

ASP.NET : Configurez vos applications et customisez leur comportement

Mis à jour le 09/07/2003

Par Elise Dupont

niveau : facile
durée : de 30 à 45 minutes

Droit de diffusion:

L'ensemble ou partie de ce document ainsi que le code mis à disposition, ne peut être diffusé sur d'autres sites Web sans l'autorisation au préalable de son créateur.

Avant Propos :

Cet article a pour but de vous expliquer l'architecture et la configuration des applications ASP.NET

Une application ASP.NET est une collection de ressources qui peuvent être appelées depuis un serveur Web, afin de procéder à une tâche. Cette application est installée dans un répertoire virtuel (virtual directory). En général, le serveur Web sera IIS. Dans ce cours, je traiterai des applications ASP.NET dans leur globalité : leur architecture, leur organisation, leur configuration, au niveau global (et pas au niveau d'un simple WebForm). Nous parlerons des sessions, des paramètres dynamiques qui peuvent être stockés dans un fichier de configuration, des événements, et bien sur de IIS.

Vous pouvez [télécharger gratuitement le code source VB.NET](#) qui s'inspire et illustre le code montré.

Sommaire:

1. Architecture générale

1.1. Répertoires virtuels, répertoires réels, fichiers

1.2. Travailler avec le fichier global.asax

2. Configuration

2.1. Notions élémentaires

2.2. Le format des fichiers de configuration

2.3. Accéder aux fichiers de configuration

1. Architecture générale

1.1. Répertoires virtuels, répertoires réels, fichiers

Les ressources qui composent une application sont nombreuses. On peut citer :

- des pages (.aspx)
- des modules
- du code exécutable
- des gestionnaires d'événements
- des services Web (Web services)
- des fichiers globaux à toute l'application

En fait, à part les Webforms et les Services Web, il existe bien d'autres ressources - situées à la racine de l'application - comme :

le répertoire \bin	contient les Assemblies chargées par l'application
le fichier global.asax	pour définir les évènements, objets et variables au niveau de l'application (et pas seulement dans un Webform)
le fichier web.config	contient les paramètres de configuration de l'application (connexions à une base de données par exemple)

Il n'y a qu'un seul répertoire \bin par application. Une fois que vous avez copié les assemblies dans le répertoire \bin, ces assemblies seront automatiquement chargées et leurs composants seront ainsi utilisables par votre application. ASP.NET permet un changement des composants du répertoire \bin très intéressant : chaque modification d'un fichier est détectée automatiquement, et vous n'avez pas besoin de redémarrer l'application ou IIS. Ainsi, votre temps de déploiement sont minimes, avec aucun temps mort pendant la mise à jour.

1.2. Travailler avec le fichier global.asax

Le fichier global.asax est un fichier optionnel, mais tellement utile dès que l'on dépasse plus de deux Webforms dans une application, que l'on devrait s'en servir presque systématiquement. Il permet de gérer les événements au niveau de l'application. Ainsi on peut implémenter les événements, objets, et variables globaux à l'application.

Dans le **global.asax** il y a notamment deux méthodes : **Application_Start** et **Application_End**. Ces méthodes sont exécutées une seule fois dans la durée de vie l'application (à ne pas confondre avec la durée de vie d'une session) :

- quand on fait la toute première requête à l'application.
- quand l'application est déchargée de la mémoire.

Pour les sessions, on a les même méthodes de début et de fin, mais cette fois ci elles sont appelées à chaque nouvelle session (nouveau navigateur, ou au bout du timeout définit (20 minutes d'inactivité par défaut)).

Imaginons que votre application aie pour but d'afficher et de tracer 3 informations :

- l'heure à laquelle l'application à démarré pour la première fois
- le nombre de sessions en cours
- le nombre de sessions déjà fermées

Dans votre page aspx vous aurez le code suivant :

VB.NET :

```
labelDateApp.Text = Application("DateApp")
labelNbSessionOuvert.Text = Application("NbSessionOpen")
labelNbSessionFerme.Text = Application("NbSessionClosed")
```

Et dans votre global.asax on gère ces données ainsi :

VB.NET :

```
Sub Application_Start(ByVal sender As Object , ByVal e As EventArgs)
    'initialiser les compteurs des variables d'application
    Application("DateApp") = Date .Now
    Application("CountSessionOpen") = 0
    Application("CountSessionClosed") = 0
End Sub

Sub Session_Start(ByVal sender As Object , ByVal e As EventArgs)
    Session.Timeout = 1
    'Dans une minute la session se fermera automatiquement et
    'les compteurs changeront (car l'événement Session_End aura été
    appelé)
    SyncLock Applock
        'bloque le code pour qu'il n'y aie qu'une seule exécution
    par session
        Dim i As Integer
        i = Application("CountSessionOpen")
        Application("CountSessionOpen") = i + 1
    End SyncLock
End Sub

Sub Session_End(ByVal sender As Object , ByVal e As EventArgs)
    SyncLock Applock
        'bloque le code pour qu'il n'y aie qu'une seule exécution
    par session
        Dim i As Integer 'diminuer le nombre de sessions ouvertes
        i = Application("CountSessionOpen")
        Application("CountSessionOpen") = i - 1
        'augmenter le nombre de sessions fermées
        i = Application("CountSessionClosed")
        Application("CountSessionClosed") = i + 1
    End SyncLock
End Sub
```

Mise en application : voila à quoi cela pourrait ressembler :



2. Configuration

2.1. Notions élémentaires

Le point fort d'ASP.NET c'est qu'un changement dans le fichier de configuration est simple et n'implique pas un redémarrage de votre application. Le système de configuration est très flexible. Il a été créé pour vous permettre de mettre à jour les paramètres sans accéder au serveur Web physiquement. Ainsi vous pouvez :

- définir des paramètres personnalisés
- étendre les paramètres de configuration de la machine, de l'application, ou même d'une partie de l'application

Tout ceci avec un impact minimal sur les autres applications web. Comme les fichiers de configuration sont stockés au format XML, cela permet d'éditer facilement les paramètres avec n'importe quel éditeur, et de répliquer facilement le contenu pour une autre application par exemple.

