Utilisation du WebService Partner.wsdl sous Visual Studio 2005

I Introduction

Ce tutoriel va nous permettre de réaliser une petite application nous permettant d'extraire quelques données issues d'un environnement Salesforce.com. Nous allons utiliser le service Web Partner.wsdl car celui-ci peut-être utilisé dans un contexte général alors que le web Service Enterprise.wsdl est lié à l'environnement sur lequel on l'a téléchargé. Nous allons voir à travers ce tutoriel comment intégrer ce service web à l'environnement de développement Visual Studio 2005. Nous allons ensuite développer une petite application qui va nous permettre d'extraire des données issues de l'objet Compte afin de les intégrer dans un tableau Excel. Donc vous apprendrez à travers ce tutoriel à utiliser des références, vous apprendrez de même à utiliser le langage d'extraction de données spécifique à l'environnement Salesforce.com qui est le SOQL. Même si celui-ci est bien moins puissant que le SQL, vous verrez que celui-ci s'en inspire très largement.

II Obtention du web Service Partner.wsdl

J'ai eu recours à ce service web dans le cadre du développement d'une petite application permettant de réaliser un « sales pipeline » pour mon directeur commercial. Je dois dire que ce service web fournit toutes les fonctions nécessaires à la connexion à l'environnement Salesforce.com, toutes les fonctions permettant d'extraire ou de modifier ou de supprimer des données. Ce webservice est téléchargeable sur votre environnement de production Salesforce.com. Si vous n'avez pas d'environnement de production, je vous invite vivement à souscrire à la developer Edition et ce même si vous disposez d'un environnement de production. En effet il est toujours préférable de disposer d'une version de test pour effectuer les tests et pour ne pas perturber l'environnement principal. Pour le télécharger, il suffit d'aller dans la rubrique :

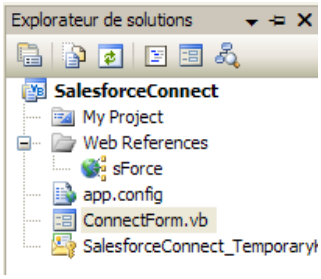
(Configuration (Setup) -> Configuration de la sous application-> Développer->API

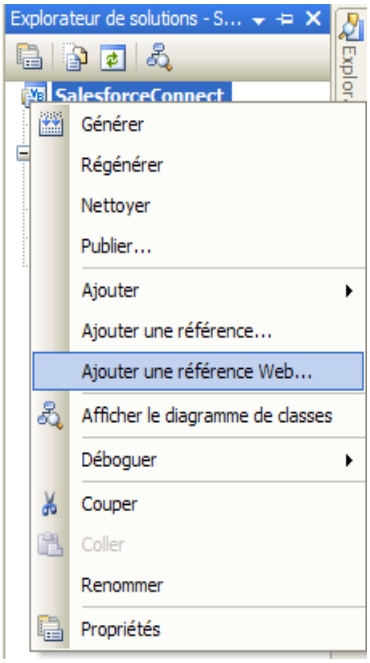
Ensuite télécharger le fichier xml en cliquant droit sur le lien Partner WSD, spécifiez de même un emplacement où copier le fichier.

A noter que le fichier téléchargé comporte l'extension xml, il faut le renommer en lui changeant son extension sous la forme .wsdl. Cette opération est nécessaire pour que l'environnement Visual Studio reconnaisse ce fichier comme étant une référence Web.

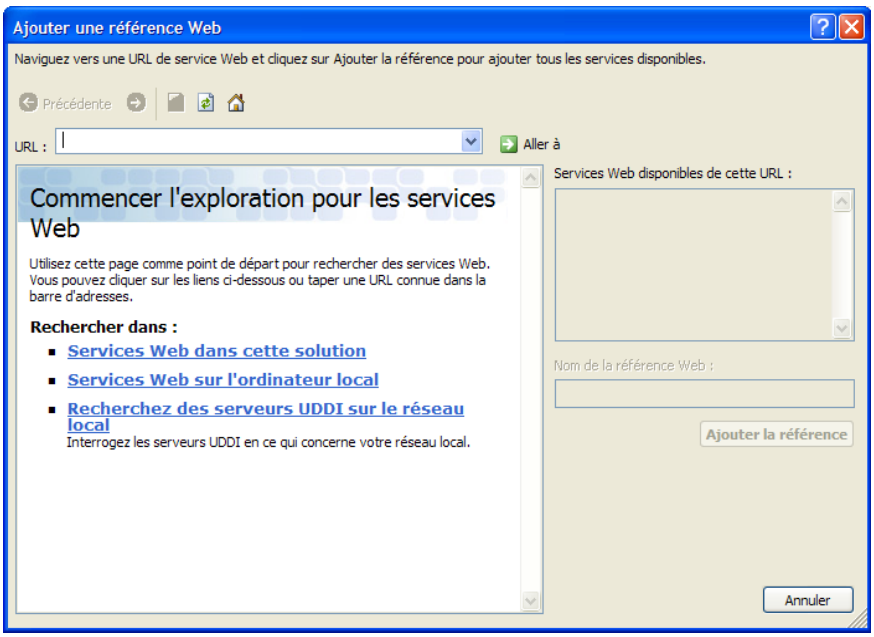
III Création du projet de l'application et Ajout de la référence Web

