

Développement en c#.NET

Ing. Meryem OUARRACHI

Plan du module

Langage C#

- ☐ L'environnement .Net
- ☐ Initiation à la programmation C#
- ☐ Programmation Orienté Objet C#

Programmation avancée en .Net ,C#

- ☐ Programmation distribuée
- ☐ Gestion de base de donnée
- ☐ Application WPF

Chapitre 4:

Gestion de Base de donnée en .Net

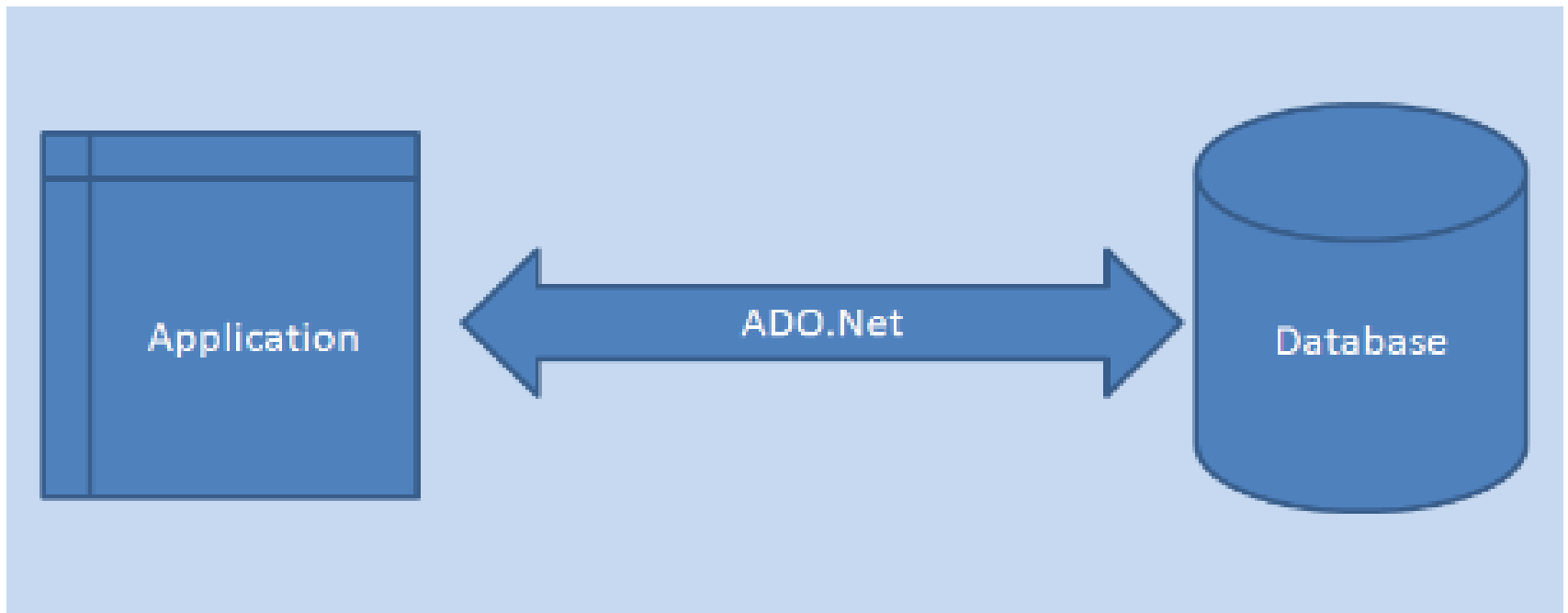
Plan du chapitre

- ❑ Mode connecté
- ❑ Mode déconnecté
- ❑ Linq to sql
- ❑ Reporting avec Crystal report

