

# E-Mails

---

## Sommaire

E-Mails .....	1
1 Introduction.....	2
2 Formatage d'e-mail simple.....	3
3 Envoyer un e-mail.....	5
3.1 Envoyer un e-mail de manière synchrone.....	5
3.2 Envoyer un e-mail de manière asynchrone.....	6
4 Formatage d'e-mail particuliers .....	9
4.1 Formatage HTML.....	9
4.2 Les fichiers joints .....	10
5 Conclusion .....	11

Dotnet-France Association

## 1 Introduction

Envoyer des mails est très courant depuis l'avènement d'internet, il ne se passe pas un jour sans que des milliards de messages transitent partout dans le monde.

Le Framework .NET implémente lui aussi un ensemble d'outils afin de créer et envoyer des e-mails. Envoyer des e-mails depuis vos applications est plus utile qu'il n'y paraît. Vous allez ainsi pouvoir envoyer directement sur une boîte e-mail, qu'elle soit locale, sur un serveur IIS, ou sur un serveur distant, toutes les erreurs de vos applications, ou bien les statistiques journalières de votre site web.



La plupart des outils permettant de créer et envoyer des e-mails se trouvent dans l'espace de nom `System.Net.Mail`. Nous verrons comment créer des e-mails en texte plein ou en HTML avec différentes priorités et des fichiers attachés. Nous verrons également comment configurer facilement leur envoi en utilisant le protocole SMTP.

## 2 Formatage d'e-mail simple

Quand vous envoyez un email, vous devez spécifier, bien sur, l'adresse du destinataire, mais également celle de l'expéditeur. A cela, vous devez ajouter un "objet de message" (le titre de votre email) puis le contenu que vous souhaitez envoyer.

Lorsque vous rédigez un email, vous pouvez le formater de différentes façons :

- En texte plein. Il ne permet pas d'envoyer autre chose que du texte, les contenus multimédias n'étant pas supportés.
- En mode HTML, beaucoup plus flexible que le mode texte. En effet, si la majorité des clients mails éditent ces messages en mode graphique (avec une interface qui interprète directement le code HTML), l'email est envoyé sous forme de document HTML. Il peut ainsi contenir des images et du texte riche.

Outre ces options de formatage, vous avez également la possibilité de choisir l'encodage de texte utilisé dans votre email (vous référer au chapitre 3), son niveau de priorité, des pièces jointes ...

Le Framework .NET vous permet de paramétrer toutes ces options à l'aide de différentes classes.

Pour commencer, nous créons l'email à proprement parler à l'aide de la classe `MailMessage`. En plus des surcharges du constructeur, vous pouvez utiliser propriétés ci-dessous :

Propriétés	Description
<code>AlternateViews</code>	Contient une collection d'objets <code>AlternateView</code> contenant les représentations alternative du corps du document.
<code>Attachments</code>	Contient une collection des objets joints à l'email.
<code>Bcc</code>	Contient une collection d'objets <code>MailAddress</code> reflétant la liste des destinataires en copie dont le nom ne sera pas mentionné dans la liste CC (Blind Copie Carbone).
<code>Body</code>	Permet d'obtenir ou de définir le corps de l'email. Cette propriété contient un objet de type String.
<code>BodyEncoding</code>	Obtient ou définit le codage de texte utilisé pour encoder la chaîne de caractère contenue dans <code>Body</code> .
<code>CC</code>	Contient une collection d'objets <code>MailAddress</code> reflétant la liste des destinataires en copie visible (Carbon Copie).
<code>From</code>	Obtient ou définit l'objet <code>MailAddress</code> utilisé pour spécifier l'adresse de l'expéditeur.
<code>Priority</code>	Contient l'une des valeurs de <code>MailPriority</code> indiquant la priorité du mail.
<code>IsBodyHtml</code>	Indique si le corps du message est au format HTML.
<code>Subject</code>	Obtient ou définit la ligne "Objet" du message.
<code>SubjectEncoding</code>	Obtient ou définit l'encodage utilisé pour la ligne "Objet".
<code>To</code>	Contient une collection d'objets <code>MailAddress</code> reflétant la liste des destinataires.
<code>ReplyTo</code>	Obtient ou définit l'adresse indiquée dans "ReplyTo".
<code>DeliveryNotificationOptions</code>	Permet de spécifier les options du serveur SMTP. Ce paramètre permet par exemple de spécifier au serveur SMTP de renvoyer un message à l'expéditeur si le message n'a pas pu être envoyé.

