

Accès et manipulation d'une base de données relationnelle

1. Rappel

- Une base de données (BD) est un « conteneur » stockant des données de natures différentes (numériques, textes, mots, dates, heures, images, vidéos, etc.) et plus ou moins reliées entre elles. Le stockage des données se fait de façon organisée et cohérente.
- Une base de données relationnelle (BDR) est une base de données qui respecte un schéma relationnel.
- Une BDR repose sur un modèle relationnel qui est une collection de :

- *Table* : chaque table est identifiée par un nom et contient plusieurs champs. Une table regroupe les informations d'une entité.

Exemple : une personne, un produit, une commande, une réservation, etc.

- *Attribut* (champ) : chaque attribut a un nom et un type de données (texte, nombre, date, heure, images, etc.).

Exemple : Pour la table « Personne », on peut avoir les attributs « Nom », « Prénom », « Adresse », etc.

- *Clé primaire* : Dans les modèles relationnels, la clé primaire est un attribut dont la valeur est différente pour chaque enregistrement de la table. Ce qui permet de retrouver un enregistrement unique.
- *Clé étrangère* : un attribut contenant une référence vers une valeur présentant une clé primaire d'une autre table.

- *Enregistrement* : une donnée composée qui comporte les données de chaque attribut de la table.

Exemple : Dans la table « Personne », on peut avoir plusieurs enregistrements chacun représentant les données d'une personne.

Enregistrement1 : Linus Torvalds 1012, chemin de la belleville, Finlande

Enregistrement2 : Richard Stallman 1213, chemin de la belleprovince, USA


- *Relations* (liaisons) : les tables sont reliées entre elles via des relations. Une relation relie la clé primaire d'une table à la clé étrangère d'une autre table.

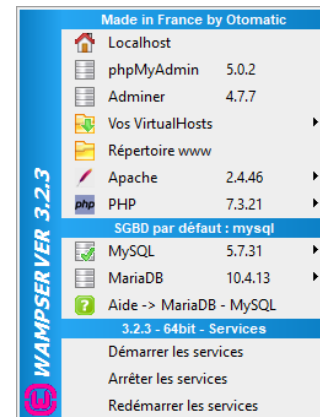
- Un SGBD (Système de Gestion de Base de Données) est un logiciel qui permet de créer, modifier, détruire, gérer et interroger concurremment plusieurs bases de données.
- Le but de l'ensemble étant l'administration et la manipulation facile des données, d'éviter la redondance des données et d'assurer la cohérence des données.
- Le SGBD reçoit les demandes de manipulation de contenu et effectue les opérations nécessaires sur les fichiers de la BD.
- Les SGBD les plus répandus sont : MySQL, PostgreSQL, Oracle, Microsoft SQL Server, IBM DB2, etc.
- Nous utiliserons le serveur de SGBD « MySQL » qui utilise le langage SQL (Structured Query Language, langage de requête structurée). Pour faire tourner notre base de données, nous utiliserons la plateforme de développement Web « WampServer ». Ce dernier contient l'application Web « phpMyAdmin » pour l'administration Web de MySQL.

2. Installation et utilisation de WampServer

- Wamp (acronyme de Windows, Apache, MySQL et PHP) est une plateforme gratuite, disponible sous licence GPL pouvant être installé sous Windows.
- Il est possible de télécharger WampServer de l'adresse suivante :

<https://sourceforge.net/projects/wampserver/files/>

- Après l'installation de WampServer, son lancement démarre 3 serveurs qui peuvent être gérés et administrés au travers de l'icône  dans la barre d'état système (dans la partie droite de la barre des tâches de Windows). Un clic sur cette icône permet d'afficher les différents services proposés par cette plateforme.
- Cliquez sur « phpMyAdmin ». Nom d'utilisateur : root, Pas de mot de passe et choix du serveur : MySQL. Cliquez ensuite sur le bouton « Exécuter ».



3. Création d'une base de données

De nos jours, la plupart des applications utilisent des BDs pour le stockage des informations. L'apparition des BDs et des SGBDs a facilité grandement la gestion des données des applications de différentes natures.

Dans le chapitre précédent, nous avons utilisé la bibliothèque externe « Newtonsoft.json » pour que notre application de gestion de participants soit capable de gérer des données (par lecture et écriture) à partir d'un fichier « .json ». Afin d'améliorer son efficacité dans la gestion des données et bénéficier pleinement des avantages des modèles relationnels, nous allons faire évoluer cette application en utilisant cette fois un SGBD Relationnel.

- Commençons par la création de la base de données dans l'interface de PhpMyAdmin.

- Créez une nouvelle BD et nommez-la « gestionparticipants ».
- Créez une nouvelle table « participant » dans la BD contenant les 6 colonnes suivantes :
 - « Matricule » de type « INT » et représente la clé primaire de la table.
 - « Prenom », « Nom », « Niveau » et « Email » de type « VARCHAR » et ne doivent pas être « NULL ».
 - « Genre » de type « CHAR ».
 - Moteur de stockage: InnoDB
- Pour insérer les données d'un participant, on peut utiliser une requête d'insertion comme la requête suivante :

```
INSERT INTO Participant (Matricule, Prenom, Nom, Genre, Niveau, Email)
VALUES (11111, 'Imelda', 'Helen', 'F', 'Débutant', 'HelenIm@gmail.com');
```
- En effet, le fichier « infosParticipants.sql » contient les requêtes SQL permettant l'insertion des données des participants dans la table « participant ». Importez ces données dans la table Participant.

