



Mercredi 23 mai 2007

Les Mercredis  
du Développement  
msdn

# back MEDC

Stéphane Sibué



Stéphane Sibué

## Webmaster et fondateur de CodePPC

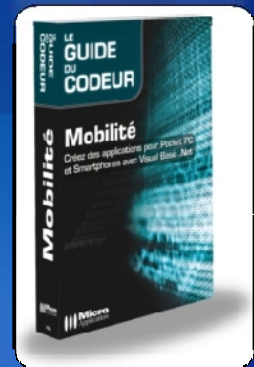
- La référence francophone du développement WM.
- En ligne depuis juin 2001.
- Des articles, des sources, des outils, des forums.
- Plus de 3000 développeurs par jour.



## Auteur du Guide du Codeur Mobilité

Chez Micro Application (octobre 2006)

*Développement d'applications Smart Device avec  
Visual Studio 2005 et le Compact Framework 2*



[www.codeppc.com](http://www.codeppc.com)



Stéphane Sibué

## Responsable technique Chez Antéis



Créé en 1991 par Stéphane Sibué

Depuis 2004, membre du groupe Magnin Gécors (230 personnes)

Basée à Chambéry (73)

Développement sur mesure d'applications dans les domaines de la gestion, l'industrie, le médical, l'Internet, et la mobilité en utilisant prioritairement les technologies Microsoft, et plus particulièrement .NET

Formations aux développeurs

## Microsoft MVP

Device Application Development



[www.anteis.fr](http://www.anteis.fr)



# Utilisation de la sérialisation avec le .NET Compact Framework 2



## La sérialisation XML

Permet de sauvegarder et de restaurer un objet en  
Écrivant les données qui le composent sous la forme  
D'un flux XML.

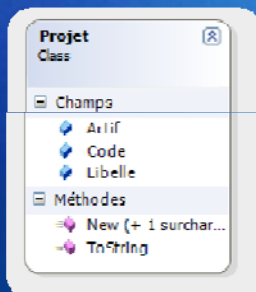
Existe dans le Framework .NET depuis le début.

Existe dans le Compact Framework .NET que depuis la version 2.

Espace de noms `System.Xml.Serialization`



## La sérialisation XML d'un seul objet



Instanciation

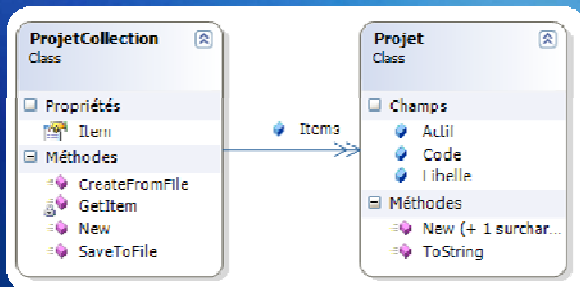
**Code = «F76291-0... »**  
**Libelle = « Calisto »**  
**Actif = True**

Sérialisation

```
<?xml version="1.0" encoding="utf-8" ?>  
- <Projet xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema">  
  <Actif>true</Actif>  
  <Libelle>Calisto</Libelle>  
  <Code>F7629491-0CC4-49E3-B283-71466A245A05</Code>  
</Projet>
```



## La sérialisation XML d'une collection d'objets



Instanciation

### Les Projets

Code = «F76291-0... »  
Libelle = « Calisto »  
Actif = True

Code = «1186C5... »  
Libelle = « Gounter »  
Actif = True

Code = «D5D683... »  
Libelle = « DoWeek 2 »  
Actif = True

Sérialisation

```
<?xml version="1.0" encoding="utf-8"?>
<ProjecCollection xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  - <Items>
  - <Projet>
    <Actif>true</Actif>
    <Libelle>Calisto</Libelle>
    <Code>F7629491-0CC4-49E3-B283-71466A245A05</Code>
  </Projet>
  - <Projet>
    <Actif>true</Actif>
    <Libelle>Gounter</Libelle>
    <Code>1186C5A9-260D-44EB-8279-FA2C3B149673</Code>
  </Projet>
  - <Projet>
    <Actif>true</Actif>
    <Libelle>DoWeek 2</Libelle>
    <Code>D5D68237-47DB-4ED8-8D33-74DE70123396</Code>
  </Projet>
  </Items>
</ProjecCollection>
```



