

Le Développement .NET sous Pocket PC **Et** **SQL Server CE**

Par LEBRUN Thomas

Dans le cadre de leurs développements professionnels, les développeurs peuvent être amenés à réaliser des applications pour Pockets PC, ces mini-ordinateurs de poche qui servent d'agenda ou de logiciel de messagerie mais qui, combinés à un appareil tiers (tel qu'un lecteur de code barre par exemple) et à une base de données embarquée, peuvent s'avérer aussi efficaces qu'un PC portable tout en étant beaucoup plus maniables.

Nous verrons donc dans cet article, les outils que nous pouvons utiliser pour développer en Dotnet sur ces plateformes, ainsi que les possibilités que nous offre le Framework de Microsoft.

I) **Le Compact Framework et SQL Server CE** :

a) **Le Compact Framework** :

Version allégée du Framework .NET de Microsoft, le Compact Framework est utilisé sur les Pockets PC, les SmartPhones et autre périphériques embarqués pour exécuter des applications .NET. Cette version comporte, par exemple, moins de contrôles que le Framework .NET.

Pour mieux comprendre le Compact Framework, je vous suggère de regarder l'article que je lui ai consacré et que vous trouverez à cette adresse :

<http://morpheus.developpez.com/CFDotnet/>

Le Compact Framework peut-être téléchargé depuis le site de [Microsoft](#).

b) **SQL Server CE** :

SQL Server CE est une version de SQL Server, mais pour périphériques mobiles. En l'utilisant, vous avez donc la possibilité d'utiliser des bases de données, de créer des tables, d'effectuer des requêtes SQL, etc..... Bref, depuis la disparition de Pocket Access (la version Pocket PC de Microsoft Access), SQL Server CE est la bonne alternative, surtout parce qu'il est possible de synchroniser la base de données du Pocket PC avec une vraie base de données SQL Server.

Vous trouverez plus d'informations sur SQL Server CE sur ce [site](#), et vous pourrez télécharger SQL Server CE à cette adresse :

<http://www.microsoft.com/sql/ce/downloads/ce20.asp>

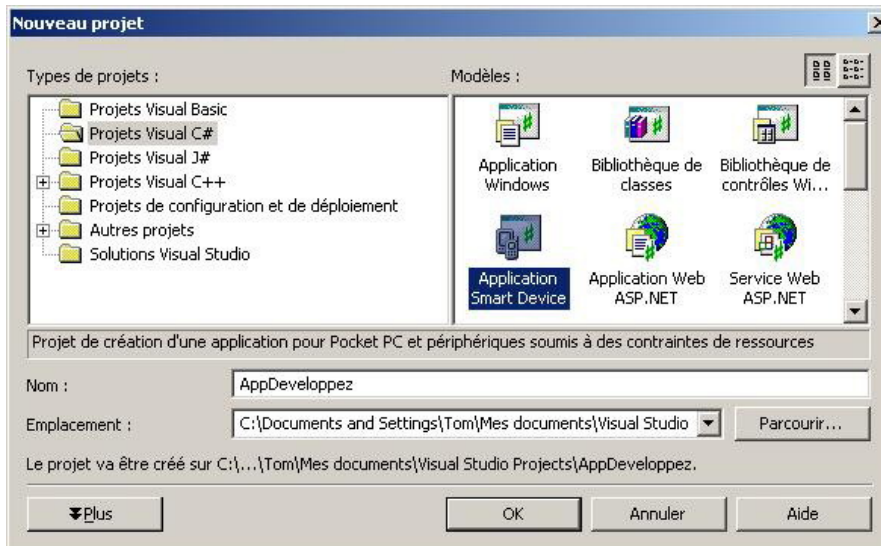
Après avoir vu ce dont nous avons besoin pour développer, voyons maintenant comment l'utiliser.

