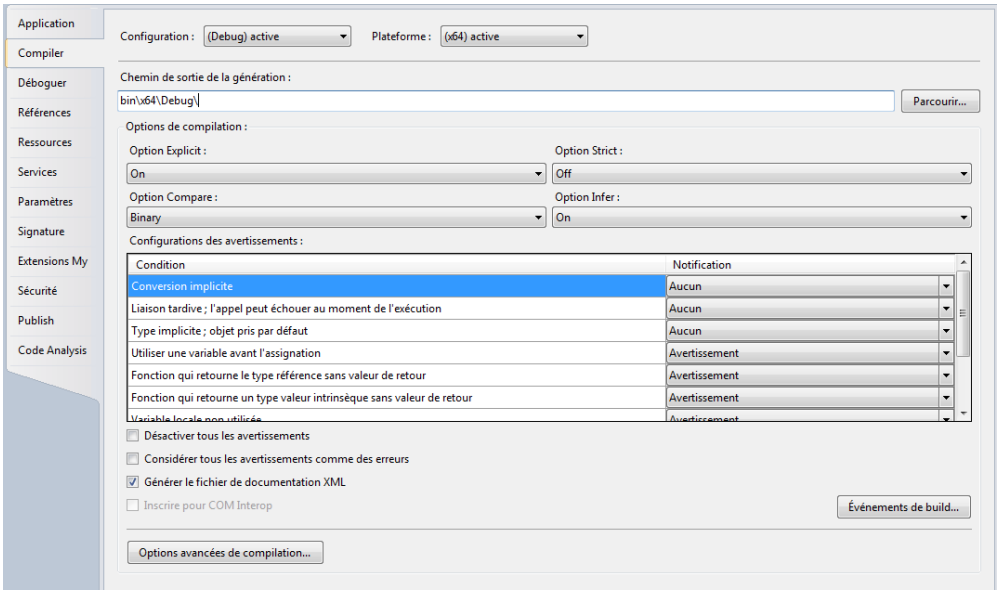
Exemple TFS 2010 & WinForm

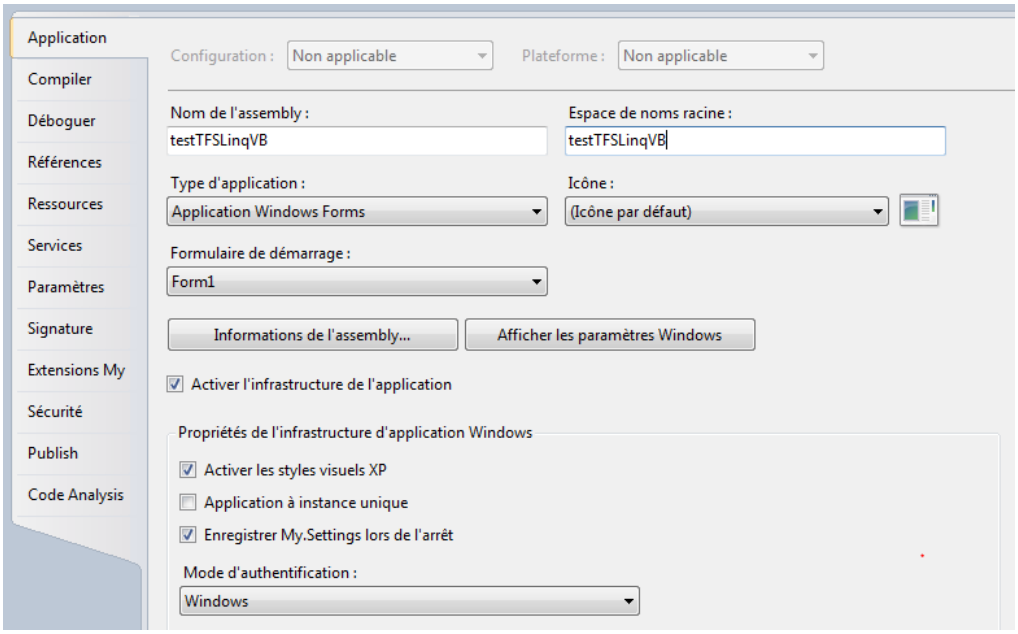
Une application WinForm pour accéder aux données de Team Foundation 2010

