Concepteur Développeur en Informatique

Module 1 Développer l'interface d'une application informatique

Composants accès données	Séance	S01	Activité	A-002

Cette activité a pour objectif de vous initier à la programmation d'applications Windows Forms avec accès aux données en mode connecté.

Sommaire de l'activité proposée :

1 Poursuite de la découverte d'ADO Net 2

2 Travail à Réaliser 3

2.1 Création de la base de données et tests.....3

2.2 Authentification de l'utilisateur 4

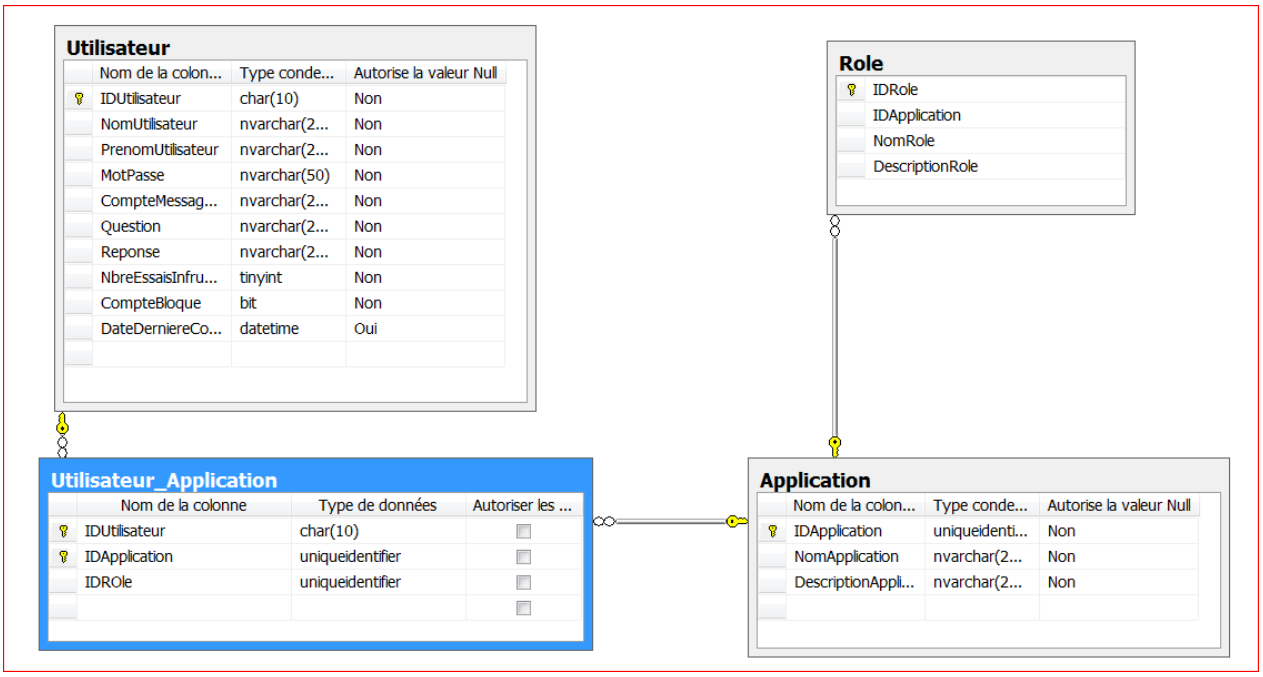
1 Poursuite de la découverte d'ADO Net

Au cours de ce deuxième atelier, vous allez poursuivre la mise en œuvre les techniques abordées précédemment avec la programmation des composants d'accès aux données associés à des procédures stockées.

Nous allons implémenter le formulaire initial d'authentification d'un utilisateur avec une base de données permettant le contrôle d'accès des utilisateurs aux applicatifs de l'entreprise.

Un premier prototype de cette base vous est communiqué. Vous pourriez vous inspirer de ce modèle dans vos développements futurs afin de gérer les accès des utilisateurs à vos applications.

Sachez toutefois qu'il existe aujourd'hui des solutions proposées par Microsoft en standard pour la gestion des utilisateurs par formulaire. Il est peut-être inutile de réinventer la roue et la gestion des authentifications via un annuaire d'entreprise ou de comptes de domaine….
Nous verrons la mise en œuvre de la gestion des accès par formulaire lors du développement d'applications Web.



Ce modèle traduit les règles d'organisation et de gestion suivantes :

1. Un utilisateur accède à une application au travers d'un rôle d'application.
2. Le rôle d'application permet de déterminer quels sont les droits accordés à l'utilisateur membre de ce rôle.
3. Un rôle est toujours associé à une et une seule application.
4. Un utilisateur est membre d'un seul rôle pour une même application.