Architecture du framework

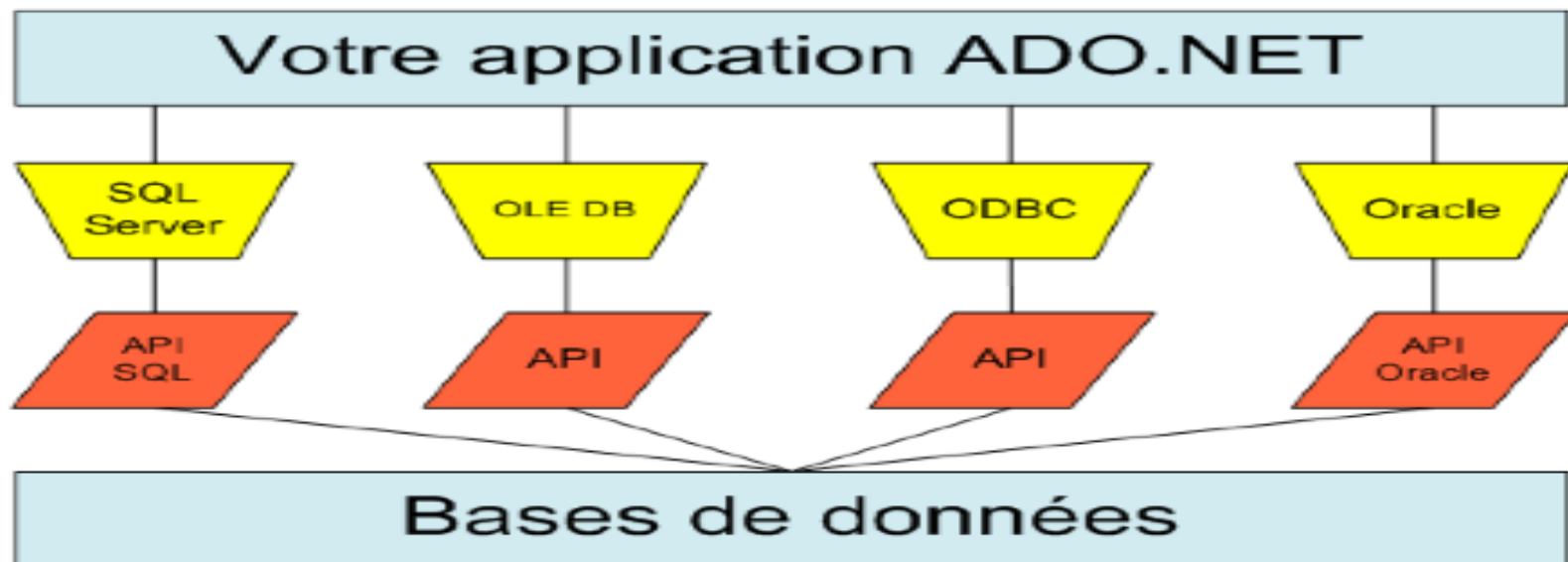


- ADO est un ensemble de classes qui vont nous permettre d'accéder aux données.



Les fournisseurs de données

- Chaque fournisseur de données permet la communication avec un type de base de données
- Ces fournisseurs se trouvent dans l'espace de nom **System.Data**



Architecture ADO

- **Mode connecté:** le client reste en connexion permanente avec le serveur de la base de donnée; Il la fermera plus tard.Ceci grâce à **DataReader**.
- **Mode déconnecté:** le client se connecte pendant un court laps de temps afin de faire une mise à jour. Ceci est possible grâce au **DataSet**.

Mode connecté

Les classes du mode connecté

- **Connection**: Permet d'établir une connexion à une source de données spécifiée.
- **Command**: permet l'exécution des commandes de client sur le serveur de base de données
- **DataReader**: (pour les requêtes select): Permet un accès en lecture seule à une source de données.
- **Transaction**: Représente une transaction dans le serveur de la base de données.

Connection 1/2

- Il faut l'instancier
- Il faut renseigner une chaîne de connexion.

```
String strConn = " Data Source=NomServer; Initial  
Catalog=nomBase; User Id=visiteur; Password=visiteur;  
Integrated Security =true";
```

```
SqlConnection con = new SqlConnection();
```

```
con.ConnectionString = strConn;
```

```
con.Open()
```

Connection 2/2

- Les principales méthodes:
 - Open
 - Close

Travailler avec les fichiers de configuration

On peut créer et stocker la chaîne de connexion dans un fichier de configuration du projet :

- Pour ajouter un fichier de configuration : bouton droit sur le projet -> add -> new item -> choisir Application Configuration File.
- N'oubliez pas d'ajouter la référence `System.Configuration`.

