Xmlx

Gregory Adam

04/07/2013

Contents

Introduction 3

Caractéristiques 3

Exemple de programme 4

Création d'un Document 8

Créer un nouveau type 9

Ajouter un élément 10

Ajouter un attribut a un élément 11

Production de l'XML 12

Appendix A – Propriétés des Types 13

Appendix B – Types inclus 14

Introduction

Xmlx est une classe qui permet de construire une arborescence XML en mémoire et de la transformer sous forme de chaine ou fichier.

Cela veut dire qu'il ne faut plus s'occuper du fichier XML (encodage/fputs()/TextMerge) et qu'on peut se concentrer sur la structure même.

Caractéristiques

- XML en UTF-8
- Méthodes pour
    - Créer des éléments (nœuds) avec ou sans texte
    - Créer des attributs (attachés aux éléments)
    - Créer des types basés sur d'autres types (18 sont déjà inclus)
    - Transformer en chaine (avec ou sans TABs/CR pour la lisibilité)
    - Transformer en fichier (avec ou sans TABs/CR pour la lisibilité)
- Contrôle des valeurs (type, valeur min/max) au moment de leur addition au document
- Encodage
    - Du codepage actif vers UTF-8
    - Traiter les caractères spécifiques a XML (< > &)
- Assez rapide.
    - La première version prenait 9 secondes pour ajouter 5000 éléments/nœuds et la transformation en fichier
    - La deuxième version le fait en moins d'une seconde