Maintenant que nous avons téléchargé notre fichier, l'étape suivante consiste à créer un projet de type Application Windows que vous nommerez comme bon vous semble. L'étape la plus importante consiste à ajouter la référence du webService à notre projet. Pour ce faire, il suffit de se déplacer dans l'explorateur de solution et de cliquer droit sur le fichier solution du projet un menu apparaît alors comme ceci :





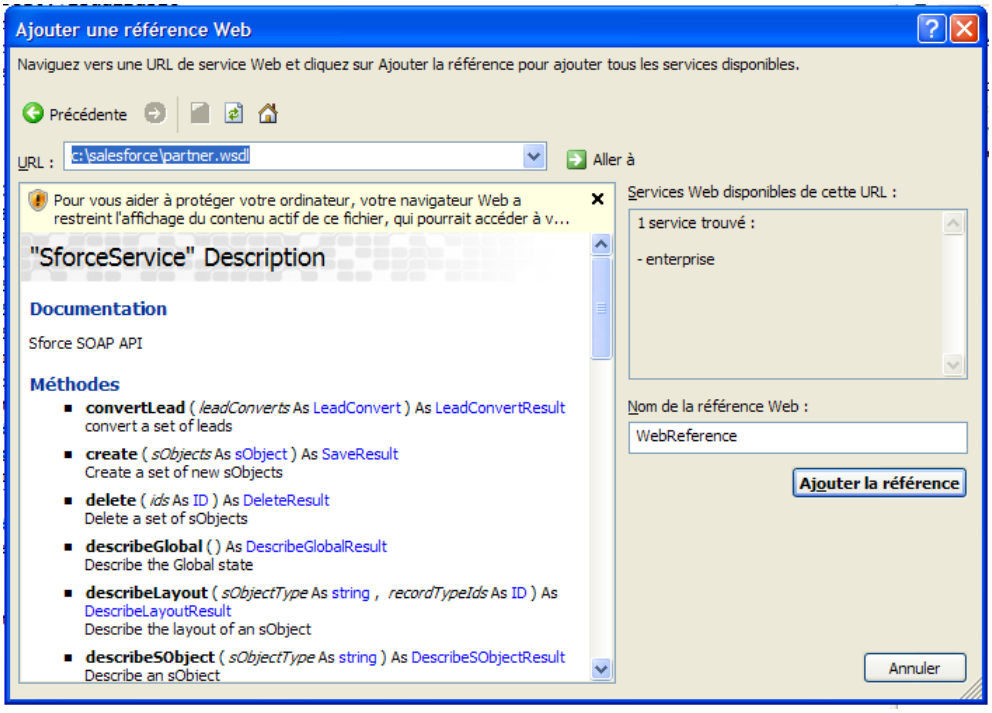
Une fois que vous avez validé l'option « Ajouter une référence Web… » une fenêtre apparaît à l'écran (voir page suivante) qui nous permet de rechercher notre référence web.



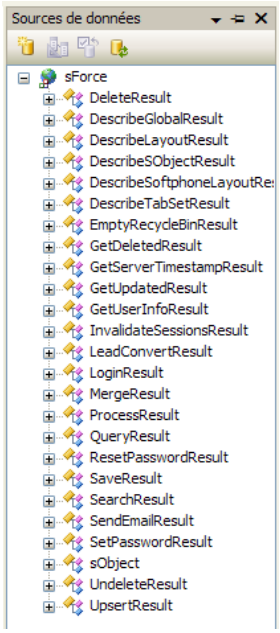
Or nous savons normalement où elle se trouve sur notre ordinateur. Il faut donc lui donner l'emplacement exact de ce fichier. Imaginons que vous l'avez copié dans le répertoire c:\Salesforce, il va falloir que vous lui donniez le chemin sous la forme suivante :

C:\salesforce\partner.wsdl.

