Envoi d'e-mail avec ASP.Net (Version VB.Net)

Sommaire

Introduction

Une fonctionnalité qui peut être très intéressante à implémenter dans une application est l'envoi d'e-mail, que cela soit automatisé ou manuel. ASP.Net permet de réaliser des choses relativement complexes sans pour autant nécessiter de grandes connaissances.

Au cours de cet article nous allons donc voir comment envoyer des e-mails grâce à ASP.Net. Les explications seront volontairement très détaillées pour permettre même aux débutants d'envoyer des e-mails par ASP.Net.

Les exemples seront donnés en Visual Basic.Net et le logiciel utilisé sera Visual Studio.Net, mais ces exemples pourront aussi fonctionner avec WebMatrix moyennant de très petites adaptations.

1. Présentation de System.Web.Mail

Il s'agit du namespace utilisé par .Net pour l'envoi d'e-mail dans les applications. Il contient 3 classes qui ont des rôles complémentaires :

- **MailMessage** : permet de créer un e-mail, en définissant ses différentes propriétés.
- **MailAttachement** : permet de créer des pièces jointes aux e-mails
- **SmtpMail** : permet la communication avec le serveur SMTP et l'expédition de l'e-mail.

Ce namespace n'autorise que l'envoi d'e-mail, et ne permet pas la consultation de boîtes aux lettres, pour cela il faut faire appel à d'autres composants pour se connecter aux serveurs, le protocole utilisé n'est alors plus SMTP mais POP ou encore IMAP.

2. Envoi d'un mail de base

Dans cette partie nous allons voir comment envoyer un e-mail de base, c'est-à-dire sans pièce jointe et au format texte.

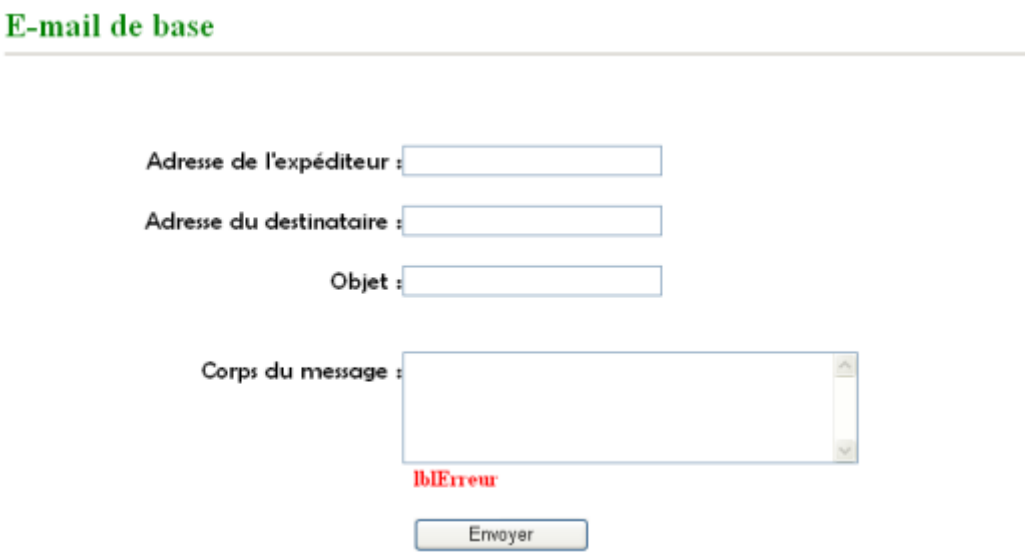
2.1. Création de l'interface

Nous allons rapidement créer une interface pour l'envoi d'e-mail, elle sera volontairement simpliste. Je l'ai réalisée avec Visual Studio mais vous pouvez bien évidemment la faire avec n'importe quel IDE. Dans ce WebForm il nous faudra les éléments suivants :

- 3 TextBox simple ligne
- 1 TextBox multiligne

- 1 bouton

Voici une capture d'écran montrant à quoi va ressembler notre interface:



Pour plus de simplicité et si vous voulez utiliser directement le code de cet article je vous recommande de nommer les différents contrôles comme moi, de haut en bas voici les ID à attribuer :

- tbExpediteur
- tbDestinataire
- tbObjet
- tbMessage
- lblErreur
- btEnvoyer

Une fois cette interface réalisée nous pouvons débuter l'écriture du code.