Travailler avec les fichiers de configuration

Exemple

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
  <configSections>
  </configSections>
  <connectionStrings>
    <add name="Maconnection"
      connectionString="Data Source=OUARRACHI;Initial Catalog=Client;Integrated Security=True"
      providerName="System.Data.SqlClient" />
  </connectionStrings>
</configuration>
```

-Utilisation de fichier app.config:

**SqlConnection con = new SqlConnection
(ConfigurationManager.ConnectionStrings["Maconnection"].
ConnectionString)**

Commande

- les requêtes SQL et les procédures stockées sont exécutées à partir de commandes.
- contiennent toutes les informations nécessaires à leur exécution et effectuent des opérations telles que créer, modifier ou supprimer des données d'une base de données.

Nom	Type de sources de données
SqlCommand	SQL Server
OleDbCommand	OLE DB
OdbcCommand	ODBC
OracleCommand	Oracle

Commande

- Permet de définir une requête SQL.

- Il faut l'instancier:

```
SqlCommand commande = new SqlCommand();
```

- Renseigner la connexion:

```
commande.Connection= connexion;
```

- Définir la requête

```
commande.CommandText= " insert into ... "
```

ou

```
commande = new SqlCommand("SELECT * FROM Employe", connexion);
```

ou

```
SqlCommand commande = connexion.CreateCommand();  
commande.CommandText = "SELECT * FROM Employe";
```

Commande

1- Requête action:

- Exécuter une requête action (insert, update, delete):

`commande.ExecuteNonQuery();`

Commande

2- Requête Select:

- L'exécution d'une requête select est assez complexe car elle renvoie des lignes de résultats.
- Il faut lire les lignes une par une.
- Utiliser un **datareader**.

SqlDataReader dr = commande.ExecuteReader();

- Imaginer le datareader comme un tableau des lignes de résultats, mais il n'expose qu'une ligne à la fois.

Commande

2- Requête Select:

- Il vous faut donc boucler pour lire toutes les lignes, et chaque champ.
- La connexion reste active pendant tout votre traitement.
- Pensez à refermer le datareader dès que vous avez obtenu les informations désirées.

Commande

2- Requête Select:

- L'exécution de la requête select retourne un datareader.
- La méthode **read** du datareader fait avancer d'une ligne.
- Chaque ligne du data reader contient une collection de colonnes: les champs du select.
- Pour extraire chaque champs.exemple

dr[0] ou dr.GetString(0);

Exemple

```
SqlCommand command = connexion.CreateCommand();
string requete = "SELECT e.ID 'ID', e.Nom, e.Prenom, r.Nom FROM Employe
e, Role r WHERE(e.Role = r.ID) ";
command.CommandText = requete;

connexion.Open();
SqlDataReader lire = command.ExecuteReader();
// Lit les informations de la base de données

while (lire.Read())
{
    Console.WriteLine("Id : {0} Nom : {1} Prenom : {2} Role : {3}",
lire[ 0], lire.GetString(1), lire.GetString(2), lire.GetString(3));
}
// Permet d'afficher

connexion.Close();
```

Les requêtes paramétrées

- Construire la requête par concaténation est lourd et souvent source de problème.
- Vous pouvez utiliser une requête paramétrée.
 - insert COMPTE (Nom,Prenom) Values (@nom,@prenom)
 - Créer des objets paramètres et les lier à la command.
 - Donner des valeurs aux paramètres.

```
nom = Console.ReadLine();  
requete = "INSERT INTO Employe VALUES(@nom, '" + prenom + "'," + role +  
"')";  
SqlParameter param = new SqlParameter("@nom", nom);  
// Permet de paramétrer "nom"  
command.Parameters.Add(param);  
// Ajoute le paramètre param à la collection Parameters  
command.CommandText = requete;
```

Utiliser des transactions

-Vous avez besoin des transactions: Si une des commandes échoue alors l'opération sera arrêtée et la base de données retrouvera son état initial.Alors:

-Utiliser l'objet connection pour débiter la transaction et récupérer ainsi un objet transaction:

```
SqlTransaction tr = connexion.BeginTransaction() ;
```

```
Commande.Transaction=tr;
```

-Utiliser l'objet transaction pour confirmer ou annuler la transaction:

```
tr.Commit() ; //confirmer
```

```
tr.Rollback(); //annuler
```

```

SqlConnection transaction = connection.BeginTransaction();
SqlCommand commande = connection.CreateCommand();
commande.Transaction = transaction;

try
{
    //commande 1
    commande.CommandText = "INSERT Stagiaire (id, nom, prenom, adresse,
telephone, mail, information) VALUES (7, 'DOLLON', 'Julien', '0', '0',
'0', '0')";
    commande.ExecuteNonQuery();

