter pour la hiérarchie parent-enfant et pour le nettoyage des objets. Cependant, cela pose un problème au sein de l'autre thread, car, dans ce contexte, `this` est l'objet **QThread** qui appartient à un thread différent. On peut toujours s'assurer qu'il y a une relation parent-enfant en passant un parent, mais ce devra être un objet créé ou appartenant au thread courant.

IV - Problème n° 3 : lancer une mise à jour d'un widget à l'intérieur de la fonction run() du thread

Cela peut sembler évident que cela est à éviter, c'est même indiqué dans la documentation de Qt, mais cela peut sortir de l'esprit des gens. Parfois, ce genre de chose peut sembler fonctionner et, par conséquent, être oublié jusqu'à ce que cela ne fonctionne plus. Cela peut être déclenché lors d'un changement de modèle ou même lorsque l'on associe un modèle à une vue. Dans certains cas, il peut être plus prudent de préparer quelque chose avant de le transmettre à l'autre thread avec **moveToThread()** et puis de lancer l'appel qui déclenche la mise à jour dans le thread principal. Il n'est cependant jamais sans risque de le faire directement. Cela pourrait fonctionner, mais c'est susceptible de poser des problèmes par la suite.

V - Conclusion

Ce sont les trois principaux problèmes à surveiller, mais d'autres choses doivent être surveillées (voir la documentation : **Le support des threads avec Qt**).

VI - Remerciements

Un grand merci à **dourouc05** pour sa relecture très attentive et ses conseils. Merci à **ClaudeLELOUP** pour sa relecture orthographique et à **djibril** pour le support pour la publication.