Projet de Langage Objet Avancé : Gestionnaire de Contacts

Bama Abougou et Davina Rungen

26 Mars 2014

Table des matières

Chapitre 1

Introduction

L'objectif de ce projet est de réaliser un gestionnaire de contact en utilisant Qt et la stl.

1.1 Spécification

Voici les différentes spécifications qui ont été données :
- Concevoir des classes permettant d'implémenter une liste de contacts et les champs qui permettront de les caractériser
- Concevoir une interface graphique en utilisant Qt
- Concevoir une interface graphique en utilisant Qt
- Concevoir la documentation du projet en utilisant Doxygen.

Le projet se divise en trois parties :
- La conception du modèle : trouver une hiérarchie pratique et efficace
- Implémentation des fonctionnalités : rendre le projet fonctionnel
- La conception de l'interface graphique : donner à l'utilisateur accès à toutes les fonctionnalités

1.2 Fichier et dossier du projet

- Pro.pro est le fichier de description du projet (our qtcreator)
- main.cpp contient la fonction main du projet
- modele/ et vue/ sont les dossiers où sont mises les sources
- rapport/ est le dossier où se trouve ce rapport

1.3 Projet réalisé avec Qt5.2

Ce projet a été réalisé avec Qt5.2 , une version téléchargeable est disponible sous le lien suivant : `http://download.qt-project.org/official_releases/qt/5.2/5.2.1/qt-opensource-linux-x64-5.2.1.run`

1.4 Fichier contacts d'exemple

J'ai réalisé un fichier de contacts pour l'exemple, afin que les images soient bien chargé il faut mettre le fichier "british.xml" ainsi que le dossier image_chat/ dans le même dossier que l'exécutable (par exemple si les shadow build sont activés dans le dossier build-pro... au dessus du dossier où se trouve le code source)

Chapitre 2

Hiérarchie des classes

2.1 Modèle

Le modèle consiste en une classe Contacts tout en haut de la hiérarchie, qui implémente une liste de Contact.

La classe Contact contient une liste de Champ et un nom. La classe Champ possède un nom , Les classes ListeChamps, Texte et Enum héritent de la classe champ.ListeChamps à 4 fille : Adresse, Email, Nom et Tel

Les classes MyModelContacts, MyModelListeChamps sont des modèles Qt qui permettent à la vue de gérer correctement l'affichage (qui permet de modifier dynamiquement l'affichage).

2.1.1 Choix de Liste

l'intérêt de la classe ListeChamps est qu'elle permet à l'utilisateur de choisir d'ajouter autant de champs qu'il veut. Elle permet aussi de faire une implémentation unique entre les champs composés (adresse, email, tel) et la liste des champs d'un contact. c'est beaucoup plus général que de fixer les champs directement dans le code.Par exemples l'utilisateur peut avoir plusieurs champs adresse :

- Adresse de la maison
- Adresse de la deuxième maison
- Adresse de l'hotel préféré
- ...

Par exemple avec le téléphone aussi :

- Numéro de tel privé
- Numéro de tel professionnel
- ...

Grâce à la classe ListeChamps on ne contraint pas du tout la façon dont est édité chaque champ et grâce aux delegate d'édition codés (listeChampsEdit imageEdit, ...) cela permet de de concentrer sur le code des modèles et des vues des champs plutôt que leur organisation dans un contact qui est faite automatiquement et mieux par Qt.

2.2 Vue

La vue est composée d'une MainWindow et de cinq widgets ContactView,NouveauContact, ListeChampsEdit, ImageEdit et EnumEdit qui servent à l'affichage et l'édition des champs et des contact dans la fenêtre. On y trouve également les boîtes de dialogue AjouterChamp qui permettent à l'utilisateur d'entrée des informations lorsqu'il veut ajouter un champ.

Chapitre 3

Fonctionnalités

Voici la liste des fonctions implémentées :
— Documentation du code et utilisation de doxygen pour générer la doc
— Conception des classes permettant de représenter les contacts et les champs les décrivant.
— L'affichage de la liste des contacts
— L'ajout/suppression de contacts dans la liste
— L'affichage des détails d'un contact lors d'un clic sur celui-ci dans la liste
— La modifications des champs d'un contact
— L'implémentation des champ
— L'ajout/suppression de champs pour un contact
— La génération de champs par défaut lors de la création d'un contact
— La sauvegarde/chargement de la liste des contact en utilisant un format XML
—

3.1 Enregistrer/Sauvegarder

3.1.1 XML

Quant au format XML, il exporte les champs et les contacts en utilisant toString(), du coup la méthode toXML() n'a besoin d'être implémentée que dans Contact et Champ. Syntaxiquement, le format XML se résume à une structure semblable à celle du modèle : une suite de ¡Contact¿ ... ¡/Contact¿ avec des balise dont le nom est type du champ et qui possède un attribut nomChamp qui contient le nom de ce champ, et dont le contenu est la valeur du champ.

Chapitre 4

Interface Graphique

L'interface graphique contient une liste de contact à gauche et l'affichage ou bien l'édition des contacts à droite. On peut aussi sélectionner plusieurs boutons dans le menu afin d'afficher l'ouverture et l'enregistrement de fichier.

La vue des contacts est composé d'une liste de champ accompagné de leur valeur et pour certains champs

En mode édition chaque champ est modifiable par un widget particulier : par exemple l'édition des adresses peut se faire ou bien via une ligne d'édition de texte ou bien via un tableau afin d'éditer chaque sous-champ séparé. Les autres type de champs sont eux aussi être édités par des widgets dédiés.

L'interface graphique se compose d'une barre de menue et d'un layout central. Toutes les fonctionnalités sont accessible via la barre de menu. Le layout central est divisé en trois : la liste de contacts, l'affichage du contact et l'édition du contact.Ces deux derniers ne sont jamais visible simultanément. Option des menus : Ajouter champ, créer un nouvyoupi!!! eau contact, supprimer un contact, sauvegarder , charger

Chapitre 5

Conclusion

5.1 Résultat

Une partie des fonctionnalités demandées sont réalisés. Le modèle implémenté permet facilement d'intégrer d'autres types de champ en créant d'autres classes héritant de Champ.

5.1.1 Avancement personnel

Ce projet a permis d'apprendre mieux la partie modèle/vue de Qt. Il nous a permis de mieux connaître la librairie Qt et qtcreator.