II) Développement pour Pocket PC

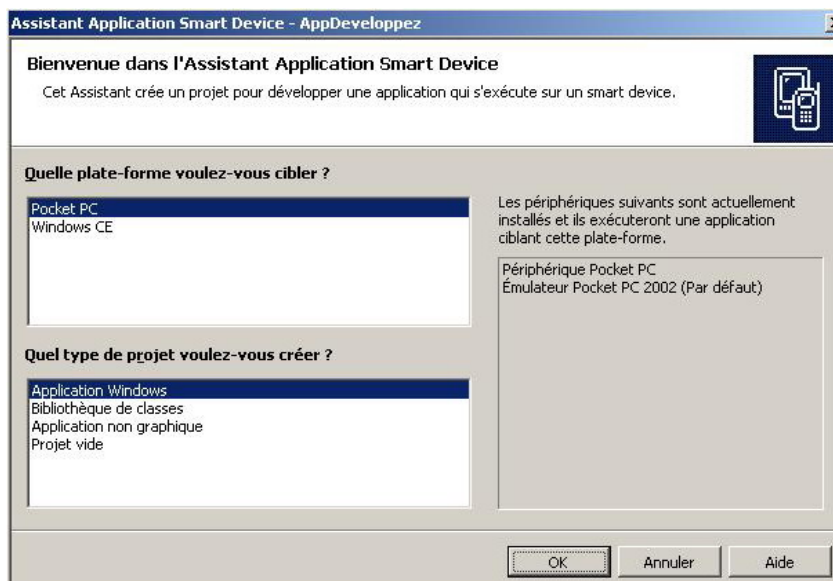
a) L'environnement de développement :

Visual Studio 2003 vous permet de développer facilement et rapidement des applications pour périphériques mobiles.

Pour ce faire, dans Visual Studio, choisissez de faire un nouveau projet (Fichier => Nouveau => Projet) et sélectionner « Application Smart Device », puis cliquer sur « OK »



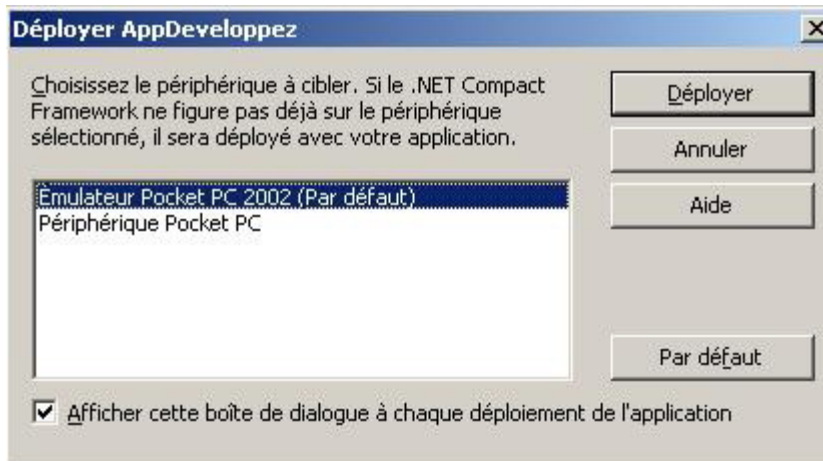
L'écran suivant vous propose ensuite de choisir le type de plateforme pour laquelle vous voulez développer votre application, ainsi que le type de projet que vous souhaitez réaliser. Dans notre cas, nous développons sur « Pocket PC » (Smart Device) et nous souhaitons réaliser une « Application Windows »



Un dernier clic sur le bouton « OK » et nous nous retrouvons face à notre environnement de développement habituel.

A partir de là, il ne vous reste plus qu'à dessiner l'apparence graphique de votre application, et à ajouter le code dont vous avez besoin.

Pour exécuter votre application, pressez la touche « F5 » de votre clavier : là, une fenêtre apparaît vous demandant si vous voulez déployer votre application sur l'émulateur (pratique lors des tests) ou bien directement sur le Pocket PC (celui-ci doit alors être connecté à la machine de développement pour recevoir les fichiers).



Comme indiqué dans la fenêtre, si le Compact Framework n'est pas déjà installé sur le Pocket PC, il sera déployé en même temps que l'application.

