ASP. NET 1.1 & les Web Forms - 1/5

Les formulaires ASP. NET constituent le cœur des applications Web ASP. NET. Nous allons voir ici comment ces Web Forms sont conçus et traités par ASP. NET.

ASP. NET a permis à Microsoft de remettre à plat le mode de programmation sur serveur. Mais l'architecture existante, avec le protocole http utilisé pour les échanges entre serveurs et navigateurs, pose un véritable problème : il ne faut renvoyer que du code HTML au navigateur (le client). Au final, on obtient un environnement vraiment remarquable, qui a séduit la plupart des développeurs web qui ont bien voulu l'examiner de plus près !
La programmation s'effectue à la façon des applications Windows "traditionnelles", avec une écriture de code correspondant à des évènements (ex. : clic sur un bouton, donc on exécute une fonction).

La page asp. Net

Les pages ASP. NET ont une extension aspx. Le. NET Framework fonctionne sur un système où IIS (Internet Information Service) est installé, ainsi que sur Apache. Lorsqu'un internaute appelle une page ASP. NET, celle-ci est redirigée par IIS vers le. NET Framework. Ces pages aspx peuvent se présenter de deux façons (sur le serveur) :

• Avec HTML et code intégrés dans la page, donc en un seul fichier portant l'extension. Aspx ;
• Avec un fichier pour le HTML et un fichier pour le code. On aura donc le fichier. Aspx, et un fichier. Vb liés. En général, si le fichier contenant le HTML s'appelle Default. Aspx, on va appeler le fichier contenant le code Default. Aspx. Vb (ou. Cs si on fait du C# par exemple... Vb c'est pour le Visual Basic. NET).

Mis à part le nombre de fichiers, chaque développeur est libre de choisir la méthode qu'il souhaite, l'architecture restant la même... La deuxième méthode s'appelle le Code Behind, et elle est la plus couramment utilisée parmi les développeurs professionnels, et par Visual Studio. NET (et Whidbey dans un futur proche).
Il peut aussi exister un troisième fichier associé, qui porte l'extension. Resx et qui contient des ressources que l'on va utiliser pour par exemple permettre d'internationaliser l'application web. Sa structure est en XML.

Pour revenir au code Behind, voici comment lier les fichiers. Aspx et. Vb. Il suffit de spécifier une directive appelée "Page" au sommet du fichier. Aspx comme ceci :

L'attribut "Language" spécifie le langage utilisé dans la page (ici VB. NET).
L'attribut "CodeBehind" spécifie le fichier contenant le code.
L'attribut "Inherits" spécifie le nom de la classe qui va gérer les évènements de la page Web (la classe est contenue dans le fichier. Vb).

Lors de la lecture du fichier. Aspx, le serveur IIS reçoit la demande, puis il la redirige vers ASP. NET. ASP. NET lit donc le fichier demandé, et grâce à la directive Page, il va créer un objet de classe (indiquée dans l'attribut "Inherits"), ici la classe Default.
Ensuite, les balises ASP. NET sont interprétées, et les autres balises, en HTML sont directement envoyées dans le flux HTML sortant. Puis le navigateur reçoit ledit flux HTML...

Les balises ASP. NET sont celles qui contiennent un attribut runat="server". Selon le type de la balise (image, lien, etc.), ASP. NET va traiter particulièrement la balise. Chaque balise ASP. NET doit être contenue entre deux balises FORM (formulaire), d'où le nom de Web Forms lorsqu'on parle d'une page ASP. NET. De même,

ASP. NET 1.1 & les Web Forms - 2/5

chaque balise ASP. NET est appelée Contrôle serveur.

Voici ce que ça donne dans Visual Studio :

Chaque balise de cette page, devra être associée à une variable dans le code. Le nom de la variable sera le même que celui qu'on a spécifié dans l'attribut "ID" du contrôle serveur. Son type va dépendre du type de contrôle serveur. Par exemple, nous allons ajouter un contrôle Label sur notre Web Form :

Son nom est "lblExemple". Par convention, et pour pouvoir se repérer plus facilement lors de la lecture du code, on essaye de spécifier le type du contrôle. Par exemple ici, les 3 lettres lbl signifient Label. De même pour un TextBox, on mettra les lettres txt, etc. Ce n'est pas une obligation, mais c'est plus pratique.