Créer un projet Winform en vb. Nommé le par exemple « TestTFSLinqVB »

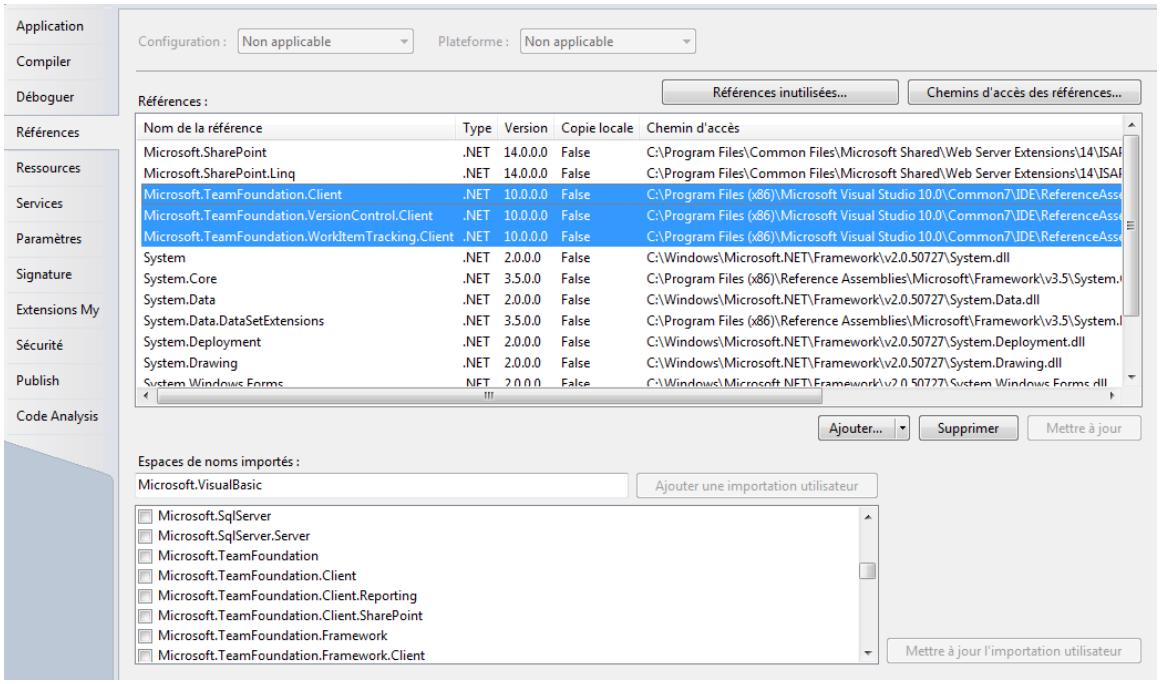
Dans les propriétés du projet, passez en mode 64bits. C'est impératif.



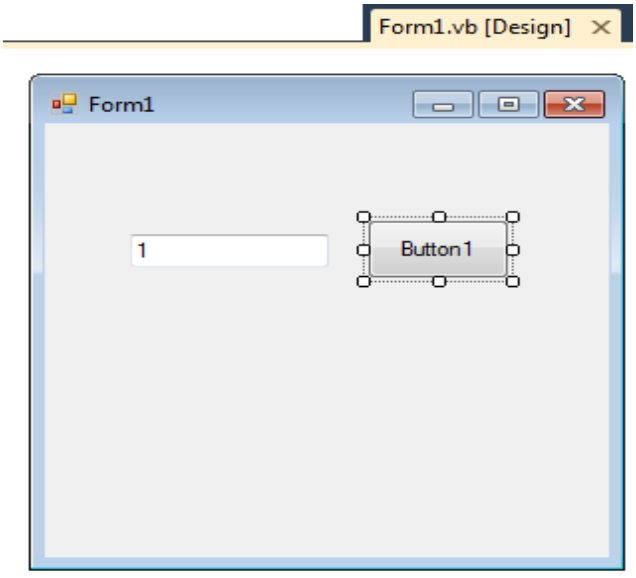
Adaptez l'espace de noms et l'assembly si besoin.