En relation avec cette classe, vous pouvez utiliser la classe `MailAddress` qui vous permet de modifier quelques propriétés liées aux adresses emails :

Propriétés	Description
<code>Address</code>	Obtient l'adresse utilisée dans l'instance de l'objet en cours.
<code>DisplayName</code>	Obtient l'alias utilisé dans cette adresse.
<code>Host</code>	Obtient la partie hôte de l'adresse utilisée (par exemple "dotnetfrance.fr" pour <a href="mailto:paul@dotnetfrance.fr">paul@dotnetfrance.fr</a> )

Là encore, plusieurs constructeurs sont à votre disposition pour vous faciliter la construction d'adresses email.

L'exemple suivant se contente de créer un email avec une adresse de destination et d'expédition. Il place également le mail en priorité haute. Nous configurons également les notifications de telle sorte à ce que le serveur SMTP nous notifie du succès et/ou de l'échec de l'envoi du mail :

```
'VB
Dim envoyeur As MailAddress = New MailAddress("paspaul@dotnetfrance.com",
"Paul")
Dim receveur As MailAddress = New MailAddress("paul@dotnetfrance.com",
"Paul")

Dim mail As MailMessage = New MailMessage(envoyeur, receveur)
mail.Body = "Corps du message. Ceci est important!"
mail.Subject = "Titre du message"
mail.Priority = MailPriority.High
mail.DeliveryNotificationOptions = DeliveryNotificationOptions.OnSuccess
Or DeliveryNotificationOptions.OnFailure

//C#
MailAddress envoyeur = new MailAddress("paspaul@dotnetfrance.com",
"Paul");
MailAddress receveur = new MailAddress("paul@dotnetfrance.com", "Paul");

MailMessage mail = new MailMessage(envoyeur, receveur);
mail.Body = "Corps du message. Ceci est important!";
mail.Subject = "Titre du message";
mail.Priority = MailPriority.High;
mail.DeliveryNotificationOptions = DeliveryNotificationOptions.OnSuccess
| DeliveryNotificationOptions.OnFailure;
```

### 3 Envoyer un e-mail

Nous avons vu comment créer un e-mail simple, nous allons maintenant voir comment l'envoyer. Par défaut le Framework .NET ne gère que le protocole SMTP pour l'envoi d'e-mail, c'est donc lui que nous allons voir en détail dans cette partie.

Nous allons tout d'abord décortiquer la classe `SmtpClient` qui permet de gérer le protocole adapté. En voici les principaux membres :

Membre	Description
<code>Credentials</code>	Permet d'obtenir ou de définir les informations d'authentification de l'expéditeur.
<code>EnableSSL</code>	Permet de spécifier si SSL doit être utilisé pour chiffrer la connexion au serveur SMTP.
<code>Host</code>	Permet d'obtenir ou de définir l'adresse du serveur SMTP.
<code>Port</code>	Permet d'obtenir ou de définir le port utilisé pour effectuer les transactions SMTP
<code>Timeout</code>	Permet de définir ou d'obtenir le délai d'attente avant d'annuler l'appel à <code>Send</code> .
<code>SendCompleted</code>	Evènement levé lorsque l'envoi d'un message par méthode asynchrone se termine.
<code>Send</code>	Permet d'envoyer un message. La méthode <code>Send</code> bloque le Thread appelant tant que le message n'est pas envoyé ou que le timeout n'est pas atteint.
<code>SendAsync</code>	Permet d'envoyer un message sans bloquer le Thread.
<code>SendAsyncCancel</code>	Permet d'annuler un envoi asynchrone.

#### 3.1 Envoyer un e-mail de manière synchrone

Pour envoyer un e-mail, la manière la plus rapide et la plus simple consiste à définir un serveur SMTP. Voici le code :

```
'VB
Dim smtp As SmtpClient = New SmtpClient("smtp.dotnet-france.com")
smtp.Send(mail)

//C#
SmtpClient smtp = new SmtpClient("smtp.dotnet-france.com");
smtp.Send(mail);
```

Nous avons créé un objet `SmtpClient` qui va se connecter au serveur SMTP de dotnet-france.com. Enfin grâce à la méthode `Send`, nous envoyons l'e-mail que nous avons précédemment préparé.