Le fichier de configuration d'une application d'appelle web.config et se situe à la racine du répertoire de votre application ASP.NET. Celui de la machine s'appelle **machine.config** et se situe dans le répertoire \config de votre installation de DOTNET.

2.2. Le format des fichiers de configuration

Le fichier machine.config contient des paramètres qui seront appliqués à toute la machine. Ces paramètres écrasent par défaut ceux définis par l'application, mais il est possible de spécifier qu'une application "l'emporte" sur les paramètres de la machine.

Tous les web.config héritent des paramètres qui ont été définis dans le machine.config. La partie qui concerne les applications ASP.NET est définie dans le noeud XML <System.Web>.

On peut citer un bon nombre de paramètres :

<Compilation>	Contrôle divers comportements de compilation, tel que le langage par défaut. <compilation defaultLanguage="c#" debug="true"/>
<CustomErrors>	Pour spécifier quelle URL le navigateur devra charger en cas d'erreur, afin d'avoir une page d'erreur personnalisée. <customErrors mode="Off" />
<Authentication>	Paramètre pour gérer la sécurité (configuration la validation d'un utilisateur). <authentication mode="Windows" />
<Authorization>	Paramètre pour gérer la sécurité (contrôle l'accès aux ressources). <authorization> <allow users="*" /> </authorization>
<Trace>	Pour fournir une trace de ce que fait l'application à un moment précis (configurable au niveau de l'application ou pour chaque page). <trace enabled="false" requestLimit="10" pageOutput="false" traceMode="SortByTime" >
<SessionState>	Configure l'état des sessions dans le module HTTP. C'est ici que l'on définit si la session doit être sans cookies, et combien de temps dure la session après une période d'inactivité (20 minutes par défaut). < sessionState cookieless="false" timeout="20" >
<Globalization>	Pour les paramètres de globalisation, tel que l'encodage (character encoding). Par exemple : UTF-8. < globalization requestEncoding="utf-8" responseEncoding="utf-8" >
<HttpModules>	Pour configurer des modules HTTP dans l'application. Les modules jouent un rôle important dans la sécurité et l'authentification.

	<pre><httpModules> <add name="MonModule" type="MonApp.MonNamespace.MonModule"/> </httpModules></pre>
<AppSettings>	<p>C'est ici que vous définissez vos paramètres personnalisés tels que sécurité, traces, états de session additionnels, ou chaînes de connexion à une base de données.</p> <pre><appSettings> <add key="SQLConnexion" value="chaîne de connexion"/> </appSettings></pre>

2.3. Accéder aux fichiers de configuration

La possibilité de changer les paramètres de votre application dynamiquement sans avoir à recompiler est souvent très importante. ASP.NET fournit de façons très simples d'accéder à ces informations par programmation.

La classe **System.Configuration.ConfigurationSettings** contient des membres très utiles pour accéder aux données stockées dans les différentes sections de votre fichier de configuration :

- GetConfig
- AppSettings

GetConfig récupère tous les paramètres définis par l'utilisateur d'une même section dans le web.config. Il prend un seul argument (le nom de section) et retourne un Objet de type System.Object. Il faut donc typer le retour avec NameValueCollection afin d'extraire les noms et données.

AppSettings a pour but de récupérer les paramètres de la section <appSettings> du fichier de configuration.

VB.NET :

'Afficher les connexions string définies dans le fichier de configuration '3 façons d'obtenir le même résultat :

```
'1. Avec GETCONFIG : avec le système des collections Dim MonDico As
System.Collections.Specialized.NameValueCollection Dim Param1 As
String MonDico = CType
(System.Configuration.ConfigurationSettings.GetConfig("appSettings")
, System.Collections.Specialized.NameValueCollection) lblConn1.Text
= CType (MonDico("Conn1"), String ) lblConn2.Text = CType
(MonDico("Conn2"), String )
```

```
'2. APPSETTINGS : avec le système des collections Dim MesAppSettings
As System.Collections.Specialized.NameValueCollection MesAppSettings
= ConfigurationSettings.AppSettings lblConn1.Text =
MesAppSettings("Conn1") lblConn2.Text = MesAppSettings("Conn2")
```

```
'3. APPSETTINGS : en récupérant un à un chaque paramètre
lblConn1.Text =
System.Configuration.ConfigurationSettings.AppSettings("Conn1")
lblConn2.Text =
System.Configuration.ConfigurationSettings.AppSettings("Conn2")
```



L'ensemble ou partie de ce document ainsi que le code mis à disposition, ne peut être diffusé ailleurs sans autorisation préalable

elise.dupont@europe.com - www.dotnet-tech.com - 2003