Ajouter à votre projet, les références à team foundation



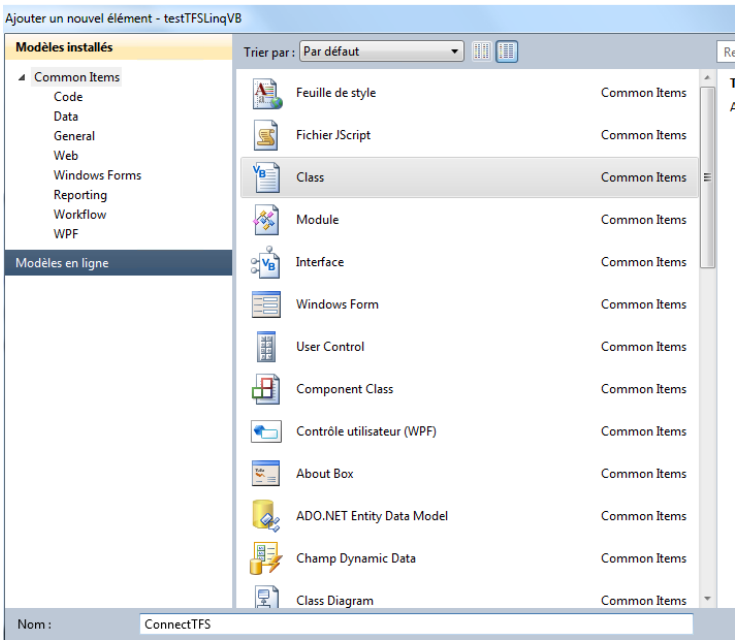
Dans votre projet, modifiez la « Form » en ajoutant un bon vieux bouton et un champ saisissable



Changez la propriété nom de l'EDIT en « `TextboxID` »

Laissez là cette form pour le moment.

Allez dans le projet et ajoutez un nouvel élément, classe VB et nommez là « `ConnectTFS` » .