2.2. Ecriture du code

En premier il faut importer le namespace permettant de créer et d'envoyer des e-mails, il s'agit de System.Web.Mail :

```
Imports System.Web.Mail
```

Dans le cas de cet exemple le mail sera créé et expédié après avoir cliqué sur le bouton *envoyer*, il faut donc que le code soit exécuté lorsque l'utilisateur déclenche l'événement _click sur le bouton btEnvoyer. Notre code sera donc inséré dans le code qui suit :

```
Private Sub btEnvoyer_Click(ByVal sender As System.Object, _
  ByVal e As System.EventArgs) Handles btEnvoyer.Click

End Sub
```

Il faut maintenant créer une instance de MailMessage, ainsi nous allons créer un objet email possédant un certain nombre de propriétés.

```
Dim email As New MailMessage()
```

Une fois que cela est fait il faut définir les prorpiétés de l'e-mail. Ces propriétés sont en fait le destinataire, l'expéditeur, la priorité l'objet et le corps du message lui-même. Voilà comment nous allons les définir :

```
email.From = tbExpediteur.Text
email.To = tbDestinataire.Text
email.Subject = tbObjet.Text
email.Body = tbMessage.Text
```

Ainsi quand l'utilisateur cliquera sur le bouton *envoyer*, l'expéditeur, le destinataire, l'objet et le corps du message seront récupérés dans les diverses TextBox de notre page.

Il faut aussi définir l'adresse du serveur SMTP que l'on va utiliser, dans le cadre de cet exemple il faut un serveur autorisant l'Open Relay, c'est-à-dire qu'il accepte d'envoyer des e-mails sans authentification préalable. Pour cela il est vivement conseillé d'avoir son propre serveur, car les serveurs sur Internet offrant cette possibilité, sont de plus en plus rares, pour ne pas dire inexistants du fait des abus dans ce domaine créés par l'envoi de SPAMS. Le serveur SMTP est déclaré de la façon suivante :

```
SmtpMail.SmtpServer = "mon_serveur_smtp"
```

Voilà le code complet permettant d'envoyer un mail:

```
Private Sub btEnvoyer_Click(ByVal sender As System.Object, _
  ByVal e As System.EventArgs) Handles btEnvoyer.Click

Dim email As New MailMessage()

email.From = tbExpediteur.Text
email.To = tbDestinataire.Text
email.Subject = tbObjet.Text
email.Body = tbMessage.Text
email.Priority = MailPriority.High
SmtpMail.SmtpServer = "mon_serveur_smtp"

Try
    SmtpMail.Send(email)
Catch ex As Exception
    lblErreur.Text = ex.Message
End Try
End Sub
```

Remarque: *si vous utilisez votre propre serveur de messagerie et que vous souhaitez envoyer vos e-mails par le biais de cette technique vous devez configurer votre serveur de façon à ce qu'il accepte l'Open Relay pour l'adresse IP du serveur faisant tourner votre application, ainsi lorsque votre application voudra envoyer un e-mail sans être authentifiée l'envoi ne sera pas refusé.*

3. Utilisation d'un compte SMTP

Le petit défaut de la technique présentée plus haut est le fait qu'il faille utiliser son propre serveur de mail interne ou alors un serveur de mail configuré en Open Relay...

Mais heureusement une solution efficace existe, il suffit tout simplement de s'authentifier auprès du serveur de mail utilisé par le biais d'un compte de messagerie ayant les droits adéquats. C'est à dire par exemple votre compte de messagerie que vous utilisez tous les jours pour envoyer et recevoir des e-mails. Vous allez voir que cela n'est vraiment pas compliqué, trois lignes suffisent pour implémenter une authentification dans votre code. Dans les lignes qui viennent nous allons voir deux méthodes, une employant le fichier Web.Config pour stocker les informations relatives à votre compte SMTP, et l'autre non.

3.1. Avec utilisation du fichier Web.Config

Un peu comme pour la chaîne de connexion à un serveur de base de données il est possible de stocker des informations dans le fichier Web.config, l'avantage de cette solution est le fait qu'il n'est plus nécessaire d'écrire en dur dans le code l'identifiant et le mot de passe de messagerie. Ainsi un nouveau compte de messagerie peut être utilisé pour toute l'application en éditant seulement ce fichier de configuration. J'indique pour ceux qui ne le savent pas que le fichier Web.Config est un fichier XML contenant des paramètres de configuration de l'application à laquelle il est rattaché, il se trouve à la racine de celle ci.