Si l'emplacement spécifié est valide, Visual Studio vous indique qu'il a bien trouvé la référence souhaité et affiche les fonctions qui seront disponibles pour le développement de notre application. Il vous est possible ensuite de donner un nom à cette référence.



Votre référence est alors ajoutée à votre projet vous pouvez avoir un aperçu des objets ainsi que des fonctions mises à votre disposition en regardant la fenêtre « source de données », comme ceci :

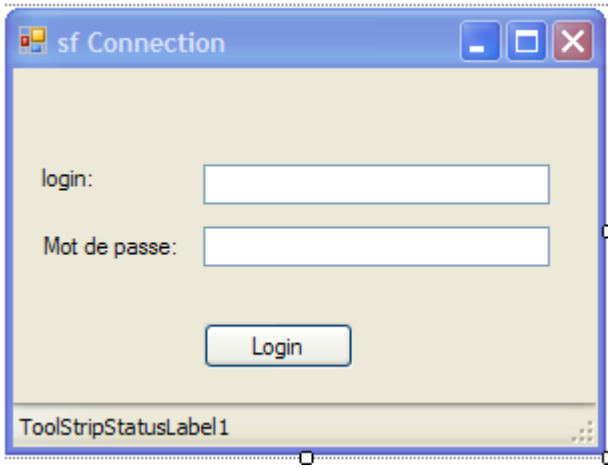


Votre référence web ayant été correctement intégrée au sein de votre projet, vous pouvez maintenant passer au codage de votre application.

III Le codage de l'application

1. Se connecter à l'environnement Salesforce.com

Avant de pouvoir extraire des données à partir de l'environnement de production Salesforce ou de l'environnement de test (sandbox). Il faut tout d'abord pouvoir se connecter à celui-ci. Le service web mis à notre disposition nous permet de concevoir cette fonction assez aisément. Tout d'abord pour pouvoir se connecter il vous faut les mêmes login et mot de passe utilisés pour vous connecter sur Salesforce.com. Il vous faudra de plus fournir un jeton de sécurité si vous vous connecter sur un ordinateur autre que celui que vous utilisez normalement. Donc l'interface graphique va, grosso modo, ressembler à ceci dans un premier temps :



Vous pouvez rajouter un composant Label et un composant Textbox pour gérer la saisie du Token.

Une fois l'interface mise en place, il faut définir une procédure Login qui va prendre en charge le paramétrage de la connexion. Avant de commencer il faut déclarer un certain nombre de variables nécessaires. Le service web met à notre disposition un objet loginResult qui nous permet de récupérer tous les paramètres de connexion. Voici la déclaration d'un tel objet :

```
Private lr As sForce.LoginResult
```

Pour réaliser cette fonction nous aurons besoin d'un second objet de type sForceService qui va nous permettre de fournir tous les paramètres de l'api Salesforce utilisés.

```
Private _Binding As sForce.SforceService
```

Il nous faut de même des variables permettant la gestion des paramètres de session :

```
Private _Host As String
```

```
Private _NextLoginTime As DateTime
Private _SessionLength As Integer
Private _SessionId As String
Private _ServerUrl As String
```

La variable ***_Host*** permet de stocker l'adresse de l'api utilisée pour la connexion à l'environnement Salesforce.com.

Vous pouvez initialiser cette variable au sein de la fonction ***OnLoad*** du formulaire que vous venez de créer :

```
Private Sub ConnectForm_Load(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles MyBase.Load
   Me._Binding = New sForce.SforceService
   Me._Host = "https://www.salesforce.com/services/Soap/u/10.0"
   …
End Sub
```

La variable ***_NextLoginTime*** permet de définir le moment où l'utilisateur va devoir se reconnecter après un temps déterminé en fonction de la variable ***_SessionLength***.

La variable ***_SessionId*** permet d'identifier la session ouverte lors de la connexion.

La variable ***_serverUrl*** nous permet de stocker l'adresse Internet du serveur sur lequel est installé notre environnement.