Lorsque vous avez fait votre choix, cliquez sur « Déployer » puis patientez : Visual Studio va alors se connecter au périphérique sélectionné (émulateur ou Pocket PC), déployer l'application (et éventuellement le Compact Framework), puis ensuite lancer automatiquement l'application.

Voici, par exemple, le résultat d'une simple application dont le but n'est que d'afficher, lors du clic sur le bouton, le contenu de la TextBox :



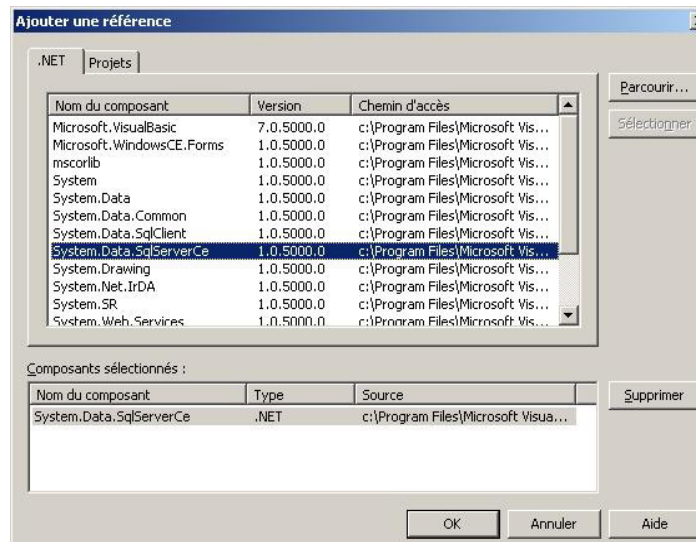
b) SQL Server CE et .NET :

Maintenant que vous avez vu à quel point il est simple de développer une application pour Pocket PC, voyons maintenant comment utiliser SQL Server CE dans nos développements.

• Ajout de la référence au projet :

Une fois le téléchargement et l'installation effectués, rien ne semble avoir changé sur votre poste de développement (hormis une nouvelle entrée dans votre dossier « Programme »).

Pourtant, une nouvelle référence à ajouter est disponible dans Visual Studio. Dans votre projet, faites un clic droit sur « Références », puis cliquez sur « Ajouter une référence ». Là, sélectionnez **System.Data.SqlServerCe** puis cliquez sur « OK » : la référence est maintenant ajoutée à votre projet.



• Ajout du « using » :

Une fois la référence ajoutée, vous devez également penser à faire le « using » (C#) ou « Import » (VB.NET) correspondant :

```
1 using System;  
2 using System.Drawing;  
3 using System.Collections;  
4 using System.Windows.Forms;  
5 using System.Data;  
6 using System.Data.SqlServerCe;
```

- **Création de la base de données :**

Pour créer votre base de données, il vous faut utiliser un objet **SqlCeEngine**, auquel vous passez comme paramètre la chaîne de connexion à la base de données locale. Ensuite, un appel à la méthode **CreateDatabase** permet la création de la base de données.

Voyons cela par le code :

```
private void CreationDatabase()  
{  
    // Chaîne de connexion à la base de donnée locale  
    // Les base de données SQL Server CE sont des fichier  
    // avec l'extension .sdf  
    string sConnexionLocal = @"Data Source=\My Documents\TestDvp.sdf";  
  
    // Objet SqlCeEngine qui permettra la création de la base de données  
    SqlCeEngine Engine = new SqlCeEngine(sConnexionLocal);  
  
    // Création de la base de données  
    Engine.CreateDatabase();  
  
    // Libération des ressources  
    Engine.Dispose();  
}
```

A partir de ce moment-là, vous avez créé la base de données. Il ne vous reste plus qu'à créer la ou les tables de la base, et à faire des requêtes sur ces tables. Pour ce faire, vous avez 2 possibilités : utiliser l'utilitaire SQLCE Query, qui est une sorte d'« Enterprise Manager » simplifié, ou bien faire votre requête par le code.

Voyons donc comment exécuter une requête en la codant entièrement.