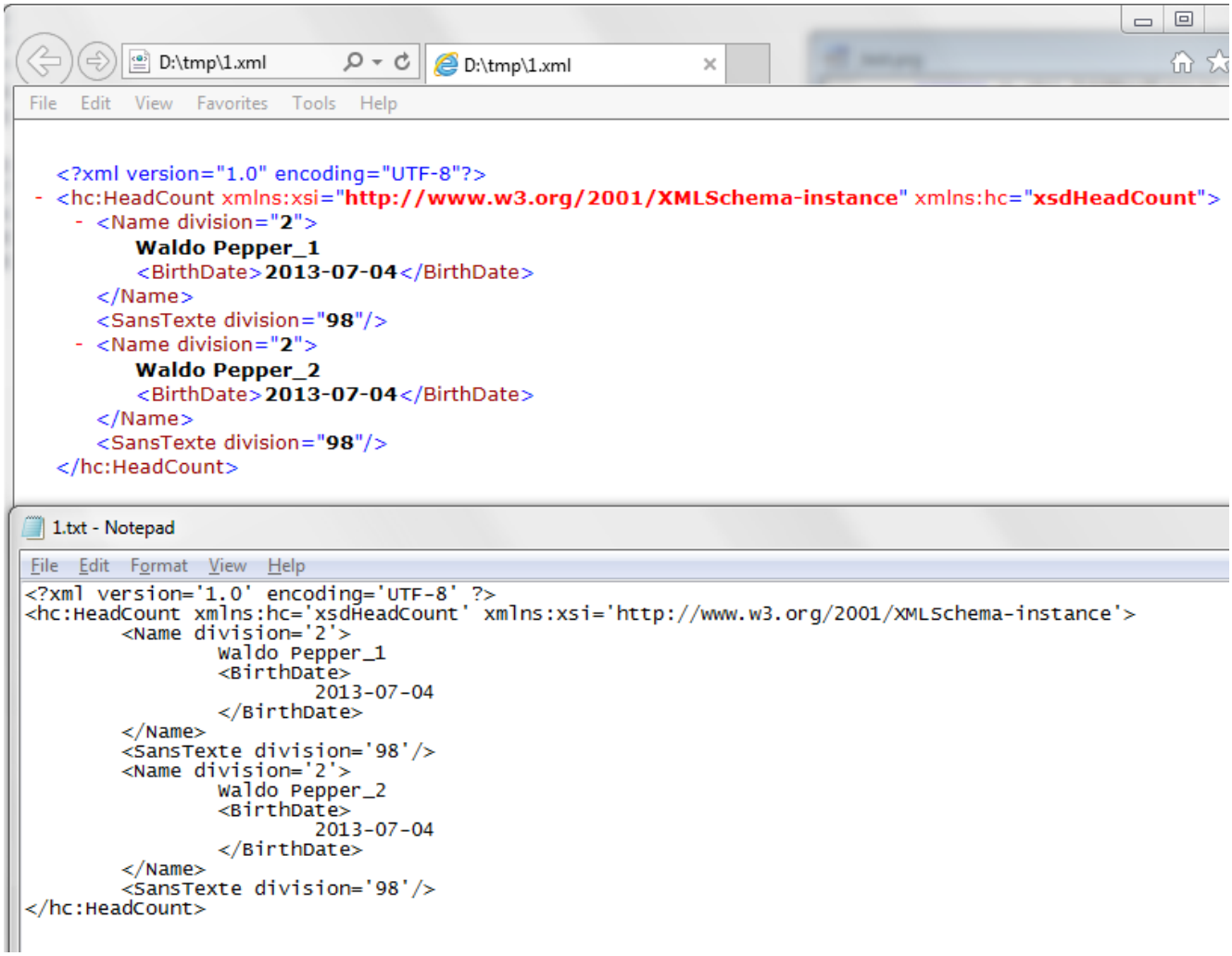
Exemple de programme

```
#define true      .t.
#define false     .f.

local success
success = true

local xmlDocObj, fileXml, rootnodeName, typeObj

fileXml = 'd:\tmp\1.xml'

rootnodeName = 'hc:HeadCount' && nom de la racine

do case
case !m.success

&& creation du document + racine
case !Xmlx_NewObject_Document(@m.xmlDocObj, m.rootnodeName)
      assert false
      success = false

endcase

&& creation d'un nouveau type 'divisionType' base: integer
do case
case !m.success

&& creation d'un nouveau type 'divisionType' base: integer
case !m.xmlDocObj.GetNewType(@m.typeObj, 'integer', 'divisionType')
      assert false
      success = false

otherwise
      && restrictions : entre 20 et 98
      typeObj.MinIncl = 20
      typeObj.MaxExcl = 99

endcase

do case
case !m.success

&& ajout du type divisionType
case !m.xmlDocObj.AddNewType(m.typeObj)
      assert false
      success = false

endcase

&& creation d'un deuxieme type 'pepperNameType' base: 'string'


do case
```

```
case !m.success

case !m.xmlDocObj.GetNewType(@m.typeObj, 'string', 'pepperNameType')
	assert false
	success = false

otherwise
	typeObj.MinLength = 10  && longueur minimale de 10
	typeObj.MaxLength = 30  && longueur maximale de 30

	&& tous les noms doivent contenir 'Pepper'
	typeObj.Pattern = '.*Pepper.*'

endcase

do case
case !m.success

&& ajout du type pepperNameType
case !m.xmlDocObj.AddNewType(m.typeObj)
	assert false
	success = false

endcase


&& ajout des namespace = attributs de la racine

do case
case !m.success

&& nom, type, valeur
case !m.xmlDocObj.AddAttribute(m.xmlDocObj.Root, 'xmlns:hc', 'string',
'xsdHeadCount')
	assert false
	success = false

case !m.xmlDocObj.AddAttribute(m.xmlDocObj.Root, 'xmlns:xsi', 'string',
'http://www.w3.org/2001/XMLSchema-instance')
	assert false
	success = false

endcase

local parentId, childId, i
parentId = m.xmlDocObj.Root

for i = 1 to 2
	do case
	case !m.success
		exit

	&& creation d'un element avec texte
	case !m.xmlDocObj.AddElement(@m.childId, m.parentId, 'Name',
'pepperNameType', 'Waldo Pepper_'+transform(m.i))
		assert false
		success = false
```

```
      && ajouter un atttribut a cet element: division = '20'
      case !m.xmlDocObj.AddAttribute(m.childId, 'division', 'divisionType',
20)
            assert false
            success = false

      && ajouter la date de naissance a cet element
      case !m.xmlDocObj.AddElement( , m.childId, 'BirthDate', 'date',
date())
            assert false
            success = false

      && creation d'un element sans texte (
<SansTexte><divisionType>98</divisionType></SansTexte> )
      case !m.xmlDocObj.AddElement(@m.childId, m.parentId, 'SansTexte',
null, null)
            assert false
            success = false

      && ajouter divisionType a cet element
      case !m.xmlDocObj.AddAttribute(m.childId, 'division', 'divisionType',
98)
            assert false
            success = false
      endcase

endfor

do case
case !m.success

&& ecrire ce document
case !m.xmlDocObj.ToFile(m.fileXml, false)
      assert false
      success = false

&& optionel pour le developpement
&& ecrire le doc sous fichier texte pour inspection visuelle
&& avec trop de TABs et CRLF
case !m.xmlDocObj.ToFile(forceext(m.fileXml, 'txt'), true)
      assert false
      success = false

endcase
```

Ce programme produit deux fichiers, un XML et un TXT, dont le fichier TXT ne sert que pendant le développement.