Voilà les étapes présentant comment modifier votre application pour mettre en œuvre cette technique.

En premier, il faut éditer le fichier Web.Config, pour cela vous pouvez utiliser NotePad ou n'importe quel éditeur de texte. A l'intérieur des balises "appSettings" il faut ajouter trois clés, qui vont contenir l'adresse du serveur SMTP, le login et le password du compte de messagerie utilisé. Si les balises <appSettings> et </appSettings> n'existent pas il faut bien entendu les créer.

Voici un exemple de ce que cela pourrait donner:

```
<appSettings>
    <add key="SmtpServeur" value="smtp.mon_serveur.com" />
    <add key="SmtpUtilisateur" value="mon_login" />
    <add key="SmtpPassword" value="mon_password" />
</appSettings>
```

Après avoir édité le fichier de configuration il faut modifier le code de notre application en conséquence.

Il faut tout d'abord déclarer trois nouvelles variables de type chaîne:

```
Dim utilisateur, password, serveur As String
```

Ensuite il faut récupérer les valeurs stockées dans le fichier Web.Config:

```
utilisateur = ConfigurationSettings.AppSettings("SmtpUtilisateur")
password = ConfigurationSettings.AppSettings("SmtpPassword")
serveur = ConfigurationSettings.AppSettings("SmtpServeur")
```

A ce stade nous pouvons récupérer tout ce dont nous avons besoin pour nous authentifier sur le serveur de messagerie, la dernière modification consiste à donner ces informations à l'application

pour qu'elle puisse les transmettre au serveur SMTP. Le code à modifier et à ajouter apparaît ci-dessous.

```
email.Fields.Add("http://schemas.microsoft.com/cdo/configuration/smtpauthenticate", "1")
email.Fields.Add("http://schemas.microsoft.com/cdo/configuration/sendusername", utilisateur)
email.Fields.Add("http://schemas.microsoft.com/cdo/configuration/sendpassword", password)
SmtpMail.SmtpServer = serveur
```

La première ligne gère le type d'authentification, le chiffre 1 correspond à une authentification de base. La deuxième ligne précise le compte de messagerie à utiliser et enfin, la troisième ligne donne le mot de passe. Ce n'est pas plus compliqué que cela. Je me borne pour le moment à utiliser cette technique pour un cas général, mais vous devez savoir que vous pouvez aussi définir d'autres paramètres que ces trois là, en voici une liste non exhaustive.

- smtpserver
- smtpserverport
- sendingusing
- smtpaccountname
- sendemailaddress
- smtpuserreplyemailaddress
- smtpauthenticate
- sendusername
- sendpassword

Voici ci-dessous le code complet de cette solution :

```
Dim email As New MailMessage()
Dim utilisateur, password, serveur As String

utilisateur = ConfigurationSettings.AppSettings("SmtpUtilisateur")
password = ConfigurationSettings.AppSettings("SmtpPassword")
serveur = ConfigurationSettings.AppSettings("SmtpServeur")

email.From = tbExpediteur.Text
email.To = tbDestinataire.Text
email.Subject = tbObjet.Text
email.Body = tbMessage.Text
SmtpMail.SmtpServer = serveur

email.Fields.Add("http://schemas.microsoft.com/cdo/configuration/smtpauthenticate", "1")
email.Fields.Add("http://schemas.microsoft.com/cdo/configuration/sendusername", utilisateur)
email.Fields.Add("http://schemas.microsoft.com/cdo/configuration/sendpassword", password)

Try
    SmtpMail.Send(email)
Catch ex As Exception
    lblError.Text = ex.Message
End Try
```

Les lignes à ajouter dans le fichier Web.Config:

```
<appSettings>
   <add key="SmtpServeur" value="smtp.mon_serveur.com" />
```

```
    <add key="SmtpUtilisateur" value="mon_login" />
    <add key="SmtpPassword" value="mon_password" />
</appSettings>
```