Vérifiez dans l'onglet « Parcourir » que les données ont bien été ajoutées dans la table « participant ».

4. Connection d'une application C# à une BD MySQL

- Nous allons continuer avec la même application de gestion des participants développée dans le chapitre précédant « Intégration et utilisation de bibliothèques externes ». On veut l'améliorer en utilisant une base de données MySQL comme source de données.
- Étant donné que nous n'allons plus utiliser le package « Newtonsoft.Json » dans cette application, nous allons le désinstaller à partir du gestionnaire de packages externe « NuGet ». Plusieurs erreurs vont apparaître dans le projet dues à la suppression du package. Mettez en commentaire toutes les lignes qui affichent des erreurs.
- Pour pouvoir connecter notre application à la BD « gestionparticipants », installez le package « MySql.Data » avec le gestionnaire NuGet dans le projet.
- Dans la configuration actuelle de notre application, la classe « Participant » contient la méthode « **ChargerFichier** » permettant la désérialisation du fichier « fParticipants.json ». L'appel de cette méthode se fait dans le constructeur de la classe du contrôle d'utilisateur « UCGestionPart ».

Dans la classe « Participant » :

- Enlevez l'instruction de déclaration de la chaîne de caractères du chemin d'accès au fichier de données « NOM_FICHIER ».
- Supprimez la méthode « ChargerFichier » et mettez en commentaire tous ses appels.

- Dans le code-behind de la classe « UCGestionPart » :
 - Ajoutez les utilisations d'espaces de noms suivants :

```
using System.Data;  
using MySql.Data.MySqlClient;
```
 - Le remplissage du « DataGrid » par les données se ferait aussi à partir de la collection observable (ObservableCollection) « participants » (utilisée dans la solution précédente).
 - Dans le constructeur de la classe « UCGestionPart » et après l'instruction qui précise la source de données des items du DataGrid, définissez le contexte de données sur cette classe et appelez la méthode « connecterBD » qui permettra de remplir le DataGrid avec les données des participants à partir de la Base de données.
 - Dans la même classe, définissez la méthode « ConnecterBD » permettant de connecter l'application C# à la base de données « gestionparticipants ».

5. Mise à jour d'une base de données

5.1. Ajout d'un nouvel enregistrement

- La classe du « Contrôle Utilisateur » « UCAjoutPart » contient la méthode « Click_Valider » qui sur clic sur le bouton « Valider » permet d'ajouter les informations saisies pour un nouveau participant.
 - Cette méthode commence par l'insertion dans la liste « participants » des informations du nouveau participant. Récupérez dans des variables les valeurs saisies et sélectionnées par l'utilisateur pour le nouveau participant.
 - Ajoutez le nouveau participant dans la collection observable « participants ».
 - Appelez la méthode « InsérerNouvPartBD » (à définir plus tard).
 - Pour mettre à jour le fichier « .json », la méthode « Click_Valider » contient à sa fin un appel de la méthode « RecrireFichier » de la classe « MainWindow ». Supprimez cet appel.
- Toujours dans la classe « UCAjoutPart », définissez la méthode « InsérerNouvPartBD » pour insérer le nouveau participant dans la base de données.
- Faites un test d'insertion d'un participant. En cliquant sur le bouton « Valider » de l'interface utilisateur « UCAjoutPart », les informations saisies pour ce nouveau participant seront ajoutées dans la base de données et le DataGrid est mis à jour en affichant le nouveau contenu dans une nouvelle ligne.

5.2. Modification d'un enregistrement

- Dans le code-behind de la classe « **UCGestionPart** », la méthode « **BtnModifierPart_Click** » permet actuellement de mettre à jour le fichier « .json » en faisant appel à la méthode « ReecrireFichier ». On veut maintenant que les modifications des informations d'un participant faites dans le DataGrid soient sauvegardées dans la BD.

Mais, avant d'apporter des modifications sur une ligne du DataGrid, on devrait avoir en mémoire les anciennes valeurs des champs d'une ligne. Nous devons créer 5 variables publics pour contenir les valeurs de la ligne sélectionnée dans le DataGrid et une méthode « **Matricule_dbClick** » qui sera déclenchée en double-cliquant sur une ligne du DataGrid.

- Dans la méthode « **BtnModifierPart_Click** » :
 - Enlevez l'appel de la méthode « **ReecrireFichier** ».
 - Étant donné que le matricule est une clé primaire, et pour éviter les doublons potentiels dans ce champ à la suite d'une modification, on ne permet pas la modification du champ « **Matricule** ». On vérifie donc que l'utilisateur n'a pas changé la valeur de ce champ.
 - Quand la modification est acceptée, on ajoute les instructions permettant de se connecter à la BD pour effectuer la modification des données d'un participant.

5.3. *Suppression d'un enregistrement*

- Pour la suppression des informations d'un participant, on sélectionne une ligne du DataGrid « **dgParticipant** » puis on clique sur le bouton « **Supprimer** ».
- Supprimez les deux instructions de la méthode « **BtnSupprimerPart_Click** » : celle de suppression à partir de la liste observable « **participants** » et celle de la réécriture du fichier « .json » sans les données du participant supprimé.
- Ajoutez le code permettant d'accéder à la base de données pour supprimer l'enregistrement correspondant à la ligne sélectionnée du participant à supprimer. Ajoutez aussi l'instruction permettant de supprimer le participant sélectionné de la liste de participants.