Déploiement d'applications sur Windows

par Kinji1

Date de publication : 09 février 2009

Dernière mise à jour :

Qt Quarterly est un journal électronique disponible exclusivement aux clients Qt. Chaque trimestre, nous envoyons un e-mail qui, nous l'espérons, ajoutera à votre expérience Qt, avec des articles de qualité écrits par des experts de Qt.

Developpez.com a reçu l'autorisation de Nokia afin de traduire ces articles.

I - Déploiement d'applications sur Windows

Par Andy Shaw

Cet article est la seconde partie d'une trilogie concernant le déploiement d'applications Qt sur différentes plateformes. Cette fois ci nous nous focaliserons sur Windows. Comme pour des applications sur Mac OS X, déployer des applications sur Windows ne nécessite pas de programmation C++. Tout ce dont vous avez besoin est de compiler Qt et votre application en mode release, et de suivre les procédures décrites dans cet article. Nous exposerons ces procédures en déployant l'application *showimg* disponible dans le répertoire *examples* de Qt.

I-A - Liaison statique

Si vous voulez faire simple en ne déployant qu'un seul fichier, c'est à dire un exécutable indépendant, vous devez alors tout compiler de manière statique. Comme pour la liaison statique sur Mac OS X, les avantages sont que l'application se chargera et s'exécutera plus rapidement puisque tous les symboles sont dans l'exécutable. Les incovénients sont une taille plus grande de celui-ci, et le fait que vous ne pouvez pas utiliser de plugins avec une application liée statiquement.

Avant de pouvoir compiler notre application, nous devons nous assurer que Qt est compilé statiquement. Pour le faire, ouvrez une invite de commandes et tapez les lignes suivantes :

```
cd %QTDIR%
configure -static <autres options dont vous avez besoin>
```

N'oubliez pas de spécifier les autres options dont vous avez besoin, comme le support des threads, en arguments de configure. Une fois que configure a terminé, tapez la commande suivante :

```
make sub-src
```

Ceci va compiler Qt de manière statique en mode release (1) . Premièrement nous devons aller dans le répertoire qui contient l'application à déployer.

```
cd examples\showimg
```

Nous devons lancer qmake pour créer un nouveau Makefile pour l'application, et réaliser une compilation propre pour créer un exécutable lié statiquement.

```
qmake showimg.pro
make clean
make
```

Maintenant, pourvu que tout ait été compilé et lié sans erreurs, nous devrions avoir un fichier *showimg.exe* prêt à être déployé. Une méthode simple pour vérifier que l'application peut vraiment s'exécuter de manière indépendante est de la copier sur un ordinateur sans Qt et sans applications Qt d'installées et de l'exécuter.

I-B - Bibliothèques partagées

Si vous ne voulez pas utiliser la compilation statique parce que vous voulez utiliser des plugins ou parce que vous voulez utiliser les mêmes DLL Qt pour une famille d'applications, alors la solution est d'utiliser une bibliothèque partagée.

Comme pour la liaison statique, avant de compiler notre application nous devons nous assurer que Qt est compilé comme une bibliothèque partagée. Pour cela ouvrez une invite de commandes et tapez ce qui suit :

```
cd %QTDIR%
configure -shared <autres options dont vous avez besoin>
```

N'oubliez pas de spécifier les autres options dont vous avez besoin, comme le support des threads, en arguments de configure. Puisque nous utilisons encore l'application *showimg* comme exemple nous devons aussi spécifier *-plugin-imgfmt-jpeg* comme option, pour qu'il y ait au moins un plugin. Une fois que configure a terminé, tapez la commande suivante :

```
make sub-plugins
```

Cela va compiler Qt en une bibliothèque partagée, en mode release, et compiler le plugin de formattage d'images JPEG. Lorsque Qt a fini de compilé nous pouvons compiler l'application *showimg*. Premièrement nous devons nous rendre dans le répertoire de l'application à déployer :

```
cd ..\examples\showimg
```

Maintenant nous lançons qmake pour créer un nouveau Makefile pour l'application et nous réalisons une compilation propre pour créer l'exécutable lié dynamiquement.

```
qmake showimg.pro
make clean
make
```

Si tout a été compilé et lié sans erreurs nous avons un fichier *showimg.exe*. Pour le déployer nous devons nous assurer de copier la DLL Qt et l'exécutable dans un répertoire approprié (Nous traiterons le cas du plugin JPEG un peu plus loin). Premièrement nous allons vérifier que l'application fonctionnera dans un environnement déployé. Copiez l'exécutable et la DLL Qt sur une machine qui n'a ni Qt, ni d'application Qt installée, ou si vous voulez faire le test sur la machine de compilation, supprimez la variable d'environnement QTDIR et retirez %QTDIR%\bin de la variable d'environnement PATH.