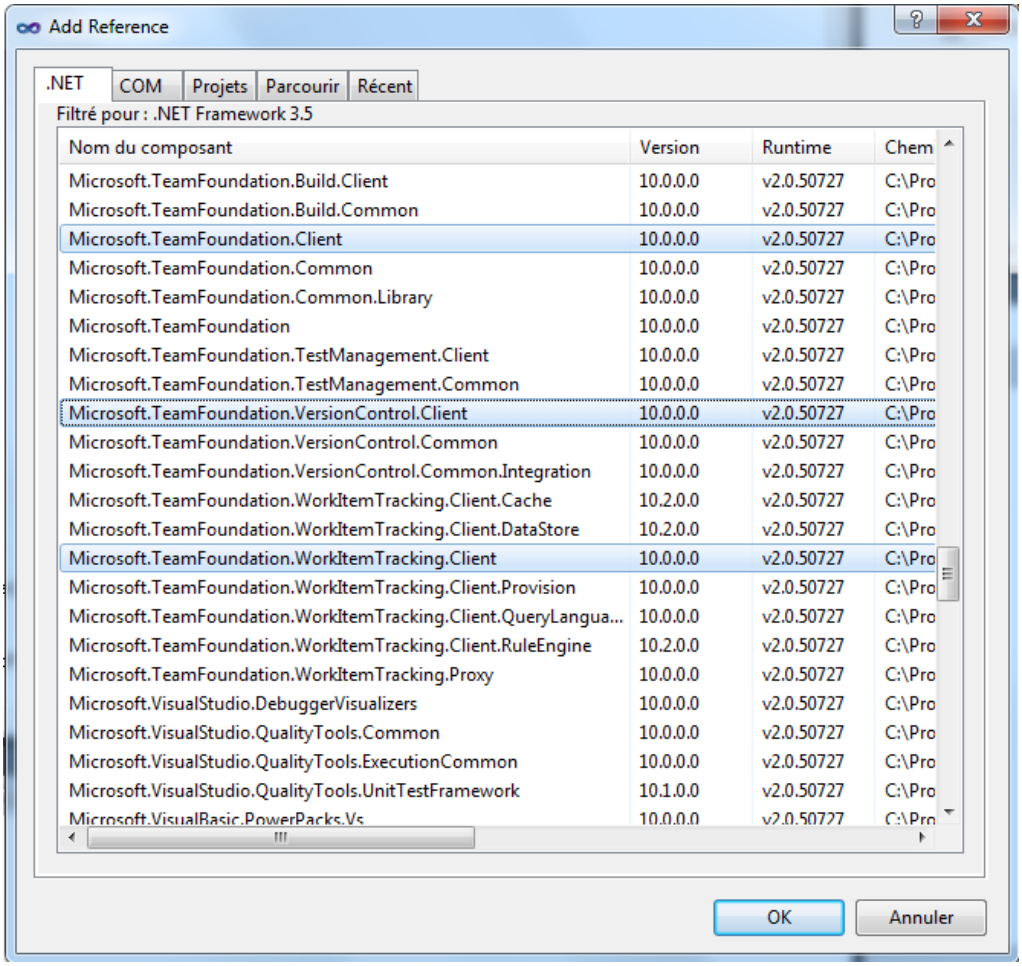


Cette classe va gérer la connexion avec TFS et offrira les quelques méthodes que l'on implémentera.

En entête de cette classe, 3 imports pour Team Foundation Server :

```
Imports Microsoft.TeamFoundation.Client
Imports Microsoft.TeamFoundation.WorkItemTracking.Client
Imports Microsoft.TeamFoundation.VersionControl.Client
```

Vous trouvez ces références dans le .Net



La classe a une entête définie comme ci-dessous. C'est la déclaration des objets privés.

```
Public Class ConnectTFS
    Private tfsUri As Uri
    Private oTFS As TfsTeamProjectCollection
    Private oTeamProjectCollection As TfsTeamProjectCollection
    Private oVCS As VersionControlServer
    Private oWorkItemStore As WorkItemStore
    Private oWorkItem As WorkItem
    Private oLinks As LinkCollection
    Private oChangeSet As Changeset
    Private nIdTFS As Integer
```

La méthode du constructeur New() est implémenté pour se connecter à un serveur TFS et à son DataStore. On en profite pour créer une connexion aussi à la partie du contrôle de sources avec « oVCS ». Cela nous permettra de lire les informations venant des éléments publiés par les développeurs liés à un élément de travail.

Bien évidemment, vous pouvez vous faire un autre constructeur avec un paramètre pour l'URL du serveur TFS !

```
    Public Sub New()
        Try
            tfsUri = New Uri("http://srvtfs:8080/tfs/DefaultCollection")
            oTFS = New TfsTeamProjectCollection(tfsUri)

            ' Connect to the server and the store.
            oTeamProjectCollection = New TfsTeamProjectCollection(tfsUri)
            oVCS = oTFS.GetService(GetType(VersionControlServer))
        Catch ex As Exception
            Exit Sub
        End Try
    End Sub
```

Du coté de la "Form", l'action sur bouton sera en un premier temps codé avec une instance de cette classe.

En entête de la classe « Form1 » les imports sont définis comme ceci …

```
Imports System.Web
Imports System.Xml
Imports System.Data
Imports System.ComponentModel
Imports testTFSLinqVB.ConnectTFS
```

Le code du bouton est pour le moment réduit à créer l'instance de la classe ConnectTFS. Cette dernière va donc se connecter au serveur TFS lors de sa création. Si vous passiez l'url en paramètre, rien ne vous empêcherait de faire plus d'une connexion à des serveurs différents.

```
    Private Sub Button1_Click(sender As System.Object, e As System.EventArgs)
Handles Button1.Click

        Dim oTestTfs As New ConnectTFS()
    End Sub
```