Donc dans le code, le concepteur Web Forms de Visual Studio va rajouter cette déclaration :

VB. NET :
Protected WithEvents lblExemple As System. Web. UI. WebControls. Label

C# :
Protected WithEvents System. Web. UI. WebControls. Label lblExemple

Nous pouvons voir que l'identificateur de la balise "lblExemple" correspond bien au nom de la variable, et que le type de la variable "System. Web. UI. WebControls. Label" correspond à la balise "asp : Label".

Pareillement, si nous ajoutons une image à notre Web Form, nous aurons la balise :

Et le code suivant :

VB. NET :
Protected WithEvents imgExemple As System. Web. UI. HtmlControls. Image

C# :
Protected WithEvents System. Web. UI. HtmlControls. Image imgExemple

Le nom de la variable correspond à l'identificateur "imgExemple" de la balise "image", et le type "System. Web. UI. HtmlControls. Image" correspond à la balise "image".

Donc lors de la création d'une Web Form, la page est chargée, et un objet de la classe associée est créé.

ASP. NET 1.1 & les Web Forms - 3/5

Chaque balise dela page est chargée, et un objet de la classe associée est créé, et la variable du même nom que l'identificateur de la balise y fait référence.

Ainsi, le code de la classe associée à la page peut intervenir sur celle-ci et sur chaque contrôle serveur, en utilisant les variables déclarées par le Concepteur Web Forms. On peut ensuite accéder à tous les attributs des balises HTML en utilisant les propriétés des classes de chaque contrôle. Par exemple, pour remplir notre Label "lblExemple" avec le texte "Hello", on va utiliser la propriété "Text", comme ceci :

VB. NET :
lblExemple. Text = "Hello"

C# :
lblExemple. Text = "Hello";

Les évènements

Le déclenchement des évènements et une caractéristique importante des Web Forms. Durant leur utilisation, la page et les contrôles serveurs déclenchent des évènements. Ces évènements peuvent être traités par l'application, ce qui lui permet d'écrire du code. Par exemple, si un utilisateur clique sur un bouton (contrôle Button), un evènement ButtonClick est déclenché. Cela permet à l'application de définir quelle procédure appeler, pour par exemple soumettre le formulaire, ou effectuer un calcul, ou encore masquer une partie de la page. C'est ce qu'on appelle le modèle de programmation sur évènements, classique lors des précédentes version de Visual Basic. Néanmoins ce modèle a certaines particularités dans le contexte d'une application web ASP. NET, qu'il faut bien comprendre...

On peut définir 4 principaux évènements pour une page. Ce sont :

• **Init** : l'objet de la classe associé à la page est créé, mais les contrôles de la page ne le sont pas.
• **Load** : se déclenche lorsque tous les contrôles de la page ont été créés.
• **PreRender** : déclenché lorsque le code HTML de la page va être généré.
• **Unload** : l'objet de la page va être déchargé de la mémoire, il faut donc libérer les ressources qui ont été utilisées par la page.

L'évènement PostBack

Une application Web ASP. NET réalise plusieurs échanges avec l'tuilisateurs, avec le même formulaire (la même page), contrairement aux applications Windows traditionnelles. Par exemple, nous avons un formulaire qui contient une grille qui va afficher le contenu d'une table d'une base de données, et un bouton pour pouvoir modifier ces résultats. La première fois que l'utilisateur va demander la page, l'application doit remplir cette grille (contrôle DataGrid), en interrogeant la base de données et en y copiant les données.

Puis lorsque l'utilisateur va cliquer sur un des boutons de la page, il envoie à nouveau le formulaire : la même page est renvoyée à nouveau. Mais le click sur le bouton a déclenché un evènement, afin que de modifier l'état de la grille pour par exemple permettre à l'utilisateur de la modifier.