Comme nous l'avons vu dans le tableau plus haut, la méthode `Send` bloque le Thread appelant jusqu'à ce que l'e-mail soit bien envoyé, ou bien que le Timeout soit atteint.

Malgré tout, il se peut que dans de nombreux cas, votre serveur SMTP nécessite une authentification. Pour cela nous allons utiliser la propriété `Credentials` de l'objet `SmtpClient` et y placer un objet `NetworkCredential`. La classe `NetworkCredential` se trouve dans l'espace de nom `System.Net`.

```
'VB
Dim smtp As SmtplibClient = New SmtplibClient("smtp.dotnet-france.com")
smtp.Credentials = New NetworkCredential("Paul", "Mot_De_Passe")
smtp.Send(mail)

//C#
SmtplibClient smtp = new SmtplibClient("smtp.dotnet-france.com");
smtp.Credentials = new NetworkCredential("Paul", "Mot_De_Passe");
smtp.Send(mail);
```

Notre serveur SMTP réclamant que l'utilisateur soit authentifié pour envoyer un message, nous instancions un objet `NetworkCredential` avec notre nom d'utilisateur et notre mot de passe, et nous envoyons la référence à la propriété `Credentials`.

**Note** : Il faut savoir que la plupart des serveurs SMTP de FAI requièrent une authentification afin d'empêcher le SPAM. De même, certaines sociétés empêchent l'utilisation d'un autre serveur SMTP que celui de l'entreprise, faites donc attention.

Enfin, il est possible également que vous souhaitiez envoyer des messages en utilisant SSL (Secure Sockets Layers) afin de chiffrer la connexion vers le serveur. Si votre serveur SMTP le supporte, il vous suffit de mettre à true la propriété `EnableSsl` de `SmtplibClient` :

```
'VB
Dim smtp As SmtplibClient = New SmtplibClient("smtp.dotnet-france.com")
smtp.EnableSsl = True
smtp.Credentials = New NetworkCredential("Paul", "Mot_De_Passe")
smtp.Send(mail)

//C#
SmtplibClient smtp = new SmtplibClient("smtp.dotnet-france.com");
smtp.EnableSsl = true;
smtp.Credentials = new NetworkCredential("Paul", "Mot_De_Passe");
smtp.Send(mail);
```

**Attention** : Envoyer un e-mail paraît simple après avoir vu les quelques exemples précédents. Cela dit, vous allez vite vous rendre compte qu'une application gérant des e-mails se doit d'avoir une bonne gestion des exceptions. Aussi, je vous renvoie vers [MSDN](#) sur la page de l'espace de nom `System.Net.Mail` afin de connaître la liste des exceptions liées à `SmtplibClient`.

### 3.2 Envoyer un e-mail de manière asynchrone

Précédemment nous avons étudié les différentes manières d'envoyer un e-mail de manière synchrone. Cette façon de faire peut poser problème dans une application lourde car lors d'un envoi de manière synchrone, le thread appelant la méthode `Send` est totalement bloqué. Si une erreur survient, vous devrez attendre le timeout pour la réceptionner, et vous ne pourrez pas annuler une transaction. Nous allons donc détailler comment envoyer des e-mails de manière asynchrone.

Tout d'abord, vous devez créer votre objet `SmtplibClient` de la même manière que précédemment. Ensuite vous devez surveiller l'évènement `SendCompleted`, et enfin vous pouvez envoyer de manière asynchrone en utilisant la méthode `SendAsync`. Vous devrez enfin créer un callback qui sera appelé quand l'évènement `SendCompleted` sera levé. Voici le code correspondant :

```
'VB
Sub Main()
    Dim smtp As SmtplibClient = New SmtplibClient("smtp.dotnet-france.com")

    smtp.Credentials = New NetworkCredential("Paul", "Mot_De_Passe")

    AddHandler smtp.SendCompleted, New
SendCompletedEventHandler(AddressOf smtp_SendCompleted)

    smtp.SendAsync(mail, Nothing)

    smtp.SendAsyncCancel()
End Sub

Public Sub smtp_SendCompleted(ByVal sender As Object, ByVal e As
System.ComponentModel.AsyncCompletedEventArgs)
    If (Not e.Error Is Nothing) Then
        Console.WriteLine(e.Error.Message)
    ElseIf e.Cancelled Then
        Console.WriteLine("Envoie annulé")
    Else
        Console.WriteLine("Message envoyé")
    End If
End Sub
```

```
//C#
static void Main(string[] args)
{
    SmtplibClient smtp = new SmtplibClient("smtp.dotnet-france.com");

    smtp.Credentials = new NetworkCredential("Paul", "Mot_De_Pase");

    smtp.SendCompleted += new
SendCompletedEventHandler(smtp_SendCompleted);

    smtp.SendAsync(mail, null);

    smtp.SendAsyncCancel();
}

static void smtp_SendCompleted(object sender,
System.ComponentModel.AsyncCompletedEventArgs e)
{
    if (e.Error != null)
        Console.WriteLine(e.Error.Message);
    else if (e.Cancelled)
        Console.WriteLine("Envoie annulé");
    else
        Console.WriteLine("Message envoyé");
}
}
```