    //commande 2
    commande.CommandText = "INSERT Stagiaire (id, nom, prenom, adresse,
telephone, mail, information) VALUES (4, 'VERGNAULT', 'Bertrand', '0',
'0', '0', '0')";
    commande.ExecuteNonQuery();

    transaction.Commit();
    MessageBox.Show("Transaction validée");
}
catch (Exception Ex)
{
    transaction.Rollback();
    MessageBox.Show(Ex.Message);
}
finally
{
    connection.Close();
}

```

Les procédures stockées

- Les procédures stockées sont des ensembles d'instructions, pouvant être exécutés par simple appel de leur nom via l'instruction EXECUTE.
- Les procédures stockées sont de véritables programmes qui peuvent recevoir des paramètres et renvoyer des valeurs.
- L'utilisation de procédure stockée améliore la performance de l'application.

Les procédures stockées

Pour exécuter une procédure stockée, il faut :

- Indiquer au programme qu'il s'agit d'une procédure stockée

Command.CommandType=CommandType.StoredProcedure

- La propriété CommandText contient le nom de la procédure

Command.CommandText = "NomProcedure "

- Définir des paramètres pour les entrées, les sorties et le code retour.

Monparametre.Direction = ParameterDirection.Input;

//Output ou ReturnValue

Les procédures stockées

Exemple

```
create proc TestProcedure @id int as  
select * from client where id=@id;
```

```
SqlConnection conn = new SqlConnection("Data
Source=server;integrated Security=sspi;initial catalog=pubs;");
SqlCommand cmd = new SqlCommand("TestProcedure", conn);

cmd.CommandType = CommandType.StoredProcedure;
SqlParameter par = cmd.Parameters.Add (" @id",SqlDbType.Int);
    par.Direction = ParameterDirection.Input;
par.Value = 15;
conn.Open();

SqlDataReader lecture = cmd.ExecuteReader();
while (lecture.Read())
    {Console.WriteLine (lecture.GetString (1),lecture.GetString (2)); }
lecture.Close() ;
```

Sql injection

Une injection SQL est un type d'exploitation d'une faille de sécurité d'une application interagissant avec une base de données. L'attaquant détourne les requêtes en y injectant une chaîne non prévue par le développeur et pouvant compromettre la sécurité du système, exemple:

- Autoriser la divulgation complète de toutes les données du système.
- Modification ou suppression des données.
- Devenir administrateur de la base de donnée.

Sql injection

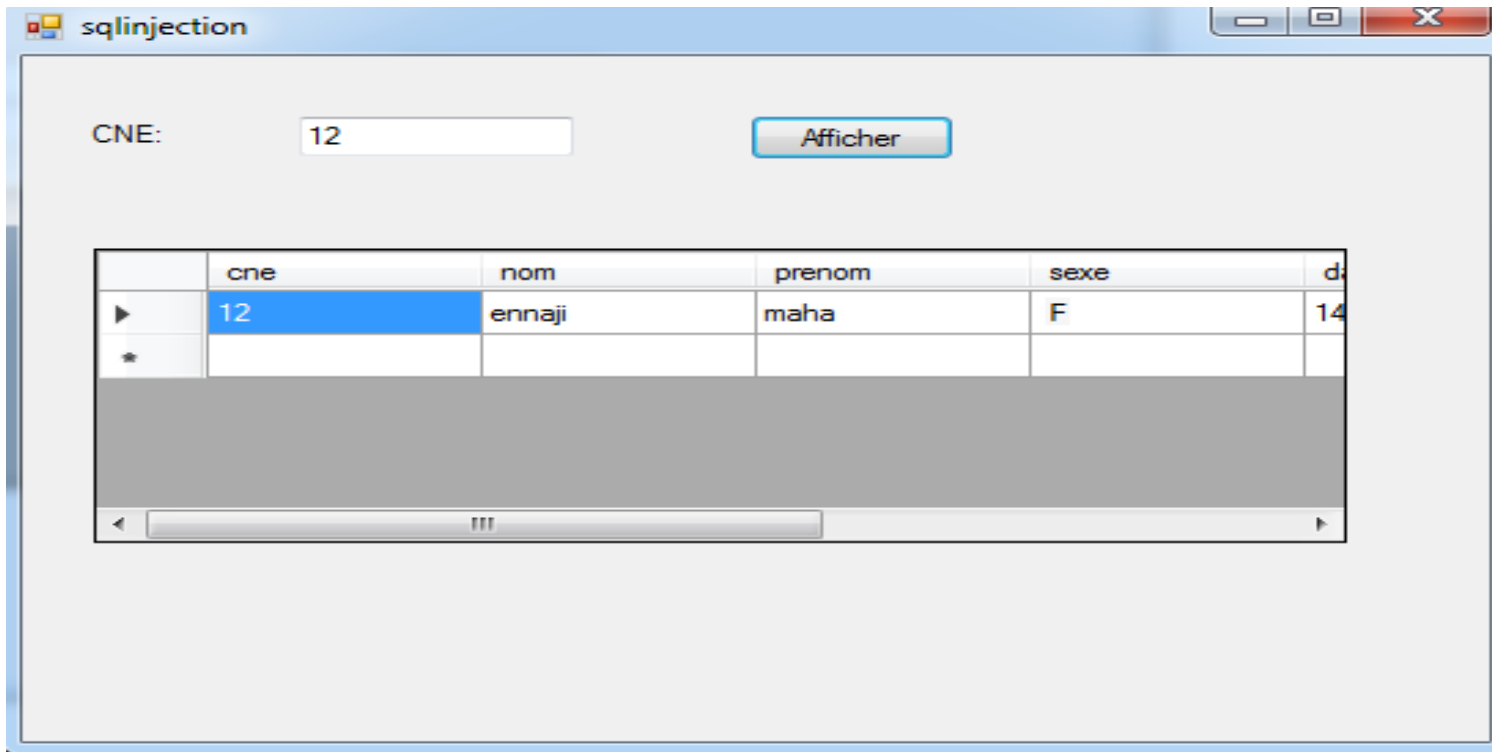
Exemple1: Soit une application contenant un champ textBox permettant de retourner des informations d'une table «étudiant » (dont le cne saisi dans le textBox) après le clic sur le button.