Il faut donc faire la différence entre la première fois que le formulaire est envoyé, et les autres fois. C'est pour cela que la page dispose d'une propriété "IsPostBack", qui renvoie une valeur fausse lors du premier envoi, puis vraie lors des envois suivants. Cela permet à l'application de savoir si c'est la première fois ou non que

ASP. NET 1.1 & les Web Forms - 4/5

l'on envoie le formulaire, et elle peut donc effectuer une ou plusieurs actions en conséquence.

Avec ASP. NET, il n'y a pas que les contrôles Buttons qui peuvent envoyer un formulaire. On peut le faire avec un contrôle ButtonLink (un lien qui fait office de bouton), ImageButton (une image qui fait office de bouton) ... Mais aussi avec une case à cocher (checkBox ou radioButton), en mettant la propriété AutoPostBack sur True. Cette fonction est à utiliser avec raison, car l'envoi d'un formulaire provoque un aller/retour client serveur et cela prend un certain temps.

Le flux html

Le but final d'une page aspx est la génération de code HTML, pour que le navigateur client puisse le lire. Ce fut la principale difficulté à laquelle se sont heurtés les développeurs de Microsoft.

Ils y sont arrivés en faisant dériver tous les contrôles, même la page en elle-même, d'une classe de base : System. Web. Control. Celle-ci possède une méthode, "Render", qui est appelée lors de l'affichage du contenu d'un contrôle.

Deux classes principales dérivent de la classe Control. Ce sont les classes WebControl et HtmlControl. Ces classes sont celles des contrôles Web et des contrôles HTML. Ces deux ensembles différent par leur niveau d'abstraction. Les contrôles Web sont plus éloignés de la syntaxe de leur balise en HTML, tandis que les contrôles HTML en sont plus proches. Il se peut même qu'un contrôle Web corresponde à plusieurs balises HTML. Ces deux ensembles on néanmoins le même comportement lors de la génération du code HTML.

Pour revenir à la génération du flux HTML, chaque classe de contrôle adapte cette méthode, dont le code va générer le code HTML qui correspond à la balise auquel il est associé. Par exemple la classe "System. Web. UI. WebControls. Label" va générer une balise comprenant les attributs nécessaire à l'affiche dudit Label.

Le code HTML de la page est généré de façon récursive, c'est-à-dire que durant le traitement de la page, ASP. NET utilise la méthode "Render" de la page comme nous l'avons vu précédemment. Cette méthode génère le code HTML de la page, puis va explorer chaque contrôle, en appelant leur méthode "Render" respective. Puis chaque contrôle génére son code HTML, et ainsi de suite (certains contrôles serveur peuvent contenir d'autres contrôles).

The end

ASP. NET et les Web Forms amènent un nouveau souffle sur le web. Maintenant la programmation d'applications Web rentre dans l'ère du professionnel. Le développement est beaucoup plus aisé, sécurisé et maintenable de part la séparation du "design" de la page, et de son code, tous les langages sont orientés objet, et l'ont peut accéder à l'ensemble des classes du. NET Framework. Il ne faut pas oublier pour autant les standards du web pour que les applications fonctionnent correctement.
L'arrivée d'ASP. NET 2.0 et de Whidbey (Visual Studio 2005) va encore plus marquer cette nouvelle ère de la programmation. De nombreux assistants vont permettre aux développeurs d'avoir moins de code à saisir, etc. Pour l'instant en phase Bêta 1, cela s'annonce plus que prometteur...

Voilà, j'espère avoir bien présenté les Web Forms, et que les débutants comme les confirmés auront réussi à suivre le déroulement de cet article.

ASP. NET 1.1 & les Web Forms - 5/5

Je vous invite à visiter le site d'ASP. NET : http://www.asp.net/
Ainsi que la communauté CodeS SourceS - ASP/ASP. NET : http://www.aspfr.com/
N'oublions pas de mentionner C²i : http://www.c2i.fr/