Tout d'abord nousinstancions notre objet `SmtplibClient` pointant sur le serveur SMTP de dotnet-France.com. Ensuite nous rajoutons les `Credentials` pour s'authentifier.

Ensuite nous définissons que lorsque l'évènement `SendCompleted` est levé, le callback appelle la méthode `smtp_SendCompleted`.

Enfin nous envoyons le mail et de suite après nous l'annulons, nous pourrons voir ainsi que le thread n'est plus bloqué lors de l'envoi.

Dans la méthode `smtp_SendCompleted`, nous gérons trois cas de figure : annulation de l'envoi, retour d'une erreur ou bien tout se passe normalement.

Voici ce qui est retourné lorsque nous annulons l'envoi :

Envoie annulé

Lorsque qu'il y a une erreur, par exemple mauvaises informations d'authentification :

Le serveur SMTP requiert une connexion sécurisée ou le client n'était pas authentifié. La réponse du serveur était : 5.7.1 Client was not authenticated

Et quand tout s'est bien passé :

Message envoyé

Dotnet-France Association

## 4 Formatage d'e-mail particuliers


Maintenant que nous savons comment envoyer un email basique, nous allons nous intéresser aux deux derniers points concernant les emails : Le formatage HTML et les pièces jointes.

### 4.1 Formatage HTML

Ce type de formatage de texte vous permet d'envoyer des messages plus riche et pouvant contenir du contenu multimédia (Comme le message HTML envoyé par MSDN si vous êtes inscrit à leur "newsletter")

De: <> À: <>  
 Objet: TR: Les JO de Pékin en Silverlight, le coach Visual Basic 2008 est arrivé, téléchargez le framework .NET 3.5 SP1


Inscrivez-vous aux autres newsletters | Désabonnez-vous | Gérez votre profil



Newsletter

Jeudi 21 août 2008

Retrouvez, deux fois par mois, l'essentiel de l'actualité de MSDN et des communautés de développeurs autour des technologies Microsoft.



Dernière ligne droite avant la rentrée, oui mais. Nous n'y sommes pas encore. En attendant, profitons des derniers jours d'août pour continuer sur une touche développement douce et sportive.

La première expérience que je vous propose c'est de suivre l'intégralité des JO de Pékin 2008 en direct sur [le site de France Télévision](#). La chaîne, en partenariat avec MSN.fr, a conçu [une solution Web](#) riche, immersive et sociale avec la technologie [Microsoft Silverlight 2](#). Vous profitez ainsi pleinement, et gratuitement, de 2200 heures de directs vidéo sur 15 chaînes. Ensuite, [téléchargez le fichier Excel spécialement développé par XaMaLa](#). Il vous permet d'élaborer un tableau de suivi des médailles, résultats et classements selon vos critères. les premiers ne sont pas ceux que vous croyez ! Enfin terminez ce triathlon avec [des jeux vidéos](#) développés en Silverlight et avec [Popfly](#) : football, tir à l'arc, boxe... il va y avoir du sport !

Avec MSDN, laissez les bons temps rouler !

Laurent ELLERBACH  
 Responsable relation avec les développeurs et l'enseignement supérieur  
 Microsoft France  
[laurent.ellerbach@microsoft.com](mailto:laurent.ellerbach@microsoft.com)

A la une

**Ressources**

- [Newsletter MSDN](#)
- [Archives newsletters](#)
- [Webcasts](#)
- [Téléchargements](#)
- [Flux RSS et Gadqet MSDN](#)
- [Evénements](#)
- [Forums](#)
- [Library en francais](#)
- [FAQ MSDN](#)
- [Archives des actualités MSDN](#)

**Services**


- [Formations & emplois](#)
- [Guide du site MSDN](#)
- [Microsoft Press](#)
- [Support](#)
- [Contactez-vous](#)
- [Centre d'évaluation MSDN](#)

Pour rédiger votre message en HTML, il n'y a pas d'autres alternatives que de mettre une chaîne de caractère contenant des balises HTML.