```
cmd = new SqlCommand("select cne,nom,prenom from etudiant where cne='"+textBox1.Text + "'",con);
```

Sql injection

Exemple1:

❑ Utilisation normale de l'application



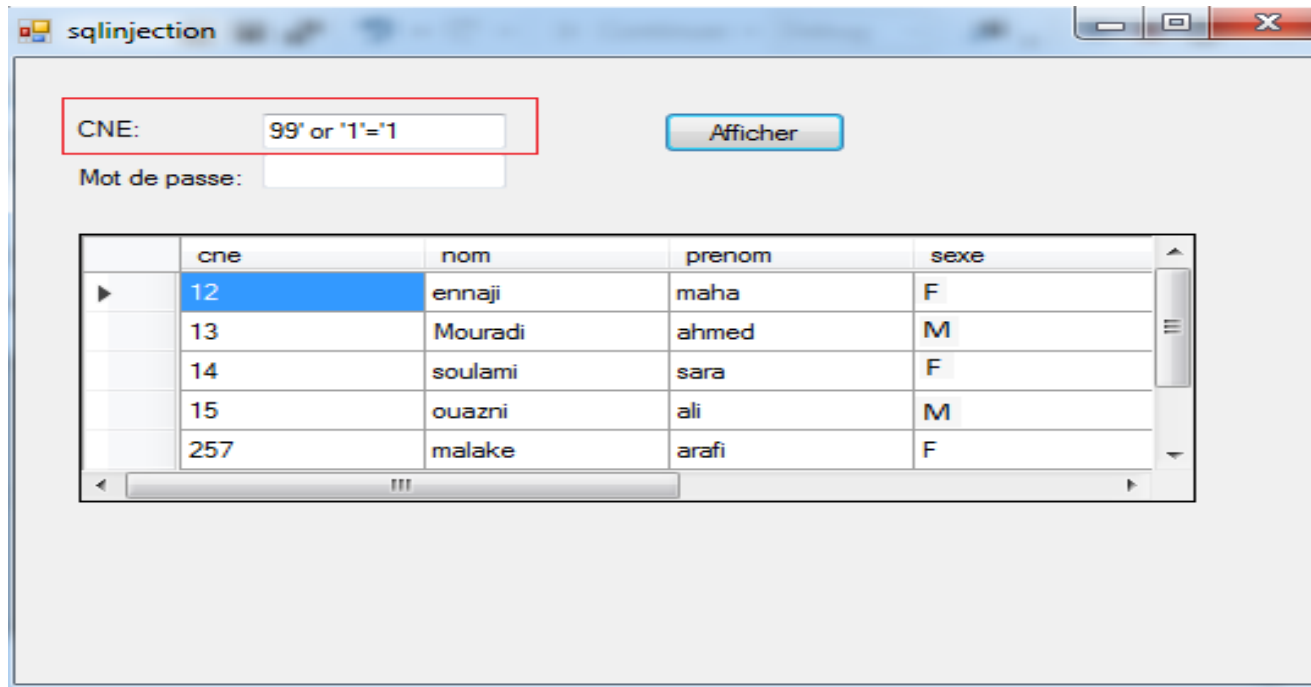
The screenshot shows a web application window titled "sqlinjection". It features a search form with the label "CNE:" followed by a text input field containing the value "12". To the right of the input field is a button labeled "Afficher". Below the search form is a table with the following columns: "cne", "nom", "prenom", "sexe", and "d". The first row of the table is highlighted in blue and contains the values "12", "ennaji", "maha", "F", and "14". Below the table is a horizontal scrollbar.

	cne	nom	prenom	sexe	d
▶	12	ennaji	maha	F	14
★					

Sql injection