Nous devrions avoir maintenant l'exécutable *showimg.exe* et la DLL Qt (p.ex. qt331.dll, ou si vous avez configuré le support des threads qt-mt331.dll) dans un répertoire. Dans une invite de commande, rendez vous dans ce répertoire et essayez de lancer le programme.

Si l'application s'exécute sans problèmes alors nous avons réussi à faire une version liée dynamiquement de l'application *showimg*. Si vous tentez d'ouvrir une image PNG avec l'application, celle-ci devrait être affichée correctement; mais si vous essayez d'ouvrir une image JPEG, rien ne se passera puisque nous n'avons pas encore déployé le plugin JPEG avec l'application.

I-C - Plugins

Les plugins fonctionnent différenmment des DLLs classiques, nous ne pouvons donc pas simplement les copier dans le même répertoire que notre exécutable comme nous l'avons fait avec la DLL de Qt. Les applications Qt recherchent les plugins dans un sous-répertoire de celui de l'application. Par exemple, un plugin pour un format d'image doit se trouver dans le sous-répertoire *imageformats* et le plugin d'un pilote SQL doit être dans le sous-répertoire *sqldrivers*.

Ainsi, pour rendre le plugin JPEG disponible pour notre application *showimg*, nous avons seulement à nous rendre dans le répertoire de l'application et à copier la DLL appropriée :

```
mkdir imageformats
copy %QTDIR%\plugins\imageformats\qjpeg100.dll imageformats
```

(Si vous aviez supprimé la variable d'environnement QTDIR, vous devrez taper le chemin absolu vers le répertoire de Qt au lieu d'utiliser %QTDIR%.) Si vous lancez l'application *showimg* et ouvrez une image JPEG, celle-ci devrait se charger correctement.

L'un des avantages de l'utilisation des plugins est qu'ils peuvent être facilement mis à la disposition de toute une famille d'applications. Par exemple, si nous avions besoin de plusieurs applications nécessitant le support des images JPEG, il serait plus pratique d'utiliser un répertoire de plugins commun. Nous pourrions alors choisir de placer tous nos plugins dans le répertoire "C:\QtPlugins". Si nous utilisons un répertoire personnalisé, nous devons tout de même conserver les sous-répertoires pour les différents types de plugins, notre plugin pour les images JPEG irait donc dans "C:\QtPlugins\imageformats". Pour que notre application sache où se trouve ce répertoire de plugins additionnels, nous devons ajouter la ligne suivante, avant que l'application n'ait besoin de charger les plugins.

```
qApp->addLibraryPath("C:\\QtPlugins");
```

Il est souvient plus pratique d'ajouter ce chemin à la fonction main() de l'application, juste après la création de l'objet QApplication. Une fois le chemin ajouté, l'application y cherchera les plugins en plus de regarder dans ses propres sous-répertoires. Il est possible d'ajouter autant de chemins que l'on souhaite.

I-D - Le problème du runtime C

Parfois vous devrez distribuer la bibliothèque de runtime C avec votre application. La plupart des versions de Windows ont déjà cette bibliothèque installée, mais si vous utilisez un Visual Studio moderne, ce ne sera peut-être pas le cas. Vous pouvez vérifier à quelles bibliothèques votre application est liée grâce à l'outil *depends*. Pour s'en servir, vous devez simplement le lancer comme cela :

```
depends <exécutable de l'application>
```

Cela vous donnera une liste des bibliothèques dont votre application dépend ainsi que d'autres informations. Si vous regardez dans la partie en haut à gauche, vous pouvez voir la liste des bibliothèques. Celles que vous voudrez vérifier commencent par "MSVC"; pour être certain que vous avez les bonnes bibliothèques disponibles, assurez-vous de copier celles-ci avec votre exécutable.

I-E - Conclusion

Cet article a couvert les points clés du déploiement d'applications Qt sur Windows. Nous n'avons pas mentionné les versions spécifiques de Windows puisqu'il n'y en avait pas besoin; les applications Qt compilées sur une version de Windows fonctionneront sur les autres sans modification.

1 : Nous avons utiliser make dans tous les exemples, mais si vous utilisez Microsoft Visual C++, vous devez utilisez nmake à la place.