## Pour déclarer une classe sérialisable

### Utiliser l'attribut `<Serializable()>` et un constructeur « simple »

```
<Serializable()> _  
Public Class Projet  
  
    Public Code As String  
    Public Libelle As String  
    Public Actif As Boolean  
  
    Public Sub New()  
    End Sub  
  
    Public Sub New(ByVal wCode As String, _  
        ByVal wLibelle As String, _  
        Optional ByVal wActif As Boolean = True)  
  
        With Me  
            .Code = wCode  
            .Libelle = wLibelle  
            .Actif = wActif  
        End With  
  
    End Sub  
  
    Public Overrides Function ToString() As String  
  
        Return Me.Libelle  
  
    End Function  
  
End Class
```



## Déclaration de la classe « ProjetCollection »

```
<Serializable()> _
Public Class ProjetCollection

    Public Items As System.Collections.Generic.List(Of Projet)

    Public Sub New()
        Items = New System.Collections.Generic.List(Of Projet)
    End Sub

    Public ReadOnly Property Item(ByVal wCode As String) As Projet
        ...
    End Property

    Private Function GetItem(ByVal wCode As String) As Projet
        ...
    End Function

    Public Function SaveToFile(ByVal wFilename As String) As Boolean
        ...
    End Function

    Public Shared Function CreateFromFile(ByVal wFilename As String) As ProjetCollection
        ...
    End Function

End Class
```



## Pour sérialiser...

```
Public Function SaveToFile(ByVal wFilename As String) As Boolean

    Dim wFile As System.IO.StreamWriter
    Dim wSerializer As System.Xml.Serialization.XmlSerializer
    Dim wRetVal As Boolean

    REM Ouverture du fichier
    Try
        wFile = New System.IO.StreamWriter(wFilename, False)
    Catch ex As Exception
        MsgBox(ex.Message, MsgBoxStyle.Exclamation)
        Return False
    End Try

    REM Création du serializer
    wSerializer = New System.Xml.Serialization.XmlSerializer(Me.GetType)
    wRetVal = True

    REM Lecture des infos XML
    Try
        wSerializer.Serialize(wFile, Me)
    Catch ex As Exception
        MsgBox(ex.Message, MsgBoxStyle.Exclamation)
        wRetVal = False
    End Try

    REM Fermeture du fichier
    wFile.Close()

    Return wRetVal

End Function
```



## Pour désérialiser...

```
Public Shared Function CreateFromFile(ByVal wFilename As String) As ProjetCollection

    Dim wFile As System.IO.StreamReader = Nothing
    Dim wSerializer As System.Xml.Serialization.XmlSerializer
    Dim wInstance As ProjetCollection = Nothing

    REM On ouvre le fichier
    Try
        wFile = New System.IO.StreamReader(wFilename)
    Catch ex As Exception
    End Try

    If wFile IsNot Nothing Then

        REM Lecture de l'objet
        wSerializer = New System.Xml.Serialization.XmlSerializer(GetType(ProjetCollection))

        Try
            wInstance = wSerializer.Deserialize(wFile)
        Catch ex As Exception
            MsgBox(ex.Message, MsgBoxStyle.Exclamation)
        End Try

        REM Fermeture du fichier
        wFile.Close()

    End If

    REM On retourne l'objet lu (ou nothing si erreur de lecture)
    Return wInstance

End Function
```



# D mo





# Utilisation d'un GPS



## Utilisation d'un GPS



**GPS = Global Positioning System**

Système de localisation terrestre.

Permet de se positionner de manière précise et rapide n'importe où sur la surface de la terre.

La réception des données provenant des satellites et les calculs complexes qui doivent être réalisés pour déterminer la position sont effectués par des appareils spécialisés, de plus en plus bon marché, que sont les récepteurs GPS.



## Utilisation d'un GPS

**NMEA** = National Marine & Electronics Association

Protocole utilisé par les GPS.