3.2. Sans utilisation du fichier Web.Config

C'est la même chose que précédemment sauf qu'il faut mettre en "dur" dans le code le nom d'utilisateur, le mot de passe et l'adresse du serveur SMTP. Cette solution est certes légèrement plus rapide à mettre en oeuvre, mais elle est beaucoup moins souple, imgainez un peu le travail si vous devez modifier plusieurs pages à chaque changement de compte de messagerie... Enfin je tiens quand même à présenter cette solution, dans la mesure où elle peut être préférable pour certaines personnes.

```
email.Fields.Add("http://schemas.microsoft.com/cdo/configuration/sendusern
ame", "utilisateur")
email.Fields.Add("http://schemas.microsoft.com/cdo/configuration/sendpassw
ord", "mot_de_passe")
SmtpMail.SmtpServer = "votre_serveur_smtp"
```

Autre chose à faire, il faut bien évidement enlever les lignes suivantes qui ne servent alors plus à rien, en effet elles permettent de récupérer les informations d'authentification dans le fichier Web.Config

```
Dim utilisateur, password, serveur As String
utilisateur = ConfigurationSettings.AppSettings("SmtpUtilisateur")
password = ConfigurationSettings.AppSettings("SmtpPassword")
serveur = ConfigurationSettings.AppSettings("SmtpServeur")
```

Maintenant que nous savons envoyer des e-mails au format texte de manière authentifiée ou non il est temps d'améliorer la présentation de nos courriers. En effet un texte brut n'est pas forcément "beau", le langage HTML peut nous permettre de remédier à ce problème.

4. E-mail au format HTML

Nous allons apprendre dans cette section comment envoyer un e-mail au format HTML. Cependant il faut quand même garder à l'esprit que tous les clients de messagerie ne gèrent pas les e-mails au format HTML. Nous allons étudier ce cas dans le cadre de la génération automatique d'un e-mail par l'application, c'est-à-dire que par exemple le texte du message ne sera pas récupéré dans une TextBox mais sera inscrit en dur dans le code. Mais il est tout à fait possible et très simple de récupérer ce texte ou d'autres éléments de manière dynamique.

Passons à la pratique. Pour envoyer des mails au format HTML il faut tout d'abord ajouter la ligne suivante dans votre code :

```
email.BodyFormat = MailFormat.Html
```

C'est en effet extrêment complexe ! ;)

Puis il faut bien entendu donner le code HTML de votre courrier, cela peut s'avérer un peu plus difficile si vous ne connaissez pas ce langage. Mais une fois de plus il ne faut pas s'affoler, il s'agit d'un langage très simple. Voici un exemple de code HTML de base (on ne peut quasiment pas faire plus simple !).

```
<html>
<head>
<meta http-equiv="Content-Language" content="fr">
<meta http-equiv="Content-Type" content="text/html; charset=windows-1252">
</head>
<body>
<p>Mon premier e-mail au format HTML</p>
</body>
</html>
```

Si vous ne souhaitez pas éditer votre page HTML à la main vous pouvez bien sûr utiliser n'importe quel logiciel permettant de créer de tels documents, mais certains ne respectent pas suffisament les standards et il est possible que suivant votre client de messagerie le résultat varie. Cet article n'ayant pas pour objet la programmation HTML je vous renvoie à la section correspondante sur Developpez.com: **ici**

Au lieu de n'avoir que le texte brut dans le Body de notre e-mail nous allons aussi lui adjoindre tout le code HTML qui va créer la mise en forme. Mais il va y avoir une petite manipulation à réaliser pour que cela fonctionne. La propriété .body d'un e-mail étant une chaîne de caractères délimitée par des " il va y avoir des problèmes de syntaxe du fait que notre code HTML contient déjà des guillemets (**"**), la solution est de doubler tous les guillemets dans notre code HTML. Pour effectuer cela vous n'avez qu'à utiliser votre éditeur de texte favori avec la fonction *remplacer* (en remplacant les " par des ""). Voilà ce que va donner notre exemple:

```
<html>
<head>
<meta http-equiv=""Content-Language"" content=""fr"">
<meta http-equiv=""Content-Type"" content=""text/html; charset=windows-
1252"">
</head>
<body>
<p>Mon premier mail au format HTML</p>
</body>
</html>
```

Rien de bien méchant...