Exemple1:

❑ Utilisation par un hacker



The screenshot shows a web application window titled "sqlinjection". It has two input fields: "CNE:" and "Mot de passe:". The "CNE:" field contains the text "99' or '1'='1", which is highlighted with a red box. To the right of the "CNE:" field is a button labeled "Afficher". Below the input fields is a table with the following columns: "cne", "nom", "prenom", and "sexe". The table contains five rows of data, with the first row (cne: 12) highlighted in blue.

cne	nom	prenom	sexe
12	ennaji	maha	F
13	Mouradi	ahmed	M
14	soulami	sara	F
15	ouazni	ali	M
257	malake	arafi	F

N'importe quoi' OR '1'='1

Sql injection

Exemple2:

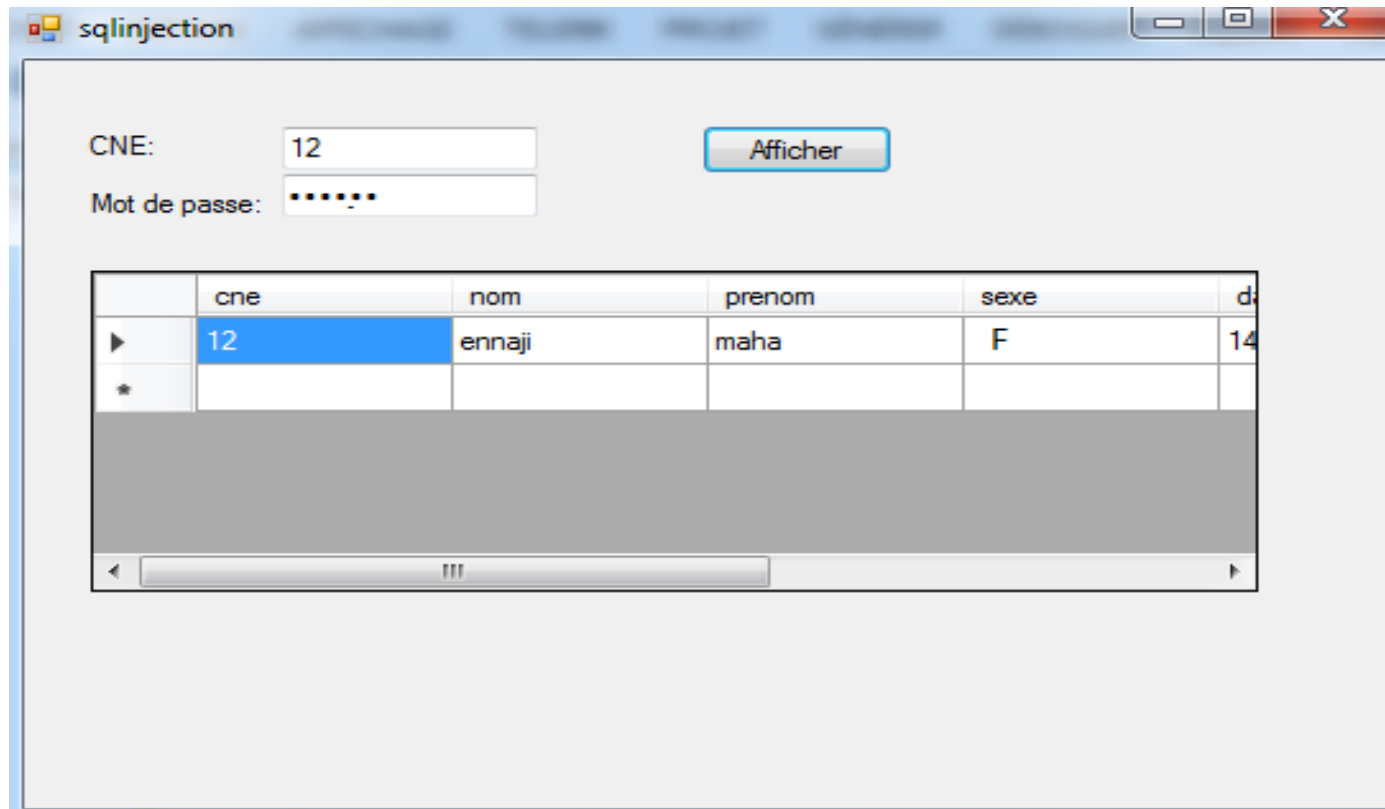
S'authentifier dans une page par le login et le mot de pass