Mais l'utilisateur Vincent Bost peut être inscrit dans le rôle Administrateur de l'application AFPA.Net et inscrit dans le rôle Formateur de l'application AFPA.Agora.
Ce modèle permet d'assurer la centralisation et l'unicité des données de l'utilisateur.

Le choix de comptabilisation des essais infructueux est discutable. Un utilisateur sera bloqué s'il a tenté de se connecter à 3 reprises sans succès à une application quelconque.

Ce modèle est incomplet. Il faudrait développer les éléments permettant de gérer les droits d'accès aux fonctionnalités d'une application par Rôle puis éventuellement par Utilisateur.

A noter : Vous observerez l'utilisation d'un nouveau type de données UniqueIdentifier qui représente un type GUI (Global Unique Identifier) bien connu des développeurs d'application.

Ce type de données est stocké sur 16 octets.

Il existe deux manières d'affecter une valeur initiale à une colonne ou à une variable locale du type de données uniqueidentifier :

- en utilisant la fonction NEWID(),
- en convertissant une constante de chaîne de la forme xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx, où chaque x est un chiffre hexadécimal compris dans la plage 0-9 ou a-f.
  Par exemple, 6F9619FF-8B86-D011-B42D-00C04FC964FF est une valeur uniqueidentifier valide. Les valeurs sont passées en chaine et converties automatiquement par le SGBDR dès lors que le format est respecté.

2 Travail à Réaliser

Vous devez finaliser le module d'authentification d'un utilisateur.
Sont mis à votre disposition sur le site de la formation :

- Le script de création des tables et procédures stockées de l'application sécurité ainsi qu'un jeu d'essai.
- Le fichier compressé des objets sql qui permettent de tester vos procédures stockées
- Un projet de type WindowsForms comportant une fenêtre principale MDI pour l'administration de la sécurité et une fenêtre fille pour la gestion de l'authentification.

2.1 Création de la base de données et tests

Créez une nouvelle base de données Securite sur votre ordinateur.
Définissez la base créée comme base courante.
Exécutez le script CreationObjetsSecurite.sql

Son exécution procède à la création :

- des objets de la base de données
- d'un jeu d'essai minimaliste.

2.2 Authentification de l'utilisateur

Installez le projet Windows après l'avoir téléchargé.
Le formulaire de connexion est affiché lors du premier chargement de la fenêtre MDI.

Dans cette application, vous devrez mettre en œuvre les mécanismes :

- De définition et d'exécution d'une procédure stockée
- D'exploitation des résultats de l'exécution d'une procédure stockée.

La procédure stockée à utiliser est la procédure psUtilisateur_Authentifier mise à votre disposition.
Vous devez vous assurer que celle-ci fonctionne correctement en procédant aux tests unitaires nécessaires en conséquence.

2.2.1 Coder la fonction authentification

Vous devez programmer le corps de la fonction **controlerInfosConnexion().**

Ne stockez pas vos chaines de connexion dans le code de votre application mais externalisez-la afin de pouvoir la modifier lors de la mise en exploitation sans devoir recompiler votre application.

Pour créer la liste des paramètres, 3 approches sont disponibles :

1. Déclarer les paramètres et les ajouter manuellement à la liste des paramètres
2. Utiliser le composant GenerateurAdo mis à votre disposition
3. Utiliser la méthode statique **DeriveParameters** de la classe **SQLCommandBuilder** pour dériver les paramètres de la procédure stockée

Si vous avez utilisez l'approche 3, vous devrez éventuellement modifier la direction des paramètres définis exclusivement en sortie au niveau de la procédure stockée. En effet, ils sont, au niveau de la commande, définis en input/outpout.
Pour accéder à un paramètre de la collection, utilisez l'indexeur sur le nom du paramètre ['@'NomParametre''] ou ''@NomParametre'' correspond au nom donné à la variable qui fait référence à l'argument de la procédure stockée.

```
SqlCommand cmd;
SqlConnection con;

0 références
private void deriverParametres()
{
    // Préparation de la commande

    cmd = new SqlCommand();
    con = new SqlConnection(Properties.Settings.Default.ComptoirAnglaisConnectionString);
    cmd.Connection = con;
    cmd.CommandType = CommandType.StoredProcedure;
    cmd.CommandText = "NomProcedureStockee";

    // Appel de la méthode statique pour dériver les paramètres
    // Le serveur est interrogé et la liste parameters de l'objet
    // de type command générée

    SqlCommandBuilder.DeriveParameters(cmd);

    // Modifier les paramètres en sortie uniquement
    // le composant les crée en entrée / sortie par défaut
    cmd.Parameters["@NomParametre"].Direction = ParameterDirection.Output;

}
```

A noter :

Alimentation des paramètres de la procédure stockée :

- Le nom de l'application doit être stocké comme paramètre de l'application.

Exploitation des résultats de l'exécution de la procédure stockée.
Utiliser l'énumération pour traiter les valeurs de retour de l'exécution de la procédure stockée.