La classe « `ConnectTFS` » comporte une propriété « `ID` » qui nous permettra de nous positionner sur un élément de travail.

```
    Public Property ID As Integer
        Get
            ID = nIdTFS
        End Get
        Set(value As Integer)
            If (value <= 0) Then Exit Property
            nIdTFS = value
            Try
                oWorkItemStore = oTeamProjectCollection.GetService(Of
WorkItemStore)()
                oWorkItem = oWorkItemStore.GetWorkItem(nIdTFS)
            Catch ex As Exception
                Exit Property
            End Try
        End Set
    End Property
```

Arrivé là, on va enrichir la classe « ConnectTFS » d'une méthode nous permettant de lire un élément de travail grâce à son ID. La méthode est nommée « GetTFSItem » et retourne si tout va bien le contenu de la donnée de l'élément de travail. Cette méthode est publique puisqu'on l'offre aux autres classes qui auront instancié « ConnectTFS ». On lui passera le nom de l'item que l'on veut lire et en option l'ID de l'élément de travail. Cette option permettrait de s'affranchir de faire deux lignes de code dans les appelants.

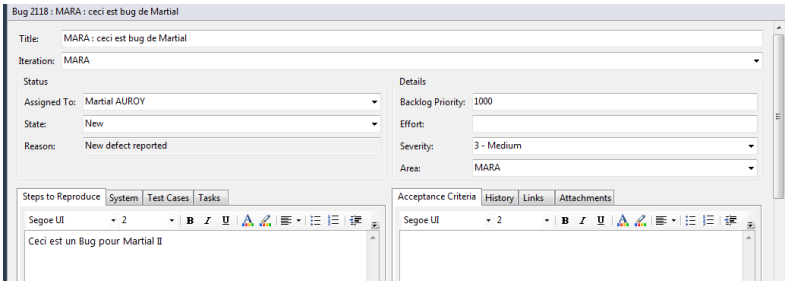
```
    Public Function GetTFSItem(ByVal sDataElementName As String, Optional ByVal
nID As Integer = 0) As String
        GetTFSItem = ""
        If (nID > 0) Then Me.ID = nID
        Try
            If (Not oWorkItem Is Nothing) Then GetTFSItem =
oWorkItem.Fields(sDataElementName).Value.ToString()
        Catch ex As Exception
            GetTFSItem = "Unable to read Item " + """ + sDataElementName +
"""
            Exit Function
        End Try
    End Function
```

Maintenant, on peut changer le code de la Form pour afficher le contenu d'un champ d'un element de travail. Le code est comme ceci :

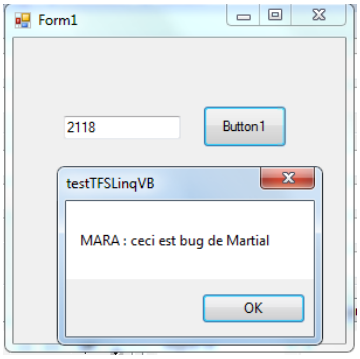
```
    Private Sub Button1_Click(sender As System.Object, e As System.EventArgs)
Handles Button1.Click

        Dim oTestTfs As New ConnectTFS()
        MsgBox(oTestTfs.GetTFSItem("Titre", Int(TextboxID.Text)))

    End Sub
```

Vous pouvez essayer, exécutez et choisissez un numéro d'élément de travail pour voir apparaître son titre. Par exemple, le bug 2118 :

Ce que vous avez dans TFS :