Voici la procédure ***Login()*** en détail, qui nous permet de nous connecter à notre environnement.

```
Public Sub Login()

        Dim sLogin, sMdp As String
        sLogin = txtLogin.Text
        sMdp = txtMdp.Text + Token

        Me._Binding.Url = Me._Host

        lr = Me._Binding.login(sLogin, sMdp)

        Me._NextLoginTime = Now().AddMinutes(Me._SessionLength)
        Me._Binding.Url = lr.serverUrl
        Me._Binding.SessionHeaderValue = New sForce.SessionHeader
        Me._SessionId = lr.sessionId
        Me._ServerUrl = lr.serverUrl




    End Sub
```

On renseigne dans un premier temps l'adresse de l'API utilisée au sein de l'objet ***sForceService*** par l'intermédiaire de sa propriété Url. Ensuite il est nécessaire d'initialiser l'Objet de type ***loginResult*** en utilisant la fonction login de l'objet ***sForceService*** qui prend en paramètre le login et le mot de passe. Ces paramètres sont donc stockés dans l'objet de type ***LoginResult***.

C'est à ce moment que l'on sait si la connexion s'est bien déroulée. Si lr est initialisé à une valeur différente de ***null*** la combinaison de login/mot de passe ou login/mdp/token est correcte. Dans le cas contraire une exception est levée.

Une fois la connexion établie, on réinitialise l'adresse de l'objet ***sForceService*** avec l'adresse retournée du serveur qui a servi à l'identification de l'utilisateur.

Une fois cette opération réalisé il faut renseigner les paramètres de la session ouverte. Ceci est effectué par l'intermédiaire de l'objet ***LoginResult*** qui fournit un id unique pour la session (par l'intermédiaire de la propriété ***SessionId***).

Pour savoir si la connexion est établie nous pouvons écrire une petite fonction :

```
Public Function IsConnected() As Boolean
        If Me._SessionId <> "" And Me._SessionId <> Nothing Then
            If Now() > Me._NextLoginTime Then
                IsConnected = True

            End If
        Else
            IsConnected = False
        End If
    End Function
```

La fonction renvoie donc un résultat de type booléen. Elle teste la validité de la variable _SessionId. Si celle-ci renferme une valeur et la date courante est supérieure à la l'heure prochaine de connexion alors la connexion est bien établie et est en cours.

La fonction suivante est utile lorsque l'on veut se connecter à partir d'un Id de Session.

```
Public Sub loginBySessionId(ByVal sid As String, ByVal sURL As String)
        Me._NextLoginTime = Now.AddMinutes(Me._SessionLength)
        Me._Binding.Url = sURL
        Me._Binding.SessionHeaderValue = New sForce.SessionHeader
        Me._Binding.SessionHeaderValue.sessionId = sid
        Me._SessionId = sid
        Me._ServerUrl = sURL


    End Sub
```

Cette fonction ressemble à la fonction de login vu précédemment, mis à part que l'on ne passe plus par l'objet de type ***LoginResult***, on renseigne directement les variable ***_SessionId*** et ***_ServerUrl*** par l'intermédiaire des paramètres fournis à la fonction.

Une fois que la connexion est établie avec le serveur il est possible de travailler sur notre environnement. Dans les sections qui suivent nous allons montrer comment extraire des données grâce à un langage de requêtes qui est le SOQL.

2. Récupérer le type d'enregistrement de notre organisation

Lorsque l'on configure notre environnement Salesforce, il est préconisé de créer des types d'enregistrement pour structurer les enregistrements. Et ce surtout lorsque l'on travaille sur l'Edition Entreprise et que celle-ci est partagée communautairement par plusieurs sociétés d'un même groupe (ce qui a été le cas me concernant).

Le webservice vous permet de récupérer l'id du type d'enregistrement de votre organisation. Pour ce faire Il va vous falloir dans un premier temps utiliser un objet qui va permettre de décrire tous les types d'objets utiliser dans votre organisation, c'est l'objet de type ***sForce.DescribeSObjectResult***. Dans un deuxième temps, il va falloir faire appel à un objet de type ***RecordTypeInfo.***

Pour utiliser ces types d'objet il faut initialiser les variables suivantes, comme ceci :

```
Dim dsr As sForce.DescribeSObjectResult

Dim rti As sForce.RecordTypeInfo
```