```
cmd=new SqlCommand("select * from etudiant where login='"+textBox1.Text+"' and password='"+textBox2.Text+"'",con)
```


Sql injection

Exemple2:

❑ Utilisateur normal



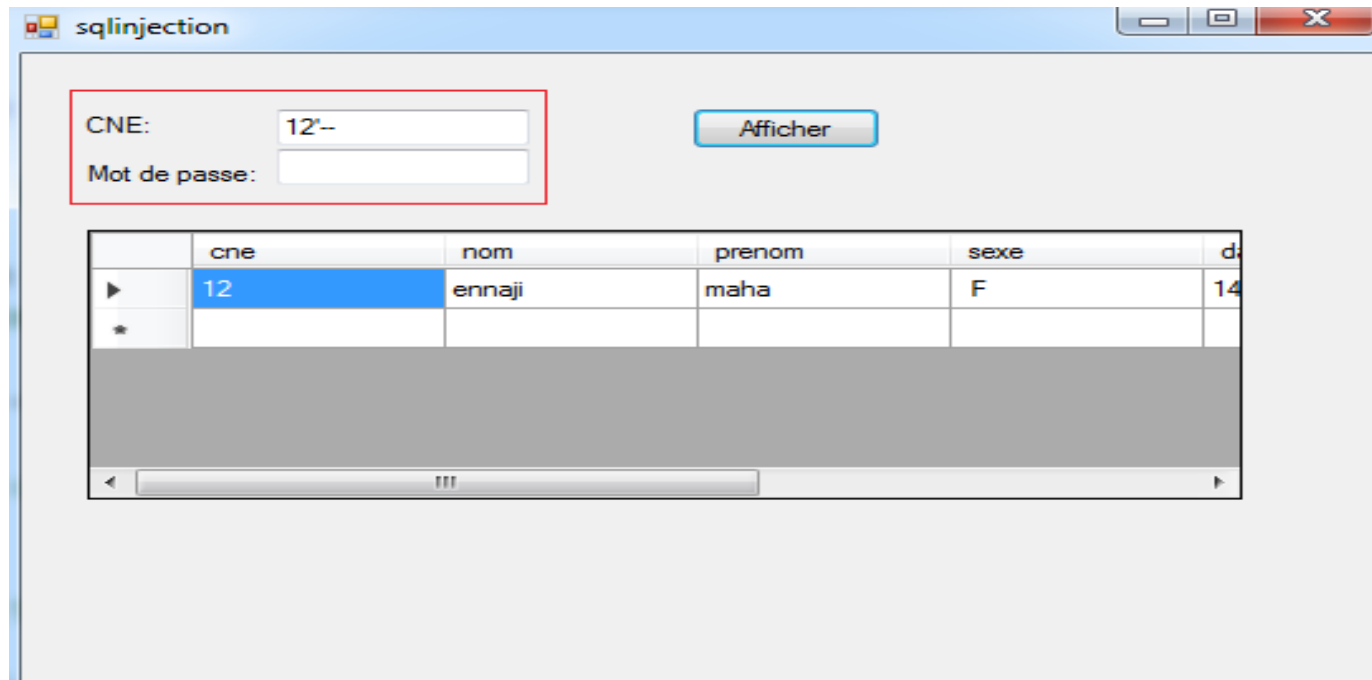
The screenshot shows a web application window titled "sqlinjection". It contains a login form with two input fields: "CNE:" with the value "12" and "Mot de passe:" with masked characters "*****". A button labeled "Afficher" is positioned to the right of the "CNE:" field. Below the form is a table with the following columns: "cne", "nom", "prenom", "sexe", and "date". The first row of the table is highlighted in blue and contains the values "12", "ennaji", "maha", "F", and "14". A scrollbar is visible at the bottom of the table.

	cne	nom	prenom	sexe	date
▶	12	ennaji	maha	F	14
★					

Sql injection

Exemple2:

❑ Un hacker



-Commenter le code qui vient par la suite après le 'and'

Valeur ' --

Sql injection

Solution:

Il y'a trois manières afin d'empêcher l'injection SQL:

- Utiliser les requêtes paramétrés
- Utiliser les procédures stockées
- Passer par un outil ORM

Le mode déconnecté

Le dataSet

- Un dataset permet un stockage local des résultats de requêtes select.
- C' est une mini base de données.

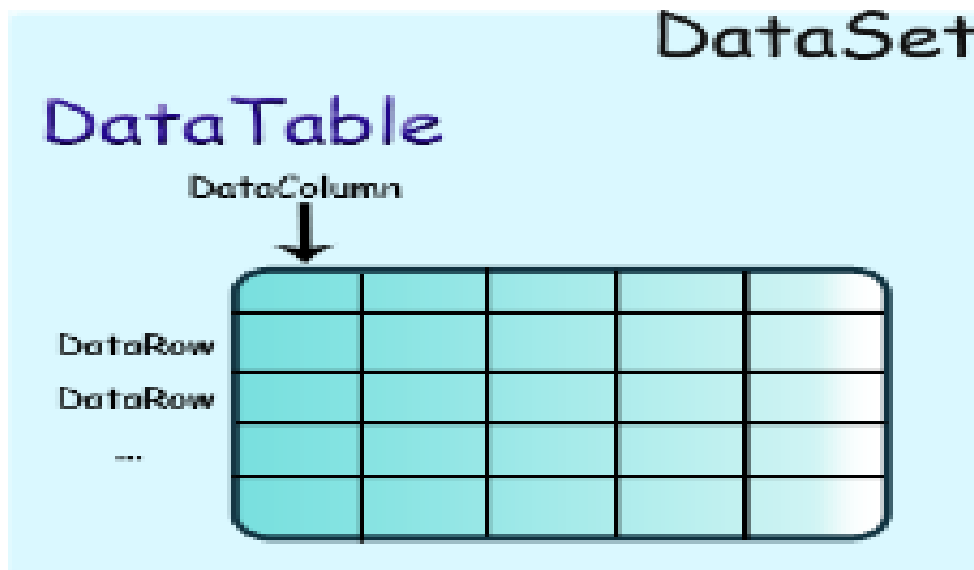
```
DataSet ds = new DataSet();
```

- Il contient:
 - Une collection de DataTable.

DataTable

- Le DataTable contient:
- Une collection de DataRow (le contenu).
- Une Collection de DataColumn (la structure).

`DataTable dt = new DataTable();`