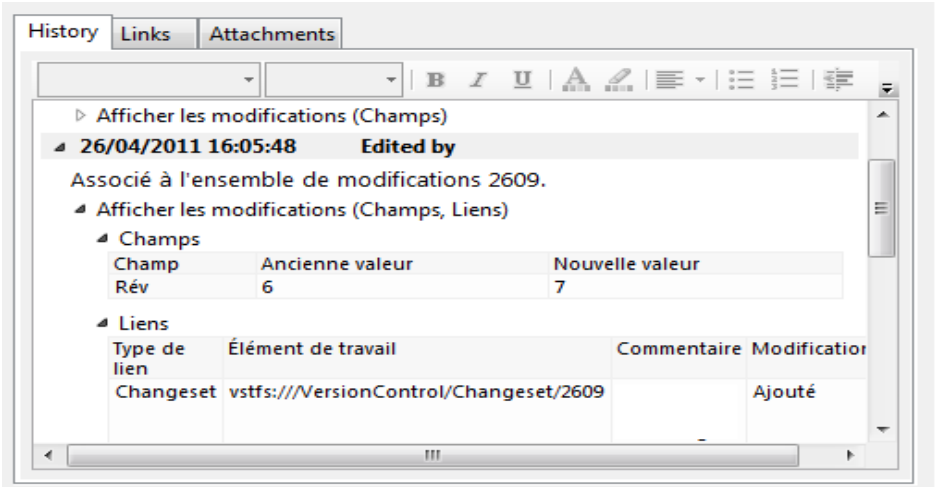


Et ce que vous avez avec notre application :



Historique

Certaines informations sont liées à la gestion des sources et au « VersionControl ». Le Version contrôle gère un autre ID unique , dans l'exemple ci-dessous l'élément de travail a l'ID 1330 alors que les « changeSet » sont liés à un ID 2609.



Ces informations ne sont pas accessibles directement par notre première méthode GetTFSItem, pour cela, on va se faire une nouvelle méthode dans la classe « `ConnectTFS` » que l'on appelera « `GetLinksComments` ».

Cette méthode va rechercher dans les liens de l'élément de travail, ceux de type « Fixed in changeset », soit ceux liés à des publications de code attachés à un élément de travail. Lorsqu'un

développeur publie des modifications, il a le choix de mettre un commentaire. On va récupérer ces commentaires et les afficher.

La fonction est codée comme ceci :

```
Public Function GetLinksComments(Optional ByVal nID As Integer = 0) As String
        GetLinksComments = ""
        If (nID > 0) Then Me.ID = nID
        If (oWorkItem Is Nothing) Then Exit Function
        Dim oURI As Uri
        oLinks = oWorkItem.Links
        For Each oIT In oLinks
            If (oIT.ArtifactLinkType.Name = "Fixed in Changeset") Then
                oURI = New Uri(oIT.LinkedArtifactUri.ToString())
                oChangeSet = oVCS.ArtifactProvider.GetChangeset(oURI)
                GetLinksComments += oChangeSet.Owner + " wrote """ +
oChangeSet.Comment + """" + vbCrLf
            End If
        Next
    End Function
```

A partir de l'objet « workItem » en cours, on parcourt les liens. Pour chaque lien de type « Fixed in changeset », on récupère le « Uri » qui contient le chemin de l'élément attaché ;
« `vstfs:///VersionControl/Changeset/2609` » dans notre exemple.

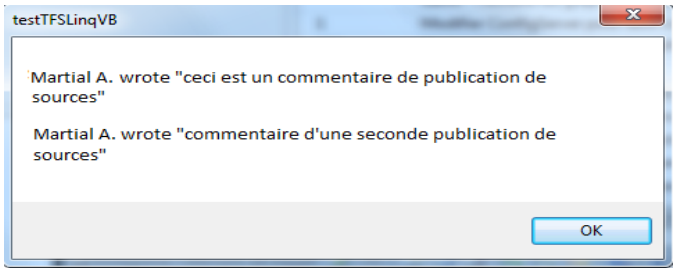
On alimente un objet de type « ChangeSet » que l'on peut alors interroger avec ses méthodes. Dans l'exemple on concatène le nom de celui qui a publié la modification avec le commentaire qu'il a déposé. La méthode retourne la concaténation de ces informations.

On retourne à la FORM

On adapte la méthode du bouton en y ajoutant un message contenant ces commentaires :

```
    Private Sub Button1_Click(sender As System.Object, e As System.EventArgs)
Handles Button1.Click

        Dim oTestTfs As New ConnectTFS()
        MsgBox(oTestTfs.GetTFSItem("Titre", Int(TextboxID.Text)))
        MsgBox(oTestTfs.GetLinksComments())

    End Sub
```

Le résultat donne ceci :



Conclusion

L'exemple nous permet de nous connecter à Team Foundation Server 2010, de lire les données des éléments de travail et de parcourir l'historique des informations attachées à la publication de documents attachés à l'élément de travail.

Bon tests.