Voici la fonction qui permet de récupérer le type d'enregistrement de votre organisation :

```
Public Sub GetRecordType(ByVal sObject As String)
        Dim dsr As sForce.DescribeSObjectResult
        Dim i As Integer

        Dim rti As sForce.RecordTypeInfo

        dsr = Me._Binding.describeSObject(sObject)
        For i = 0 To dsr.recordTypeInfos().Length - 1
            rti = dsr.recordTypeInfos(i)
            If dsr.recordTypeInfos(i).name = "TNSF" Then
                rtId = rti.recordTypeId
            End If

        Next
```

On envoie comme paramètre à cette fonction une variable de type string qui prend comme valeur le nom du type d'objet Salesforce.com. auquel est rattaché le type d'enregistrement (par exemple « Account », ou encore « Opportunity »). Cette procédure initialise la variable ***rtId*** qui est de type String avec l'Id récupéré sur le serveur.

Une fois que la variable ***dsr*** est initialisée par l'intermédiaire de la fonction ***describeSObject*** prenant en paramètre le type d'objet à scruter, nous faisons une recherche sur les types d'enregistrements retournés par la structure de données ***recordTypeInfos***, jusqu'à ce que l'on atteigne le type d'enregistrement recherché.

On peut alors récupérer l'ID du type d'enregistrement par l'intermédiaire de la propriété ***RecordTypeId*** de l'objet ***RecordTypeInfo***.

3. Extraction des données sur le serveur

L'extraction des données sur le serveur se fait par l'intermédiaire de requête en utilisant le langage SOQL (***sForce Object Query Language***) qui est semblable à SQL, mais beaucoup plus limité. En effet il est impossible de faire des jointures, des tris, etc…

Voici un exemple permettant d'extraire les noms de tous les comptes qui sont rattachés à notre organisation :

```
select name from account where RecordTypeId=" + "'" + rtId + "'
```

Bien entendu avec Visual Basic il va falloir utiliser cette requête comme valeur associée à une propriété d'un objet permettant d'extraire des données. Pour ce faire il faut utiliser la fonction ***Query*** de l'objet de type ***sForceServices*** et qui renvoie un résultat de type ***sForce.QueryResult***. Il faut donc créer un objet de type ***sForce.QueryResult***

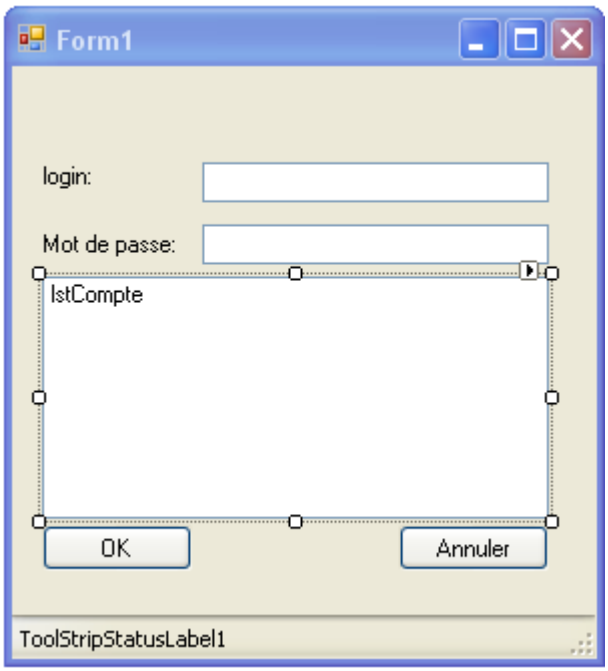
```
Dim qr As sForce.QueryResult
```

On peut donc ensuite utiliser la ligne de commande suivante pour extraire par exemple tous les noms de comptes se rapportant à un type d'enregistrement donné :

```
qr = _Binding.query("select name from account where RecordTypeId=" +
"'" + rtId + "'")
```

L'objet de type QueryResult renferme tous les noms de comptes, s'ils existent bien entendu. Nous pouvons donc les énumérer un par un. Pour ce faire nous avons besoin d'une boucle permettant de scruter tous ces enregistrements. Pour obtenir le nombre total d'enregistrements nous devons faire appel à la propriété ***QueryResult.Records.Length***.