- **Utilisation de la base de données :**

Pour effectuer une requête sur une base de données SQL Server CE, il nous faut :

- un objet **SqlCeConnection**, qui servira à faire la connexion à la base de données
- un objet **SqlCommand**, qui exécutera la requête
- éventuellement, un ou des objets **SqlCeParameter** ; qui serviront de paramètre à la requête

A noter que certaines méthodes ou propriétés dont vous aurez besoin ne sont pas ajoutées automatiquement : vous devez ajouter une référence à « System.Data.Common » et ajouter le using (ou Import) suivant :

```
using System.Data.Common ;
```

Voici un exemple de code :

```
private void SqlInsertValuesRequest()
{
    try
    {
        // Connexion à la base
        SqlCeConnection SqlCnx = new SqlCeConnection(@"Data Source=\My
Documents\TestDvp.sdf");

        // Requête SQL à exécutée
        // Nous partons du principe que la table ma_table existe déjà dans
votre base de données
        string sSQL = "INSERT INTO ma_table VALUES (?, ?)";

        // Objet SqlCeCommand
        SqlCeCommand SqlCommand = new SqlCeCommand(sSQL, SqlCnx);

        // SQL Server CE accepte les requêtes paramétrées mais n'accepte pas
les paramètres nommés (@name)
        // Le caractère à utilisé à la place est le point d'interrogation (?)

        // Déclaration des paramètres de la requête, avec leur spécificité
        // Leur nom importe peu : seule leur position dans le code indiquera
leur position dans la requête
        SqlCeParameter ParamNom = new SqlCeParameter("Nom",
SqlDbType.NVarChar);
        ParamNom.Value = tbNom.Text.Replace("'", "'");
        SqlCeParameter ParamPrenom = new SqlCeParameter("Prenom",
SqlDbType.NVarChar);
        ParamPrenom.Value = tbPrenom.Text.Replace("'", "'");

        // Ajout des paramètres de la requête paramétrée
        SqlCommand.Parameters.Add(ParamNom);
        SqlCommand.Parameters.Add(ParamPrenom);

        // ouverture de la connexion
        SqlCnx.Open();

        // Création d'une version "compilée" de la requête
        SqlCommand.Prepare();

        // Exécution de la requête
        SqlCommand.ExecuteNonQuery();

        // Si la connexion est ouverte, on la ferme.
        if ( SqlCnx.State == ConnectionState.Open )
        {
            SqlCnx.Close();
        }

        MessageBox.Show("Requête SQL effectuée avec succès", "OK",
MessageBoxButtons.OK, MessageBoxIcon.Exclamation, MessageBoxDefaultButton.Button1);
    }
    catch(SqlCeException Ex)
    {
        MessageBox.Show("Erreur SQL inattendue.\n" + Ex.Message, "Erreur",
MessageBoxButtons.OK, MessageBoxIcon.Exclamation, MessageBoxDefaultButton.Button1);
    }
    catch(Exception Ex)
    {
        MessageBox.Show("Erreur inattendue.\n" + Ex.Message, "Erreur",
MessageBoxButtons.OK, MessageBoxIcon.Exclamation, MessageBoxDefaultButton.Button1);
    }
}
```

III) Synchronisation avec une base de données SQL Server

Plutôt que de refaire ce qui existe déjà, je vais vous renvoyer vers l'excellent article de [Christian Peyrusse](#) qui traite de « la Synchronisation de données sur Pocket PC en utilisant la méthode Remote Data Access » et que vous pouvez visualiser à cette adresse : <http://leduke.developpez.com/PocketPC/>

IV) Conclusion

En conclusion, nous avons vu, à travers cet article, à quel point le développement d'application Smart Device (comprenez le développement d'applications pour périphériques mobiles) était relativement aisé et ne différençait guère du développement d'applications WindowsForms.

De plus, SQL Server CE est une base de données relativement simple d'utilisation mais qui, combinée à l'excellent Compact Framework de Microsoft, permet de réaliser des applications puissantes.

A bientôt pour un autre article ☺