TP Qt : La suite

(1.5 scéances)

Yannick Kergosien

Objectif du TP : *se familiariser avec QtSQL et de finir l'architecture MVC du projet.*
Lien utile : `http://doc.qt.io/qt-5/classes.html`.

1 Préambule

Pour ce projet, vous allez utiliser SQLite (les pilotes sont déjà installés avec Qt) et une base de données en mémoire stockée dans le fichier "base_tmp.sqli". Cette base sera créée grâce à l'appel de la fonction statique "Creation_BD" de la classe "c_init_bd".
Cette base contient 5 tables :

- Une table TClient avec 11 colonnes :
  - Id : clé primaire, entier,
  - Nom : le nom, texte,
  - Prénom : le prénom, texte,
  - Adresse : l'adresse, texte,
  - Ville : la ville, texte,
  - CP : le code postal, texte,
  - Commentaire : le commentaire, texte,
  - Tel : le numéro de téléphone, entier,
  - DateRdv : le jour du rendez-vous, date,
  - DureeRdv : la durée du rendez-vous, entier,
  - Priorite : la priorité, entier.
- Une table TRessource avec 4 colonnes :
  - Id : clé primaire, entier,
  - Nom : le nom, texte,
  - Prénom : le prénom, texte,
  - IdType : clé étrangère, entier.
- Une table TRdv avec 3 colonnes :
  - Id : clé primaire, entier,
  - IdClient : clé étrangère, entier,
  - IdRessource : clé étrangère, entier.
- Une table TType avec 2 colonnes :
  - Id : clé primaire, entier,
  - Label : dénomination, texte.
- Une table TCompte avec 4 colonnes :
  - Id : clé primaire, entier,
  - IdRessource : clé étrangère, entier,
  - Login : le login, texte,
  - MdP : le mot de passe, texte.

Cette base de données contient quelques enregistrements, vous pouvez en ajouter. Attention, cette base sera supprimée et recrée à chaque appel de la fonction statique de création. Afin de manipuler la base de données, utilisez les classes QSqlDataBase et QSqlQuery ainsi que de requêtes SQL.

2 MVC

Maintenant que la base de données est en place, vous pouvez accomplir la partie donnée (et contrôleur) du projet et la relier aux vues. Afin de respecter une architecture MVC, les points suivants sont conseillés :

- Créez des classes modèles (Client, Ressource, etc.) qui contiendront les données en mémoire,
- Créez une classe réalisant l'interface avec la base de données,
- Utilisez les éléments que Qt propose pour ce type d'architecture (cf. QTableView, QSqlTableModel, etc.),
- Vous pouvez cependant intégrer la partie contrôleur aux modèles, par exemple créer une méthode de vérification des données dans la classe client avant de l'ajouter dans la base de données.

N'hésitez pas à poser des questions en cas de doute, pour des conseils, pour s'assurer de la validité de votre architecture, etc.