Pour pouvoir visualiser le résultat il faut rajouter à la forme un composant de type ***ListBox*** que j'ai appelé lstCompte :



Pour récupérer les données concernant les noms de compte créons d'abord un objet de type sObject :

```
Dim Account As sForce.sObject
```

Donc pour récupérer chacun des enregistrements nous aurons recours à la boucle suivante :

```
Do
        For i = 0 To qr.records.Length - 1
            Account = qr.records(i)


            lstCompte.Items.Add(Account.Any(0).InnerText)
        Next
        If qr.done Then Exit Do
        qr = Me._Binding.queryMore(qr.queryLocator)


    Loop
```

Pour accéder à chaque enregistrement on prend donc les enregistrements un par un fournis par le tableau records :

```
Account = qr.records(i)
```

On récupère le nom de chaque enregistrement par l'intermédiaire de la propriété InnerText : `Account.Any(0).InnerText.`

Il est à noter que la fonction query ne retourne que les 2000 premiers enregistrements s'il s'avère que le nombre d'enregistrements à extraire est beaucoup plus élevé il nous faut recourir à la fonction ***QueryMore*** est lui rajouté en paramètre le ***queryLocator*** qui permet d'obtenir l'index du dernier enregistrement extrait dans la base ***Salesforce***.

Voici la fonction complète permettant de faire l'extraction des enregistrements de noms de compte :

```
Public Sub GetAccountData()
        Dim Account As sForce.sObject
        Dim qr As sForce.QueryResult
        'Account = New sForce.sObject()
        Dim i As Integer


        qr = _Binding.query("select name from account where RecordTypeId="
+ "'" + rtId + "'")


        Do
            For i = 0 To qr.records.Length - 1
                Account = qr.records(i)


                lstCompte.Items.Add(Account.Any(0).InnerText)
            Next
            If qr.done Then Exit Do
            qr = Me._Binding.queryMore(qr.queryLocator)
```

```
        Loop

    End Sub
```

4. Obtenir l'ID de type d'enregistrement d'un objet

Afin d'obtenir le type d'enregistrement d'un objet Salesforce, il faut déclarer les variables suivantes :

```
Dim dsr As sForce.DescribeSObjectResult
Dim i As Integer

Dim rti As sForce.RecordTypeInfo
```

Nous créons donc des instances d'objets DescribeSObjectResult et RecordTypeInfo.

La variable dsr va nous permettre d'avoir une vue globale de tous les éléments utilisés dans environnement de production Salesforce pour un objet donné:

```
dsr = Me._Binding.describeSObject(sObject)
```

La fonction describeSObject prend en paramètre le type d'objet Salesforce à scruter (« Account », « Lead », « Opportunity », etc…).

Pour scruter tous les types d'enregistrements se rapportant à l'objet que l'on vient de passer en paramètre, il faut utiliser le tableau `dsr.recordTypeInfos()`. Il renferme toutes les informations concernant les types d'enregistrements.

Si l'organisation dans laquelle vous travailler comporte plusieurs types d'enregistrements se rapportant à l'objet scruté il va falloir rechercher celui qui correspond à vos besoins, pour ce faire il faut donc balayer tous les types d'enregistrement :

```
For i = 0 To dsr.recordTypeInfos().Length - 1
      rti = dsr.recordTypeInfos(i)
      If dsr.recordTypeInfos(i).name = nom_recordType Then
          rtId = rti.recordTypeId
      End If

Next
```

ON obtient finalement l'ID du type d'enregistrement souhaité par l'intermédiaire de la propriété ***recordTypeId***.

Voilà la fonction complète :

```
Public Sub GetRecordType(ByVal sObject As String)
        Dim dsr As sForce.DescribeSObjectResult
        Dim i As Integer

        Dim rti As sForce.RecordTypeInfo

        dsr = Me._Binding.describeSObject(sObject)
        For i = 0 To dsr.recordTypeInfos().Length - 1
            rti = dsr.recordTypeInfos(i)
            If dsr.recordTypeInfos(i).name = "TNSF" Then
```

```
                rtId = rti.recordTypeId
            End If

        Next


    End Sub
```

IV. Conclusion

Vous avez pu, par l'intermédiaire de ce tutoriel avoir un aperçu des possibilités qu'offre le service web partner.wsdl fourni dans l'environnement Salesforce. Celui-ci permet donc de mettre au point des applications assez facilement permettant d'extraire des données mais aussi les mettre à jour par exemple. Un exemple d'application est l'Apex Data Loader. Je m'en suis servi personnellement pour développer une application permettant d'élaborer des Sales pipeline pour le service commercial de ma société.