Une dernière petite difficulté est encore à surmonter. En effet notre code HTML devra se trouver sur une seule et même ligne dans le code de notre application, immaginez la longeur de la ligne si notre page est très chargée ! Heureusement une fois de plus il y a une solution, nous allons considérer chaque ligne de notre page comme une chaîne, et les concaténer. En VB.Net pour concaténer des chaines de caractères sur plusieurs lignes il faut encadrer chaque ligne par des **"** et ajouter **& _** à la fin de chaque ligne, sauf la dernière. Voilà ce que va donner notre exemple :

```
"<html>" & _
"<head>" & _
"<meta http-equiv=""Content-Language"" content=""fr""> " & _
"<meta http-equiv=""Content-Type"" content=""text/html; charset=windows-
1252""> " & _
"</head> " & _
"<body> " & _
"<p>Mon premier mail au format HTML</p> " & _
"</body>" & _
"</html>"
```

Il ne vous reste plus qu'à copier tout ça dans le code de votre application en le mettant dans la propriéré .body de votre mail, voilà le résultat:

```
email.Body = "<html>" & _
"<head>" & _
```

```
"<meta http-equiv=""Content-Language"" content=""fr"">" & _
"<meta http-equiv=""Content-Type"" content=""text/html; charset=windows-
1252"">" & _
"</head>" & _
"<body>" & _
"<p>Mon premier mail au format HTML</p>" & _
"</body>" & _
"</html>"
```

Comme pour les étapes précédentes voilà le code complet, celui-ci permet d'envoyer de manière authentifiée un e-mail au format HTML:

```
Dim email As New MailMessage()
Dim utilisateur, password, serveur As String
utilisateur = ConfigurationSettings.AppSettings("SmtpUtilisateur")
password = ConfigurationSettings.AppSettings("SmtpPassword")
serveur = ConfigurationSettings.AppSettings("SmtpServeur")

email.From = tbExpediteur.Text
email.To = tbDestinataire.Text
email.Subject = tbObjet.Text
email.BodyFormat = MailFormat.Html
email.Body = "<html>" & _
             "<head>" & _
             "<meta http-equiv=""Content-Language"" content=""fr"">" & _
             "<meta http-equiv=""Content-Type"" content=""text/html;
charset=windows-1252"">" & _
             "</head>" & _
             "<body>" & _
             "<p>Mon premier mail au format HTML</p>" & _
             "</body>" & _
             "</html>"

SmtpMail.SmtpServer = serveur

email.Fields.Add("http://schemas.microsoft.com/cdo/configuration/smtpauthe
nticate", "1")
email.Fields.Add("http://schemas.microsoft.com/cdo/configuration/sendusern
ame", utilisateur)
email.Fields.Add("http://schemas.microsoft.com/cdo/configuration/sendpassw
ord", password)


Try
    SmtpMail.Send(email)
Catch ex As Exception
    lblErreur.Text = ex.Message
End Try
```

Pour envoyer des e-mails comme avec n'importe quel client de messagerie il nous manque une chose, la pièce jointe, mais plus pour très longtemps...

Remarque : vous pouvez par exemple ajouter dans le corps du mail des variables contenant des valeurs récupérées dans une base de données, une utilisation de cela pourrait être la confirmation automatique par e-mail d'une inscription.

5. Pièce jointe avec l'e-mail

Dans cette partie nous allons voir comment adjoindre une pièce jointe à notre e-mail. Une fois de plus avec .Net cela va être un jeu d'enfant. Il suffit d'ajouter (au même endroit que les autres propriétés) le code suivant à votre application:

```
Dim chemin As String
chemin = "C:\Inetpub\wwwroot\developpez\fichiers\toto.txt"
email.Attachments.Add(New MailAttachment(chemin))
```

Nous pouvons voir qu'il faut définir une variable de type chaîne pour y stocker le chemin physique complet du fichier à mettre en pièce jointe, puis de passer ce chemin dans la ligne créant une instance de la pièce jointe. Mon code n'est pas très affiné, vous pouvez bien entendu récupérer le chemin de facon dynamique en utilisant par exemple le File Field HTML dans Visual Studio ou son équivalent dans WebMatrix.

6. Conclusion

On voit donc qu'il est très aisé d'envoyer des e-mails avec ASP.Net, les trois classes du namespace System.Web.Mail du .Net Framework permettent de paramétrer finement la création et l'envoi d'e-mail. On peut par exemple envoyer automatiquement des mails au format HTML, qui sont alimentés par une base de données, sans pour que cela nécessite des connaissances trop poussées. System.Web.Mail est donc très puissant et brille surtout par sa simplicité de mise en oeuvre, en effet même un programmeur .Net débutant pourra s'en sortir.

Merci beaucoup à Anomaly, Ditch, Abelman, Morpheus et David Pédehourcq pour la relecture.