Cependant, il faut garder en tête que le contenu multimédia peut être bloqué par le client mail (Comme c'est le cas avec Windows Mail).

De: Clubic <> À: <>  
 Objet: Clubic Jour Express du Lundi 25 Août 2008

Certaines images ont été bloquées pour empêcher l'expéditeur d'identifier



Magazine informatique

Compara

La première Co

Dans l'exemple suivant, nous créons un email dont le corps contient une image. Cette image est en fait définie dans les vues alternatives et est accessible via la propriété cid (ContentID) "Image" (Seul le code ajouté/modifié a été mis ici).

Pour joindre du contenu multimédia (ici, une image), il vous faudra utiliser la propriété `AlternateViews` ainsi que des ressources liées en utilisant des objets `LinkedResources` :

```
'VB
mail.Body = "<html><body>Corps du message. Ceci est important!<br/><img
src='cid:Image' /></body></html>"
mail.IsBodyHtml = True

Dim alternate As AlternateView =
AlternateView.CreateAlternateViewFromString(mail.Body, Nothing,
MediaTypeNames.Text.Html)

Dim img As LinkedResource = New LinkedResource("D:\Documents\dotNet
France\Images\mail.jpg", MediaTypeNames.Image.Jpeg)
img.ContentId = "Image"
alternate.LinkedResources.Add(img)
mail.AlternateViews.Add(alternate)

//C#
mail.Body = "<html><body>Corps du message. Ceci est important!<br/><img
src='cid:Image' /></body></html>";
mail.IsBodyHtml = true;

AlternateView alternate =
AlternateView.CreateAlternateViewFromString(mail.Body, null,
MediaTypeNames.Text.Html);
LinkedResource img = new LinkedResource(@"D:\Documents\dotNet
France\Images\mail.jpg", MediaTypeNames.Image.Jpeg);
img.ContentId = "Image";
alternate.LinkedResources.Add(img);
mail.AlternateViews.Add(alternate);
```

La classe `MediaTypeNames` de l'espace de noms `System.Net.Mime` permet de spécifier le type MIME (Multipurpose Internet Mail Extension) de contenu multimédia lié.

## 4.2 Les fichiers joints

Pour joindre un fichier à vos emails, rien de plus simple : Il vous suffit de créer une ou plusieurs instances de la classe `Attachment` en y indiquant soit un nom de fichier, soit un flux vers un fichier.

Dans l'exemple suivant nous lions un fichier à l'email en utilisant le nom de fichier (seul le code ajouté est présent ici).

```
'VB
Dim fichier_joint As Attachment = New
Attachment("c:\Windows\System\user32.dll",
MediaTypeNames.Application.Octet)

mail.Attachments.Add(fichier_joint)

//C#
Attachment fichier_joint = new
Attachment(@"c:\Windows\System\user32.dll",
MediaTypeNames.Application.Octet);

mail.Attachments.Add(fichier_joint);
```