Une trame = Une information typée.

```
$GPRMC,225446,A,4916.45,N,12311.12,W,000.5,054.7,191194,020.3,E*68
```



## Utilisation d'un GPS

```
$GPRMC,225446,A,4916.45,N,12311.12,W,000.5,054.7,191194,020.3,E*68
```

Identifiant de la trame (type)



## Utilisation d'un GPS

```
$GPRMC,225446,A,4916.45,N,12311.12,W,000.5,054.7,191194,020.3,E*68
```

Somme de contrôle de la trame



## Utilisation d'un GPS

```
$GPRMC,225446,A,4916.45,N,12311.12,W,000.5,054.7,191194,020.3,E*68
```

Données de la trame



## Utilisation d'un GPS

### Exemple de la trame de type **\$GPRMC**

<b>\$GPRMC</b>	Type de trame
<b>225446</b>	Heure du fix
<b>A</b>	Alerte (A=OK ; V=WARNING)
<b>4916.45</b>	Latitude au format ddm. ss
<b>N</b>	Sens de la latitude (N=Nord=Positif, S=Sud=Négatif)
<b>12311.12</b>	Longitude au format dddmm. ss
<b>W</b>	Sens de la longitude (E=Est=Positif, W=Ouest=Négatif)
<b>000.5</b>	Vitesse au sol en noeuds
<b>054.7</b>	Cap vrai
<b>191194</b>	Date du fix
<b>020.3</b>	Déclinaison magnétique
<b>E</b>	Sens de la déclinaison magnétique
<b>*68</b>	Somme de contrôle



## Utilisation d'un GPS

### Très important !!!!

Les données numériques sont toujours notées avec le point comme séparateur décimal !

Attention aux conversions String -> Numérique

12311.12	Longitude au format dddmm.ss
054.7	Cap vrai
020.3	Déclinaison magnétique



## Utilisation d'un GPS

Il existe d'autres types de trames

**\$GPGGA**

**\$GPRMC**

**\$GPGSA**

...

Quelques liens utiles sur le GPS et le NMEA...

<http://www.gpspassion.com>

<http://ditwww.epfl.ch/SIC/SA/publications/FI98/fi-5-98/5-98-page1.html>



## Utilisation d'un GPS

### La communication par liaison série.

Le récepteur GPS utilise la **liaison série** pour communiquer.

Ceci est valable quelque soit le type de récepteur :

- Interne
- Externe par Bluetooth
- Externe par Câble
- Externe par SDIO
- Externe par Compact Flash

Il faut donc utiliser les fonctions de gestion des ports série pour communiquer avec un récepteur GPS.

Le CF 2 propose **le contrôle SerialPort** pour réaliser cela.



## Utilisation d'un GPS

### Le contrôle SerialPort

- Panel
- PictureBox
- ProgressBar
- RadioButton
- saveFileDialog
- SerialPort**
- Splitter
- StatusBar
- TabControl
- TextBox
- Timer



## Utilisation d'un GPS

### Pour ouvrir le port de communication :

```
With pSerial
    .PortName = "COM1"
    .BaudRate = 9600
    .Parity = IO.Ports.Parity.None
    .DataBits = 8
    .StopBits = IO.Ports.StopBits.One
End With

Try
    pSerial.Open()
Catch ex As Exception
End Try

If pSerial.IsOpen Then ...
```



## Utilisation d'un GPS

Pour lire des données, il suffit de vérifier qu'il y en a dans le buffer d'entrée et de les lire :

```
Dim wLen As Integer
Dim wBuffer As String

Try
    wLen = pSerial.BytesToRead
Catch ex As Exception
End Try

If wLen > 0 Then wBuffer = pSerial.ReadExisting()
```



## Utilisation d'un GPS

Gestion de l'énergie = Arrêt du device pendant la lecture des données GPS !

A intervalle régulier, utiliser **SystemIdleTimerReset**

A chaque réception d'une nouvelle trame GPS par exemple.

Déclaration en VB :

```
Declare Sub SystemIdleTimerReset Lib "coredll" ()
```



## Utilisation d'un GPS

# Démo