DataTable

- Accéder à un DataRow de l'index i: `DataRow dr=dt.`

`Rows[i]`

- Accéder à champ de position i,j : `dt.Rows[i][j]`

- Ajouter une ligne dans un dataTable: `dt.NewRow();`

- Ajouter un DataRow à un dataTable: `dt.`

`Rows.Add(dr);`

DataTable

- Supprimer la ligne de l'index i de dataTable:

`dt.Rows[i].Delete();`

- Obtient un tableau de tous les objets DataRow qui correspondent à une condition quelconque: `dt.Select(string r)`

Ex: `DataRow[] tab= dt.Select("nom="+textBox1.Text+"");`

DataTable

- Ajouter un dataTable dans un dataSet

```
dt.TableName = "client";
```

```
ds.Tables.Add(dt);
```

```
ds.Tables["client"] //pour appeler ce dataTable
```

DataAdapter

- Le DataAdapter permet de remplir facilement un DataSet, méthode fill().
- Il utilise un objet Command select.

```
SqlDataAdapter dap = new SqlDataAdapter();  
cmd.CommandText = "select * from client";  
dap.SelectCommand = cmd;  
dap.Fill(dt);
```

DataAdapter

-Il permet également de remonter des modifications du dataSet vers la base de données par sa méthode update().

```
dap.Update(ds.Tables["client"]);
```

- Il utilise des Command insert, update, delete

```
dap.InsertCommand = cmd;  
dap.UpdateCommand = cmd;  
dap.DeleteCommand = cmd;
```



On peut les générer automatiquement en utilisant l'objet **SqlCommandBuilder**.

SqlCommand Builder

1.Mettre le code nécessaire pour une requete Action

2.SqlCommandBuilder cmdBuilder = new

SqlCommandBuilder();

cmdBuilder.DataAdapter = dap;

dap.Update(dt);