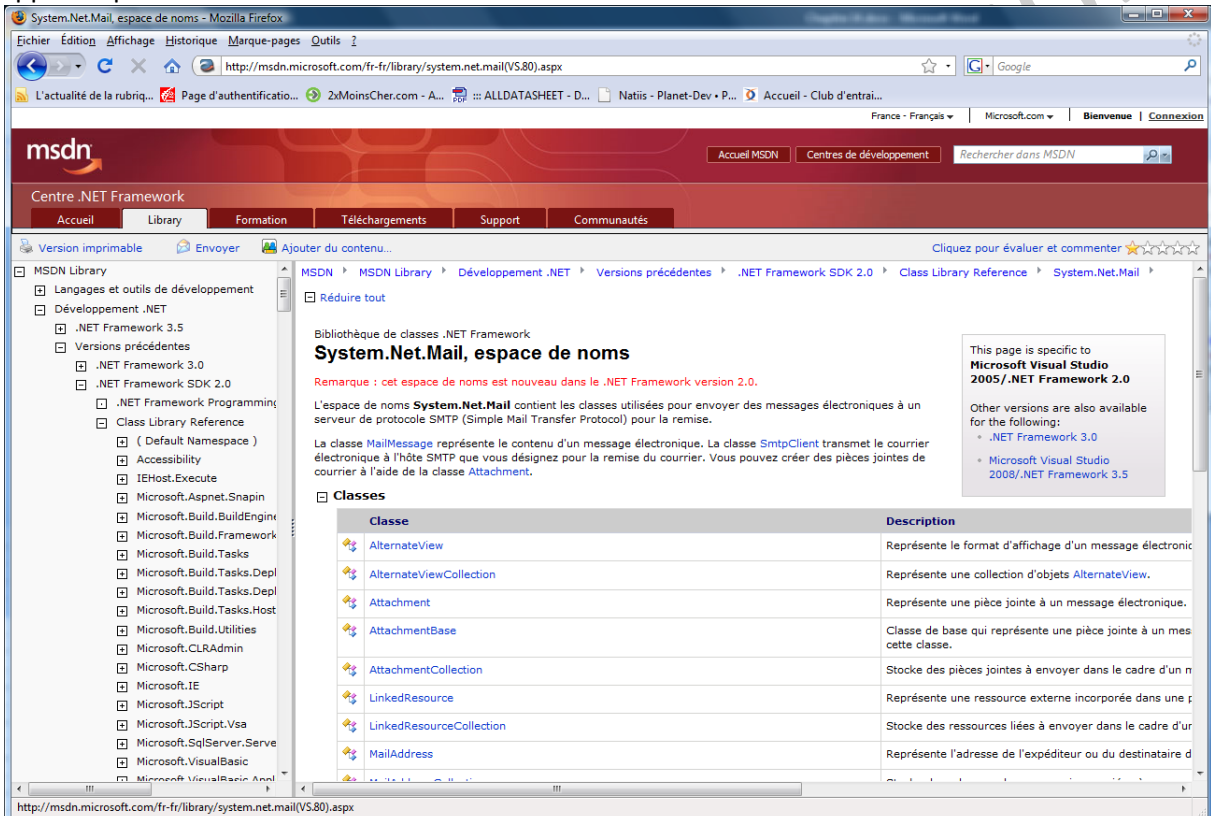
## 5 Conclusion

Ce court chapitre vous a montré comment créer des messages électroniques et comment les envoyer à l'aide du protocole SMTP. S'il est possible (et assez aisé) d'envoyer des messages, il est beaucoup moins pratique de les recevoir via un serveur POP3.

À la fin de ce chapitre vous devriez être capable de :

- Savoir créer des emails grâce à la classe `MailMessage` et connaître l'éventail de ses options.
- Savoir envoyer les emails créés grâce à la classe `SmtpClient` et connaître les possibilités de sécurisation.

Dans tous les cas, vous pourrez vous aider du [MSDN](http://msdn.microsoft.com) dans vos développements pour vous apporter plus d'informations.



The screenshot shows the MSDN website for the `System.Net.Mail` namespace. The page title is "System.Net.Mail, espace de noms". It includes a breadcrumb trail: MSDN > MSDN Library > Développement .NET > Versions précédentes > .NET Framework SDK 2.0 > Class Library Reference > System.Net.Mail. The main content area is titled "Bibliothèque de classes .NET Framework" and "System.Net.Mail, espace de noms". A note states: "Remarque : cet espace de noms est nouveau dans le .NET Framework version 2.0." The text explains that the `System.Net.Mail` namespace contains classes for sending electronic messages via SMTP. It mentions that the `MailMessage` class represents the content of an electronic message, and the `SmtpClient` class transmits the message to the SMTP host. A table of classes is provided:

Classe	Description
<code>AlternateView</code>	Représente le format d'affichage d'un message électronique.
<code>AlternateViewCollection</code>	Représente une collection d'objets <code>AlternateView</code> .
<code>Attachment</code>	Représente une pièce jointe à un message électronique.
<code>AttachmentBase</code>	Classe de base qui représente une pièce jointe à un message électronique.
<code>AttachmentCollection</code>	Stocke des pièces jointes à envoyer dans le cadre d'un message électronique.
<code>LinkedResource</code>	Représente une ressource externe incorporée dans un message électronique.
<code>LinkedResourceCollection</code>	Stocke des ressources liées à envoyer dans le cadre d'un message électronique.
<code>MailAddress</code>	Représente l'adresse de l'expéditeur ou du destinataire d'un message électronique.