Toute méthode retourne .T. si OK et .F. en cas de problème (mauvais paramètres etc.)

Création d'un Document

Local docObj

```
If( !Xmlx_NewObject_Document(@m. docObj, rootElementName))
            && erreur
Else
            && ok
Endif
```

Paramètres

- docObj : passer par référence, c'est l'objet document même
- rootElementName : le nom de la racine

Créer un nouveau type

Se fait en trois étapes

1. Obtenir un nouveau type
    - `xmlDoc.GetNewType(@m.typeObj, parenttypename, typeName)`
        - `typeObj` : nouvel objet type
        - parenttypename : le type sur lequel le nouveau type est basé
        - typeName : le nom du nouveau type
2. Modifier les propriétés du nouveau type (voir Appendix A)
    - Exemple : typeObj.minLength = 20
3. Ajouter ce type
    - xmlDoc.`AddNewType(m.typeObj)`

Ajouter un élément

xmlDoc. `AddElement(@m.childId, parentId, elementName, datatype, value)`

- childId : id du nouvel élément créé
    - on pourra ajouter des éléments a ce nouvel élément en passant ce childId comme deuxième paramètre a AddElement()
- parentId : id de l'élément auquel on ajoute
- elementName : le nom de cet élément
- datatype : le type de cet élément
- value : la valeur de cet élément

Notes

- Si c'est un élément sans texte (valeur) il faut passer null pour datatype et value

Ajouter un attribut a un élément

=xmlDoc. `AddAttribute(parentId, attributeName, datatype, value)`

- parentId : id de l'élément auquel on ajoute cet attribut
- attributeName : le nom de cet attribut
- datatype : le type de cet attribut
- value : la valeur de cet attribut

Production de l'XML

xmlDoc. `ToFile(filename [, lBeautify])`

`xmlDoc. GetString(@m.s [, lBeautify] )`

ToFile() produit un fichier et GetString() produit une chaine

- filename : nom du fichier
- s : chaine à passer par référence
- lBeautify
    - si .F. ou absent, produit de l'XML très compacte (sans TAB et CRLF)
    - si .T. insère des TAB et CRLF pour produire un fichier qui montre les niveaux

Appendix A – Propriétés des Types

Nom	
length	Longueur fixe
minLength	Longueur minimale
maxLength	Longueur maximale
totalDigits	Nombre maximal de chiffres
fractionDigits	Nombre maximal de chiffres après le séparateur décimal Note : si spécifié, c'est le nombre qu'on retrouvera dans le fichier xml. Exemple : fractionDigits= 2 Valeur = 0, dans l'XML on retrouvera 0.00
minIncl	Valeur minimale (v >= minIncl)
maxIncl	Valeur maximale (v <= maxIncl)
minExcl	Valeur minimale (v > minExcl)
maxExcl	Valeur maximale (v < maxExcl)
pattern	Pattern regex

Appendix B – Types inclus

Nom	Parent	Caractéristiques
decimal		
string		
boolean		
date		
dateTime		
byte	decimal	fractionDigits = 0, minIncl = -128, maxIncl = 127
int	decimal	fractionDigits = 0, minIncl= -0x80000000, maxIncl = 0x7fffffff
integer	decimal	fractionDigits = 0
long	decimal	fractionDigits = 0, minIncl = `-9223372036854775808`, maxIncl = `9223372036854775807`
negativeInteger	decimal	fractionDigits = 0, maxIncl = -1
nonNegativeInteger	decimal	fractionDigits = 0, minIncl = 0
nonPositiveInteger	decimal	fractionDigits = 0, maxIncl = 0
positiveInteger	decimal	fractionDigits = 0, minIncl = 1
short	decimal	fractionDigits = 0, minIncl = `-0x8000`, maxIncl = `0x7fff`
unsignedLong	decimal	fractionDigits = 0, minIncl = 0, maxIncl = `18446744073709551615`
unsignedInt	decimal	fractionDigits = 0, minIncl = 0, maxIncl = `4294967295`
unsignedShort	decimal	fractionDigits = 0, minIncl = 0, maxIncl = `0xffff`
unsignedByte	decimal	fractionDigits = 0, minIncl = 0, maxIncl = `0xff`