ASP Microsoft .net

Etape par étape

3ème partie : Utilisation des variables de sessions, d'application et passage d'arguments à une url.

(Révision : 1 du 08/11/2004 – 9 pages)

Avertissement :

Ce document peut comporter des erreurs. Cependant, tout a été mis en œuvre afin de ne pas en inclure dans ce texte. Tout code qui trouve sa place ici a été testé au préalable.

Tables des matières :

Bibliographie

Gérard Leblanc, c# et .NET, ed. Eyrolles

MSDN

Dans le tome 2, nous avons créé un formulaire de login simple. Pour ce tome, nous utiliserons une telle page mais cependant elle sera plus légère puisqu'elle ne comprendra que deux textboxes : l'un permettant l'encodage du nom d'utilisateur, l'autre son mot de passe. Supposant que nous n'avons aucune connaissance en ADO.NET (utilisation des bases de données), nous stockerons les correspondances login/password en mémoire, dans un tableau à deux dimensions.

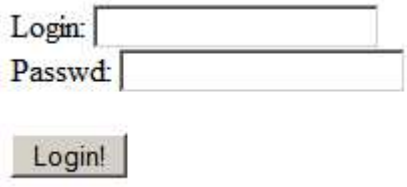
La page de login utilisée dans ce tome

Comme il a été dit, il y a donc deux textboxes et un bouton qui appellera la fonction chargée de vérifier la concordance des données encodées.

En voici la forme :

```
<%@ Page Language="C#" %>
<script runat="server">
        ...
</script>
<html>
<head>
</head>
<body>
   <form runat="server">
      Login:
      <asp:TextBox id="TbLogin" Runat="server"></asp:TextBox>
      <br />
      Passwd:
      <asp:TextBox id="TbPasswd" Runat="server"></asp:TextBox>
      <br />
      <br />
      <asp:Button id="BuLogin" onclick="Check" Runat="server"
Text="Login!"></asp:Button>
   </form>
</body>
</html>
```

Ce qui donne à l'affichage quelque chose de très simple :



Le but de cette page est de permettre à un utilisateur de se loguer sur notre site. Lorsqu'il sera logué, il sera connu sur toutes les pages qu'il visitera sur ce site. Il existe plusieurs solutions pour arriver à nos fins, la première étant le passage d'arguments à une url.

Passage d'arguments à une url

Souvent, lorsque l'on visite un site, nous voyons la syntaxe suivante pour l'url : *http://monsite.com/mapage.aspx?param1=val1&param2=val2.* Il s'agit d'appeler la page mapage.aspx se trouvant sur le serveur monsite.com en lui passant des paramètres dont les noms sont param1 et param2, leurs valeurs respectives étant val1 et val2 (celles sont toujours considérées comme étant des chaînes de caractères). Cette syntaxe est utilisée quelque soit le langage utilisé dans la page Internet.

Cette syntaxe est utilisée dans le traitement du clic sur le bouton de login. Cette méthode renverra l'utilisateur vers une page « MesInfos.aspx ». Cette page ne fera qu'afficher les informations inscrites dans les TextBoxes. Nous passerons ces informations en arguments dans l'url dans un premier temps.

Le code de traitement associé est dès lors le suivant :

```
<%@ Page Language="C#" %>
<script runat="server">

public void Check(object sender, EventArgs e)
{
        String[,] tab=
        {
                {"Ditch", "aspnet"},
                {"Test", "Test"}
         };

        int i=0;
        while (i<tab.Length/2) // Nb total d'elements / nombre de dimensions...
        {
                if (TbLogin.Text == tab[i,0].ToString() && TbPasswd.Text == tab[i,1].ToString())
                        Response.Redirect("MesInfos.aspx?L=" + TbLogin.Text + "&P=" + TbPasswd.Text);
                i++;
        }
        Response.Redirect("MesInfos.aspx");
}

</script>
<html>
…
</html>
```

La page « MesInfos.aspx » a la forme suivante :

```
<%@ Page Language="C#" %>
<script runat="server">

   void Page_Load(object sender, EventArgs e)
   {
      La.Text = "Nous sommes passés par le Page_Load() pour vérification()";

      if (Request.Params["L"]!=null)
         La.Text += "<br>Login: " + Request.Params["L"].ToString();

      if (Request.Params["P"]!=null)
         La.Text += "<br>Pass: " + Request.Params["P"].ToString();
   }

</script>
<html>
<head>
</head>
<body>
   <form runat="server">
      Page MesInfos<br><br>
      <asp:Label Runat=server ID=La></asp:Label>
   </form>
</body>
</html>
```

Comme on peut le constater, cette page n'est pas très compliquée. Qu'y trouve-t-on de nouveau ? Seulement une chose : un « **Request.Params["L"]** ». Ceci permet de récupérer des valeurs passées dans l'url de la page.

Ces variables ne sont pas initialisées d'office. Effectivement, parfois une erreur de frappe peut être insérée sans que l'on s'en appercoive. Il est donc important de vérifier que ces variables existent (ou plus exactement qu'elles ont été initialisées). Ceci est fait grâce au **if (Request.Params["L"] !=null) {...}**.

Dans notre exemple, si la page a reçu en arguments le login et le mot de passe de l'utilisateur, ceux-ci sont affichés dans le label prévu à cet effet. Dans le cas contraire, elle n'affiche rien si ce n'est le fait que nous sommes bien passés dans le Page_Load().

Avec ce simple exemple, nous venons de voir comment nous pouvions récupérer des paramètres au travers d'une url.

Les navigateurs Internet possèdent généralement un historique des urls, il est donc inconcevable de passer à chaque page le login et le mot de passe afin de vérifier que l'utilisateur a bien les droits. Nous allons donc améliorer ces pages afin d'en faire quelque chose de plus sécurisé, de plus facilement utilisable, … (c'est à dire ne plus passer ces valeurs dans l'url).

Pour ce faire, nous allons utiliser une autre technique : *les variables de session.*

Les variables de session

Les variables de session sont stockées en mémoire sur le serveur. Il peut y avoir autant de variables de session que l'on le désire, cependant il ne faut pas perdre de vue qu'avec l'accroissement du nombre de variables utilisées, ce sont souvent les performances qui sont en chute libre. Il faut donc trouver le juste milieu entre performances et facilité. Les informations utilisées de manière répétée, si elles n'ont pas un trop haut coût en mémoire, peuvent donc être stockée dans ce type de variables (par exemple : le UserName de l'utilisateur).

Ces variables sont accessibles du début à la fin de la session d'un navigateur, c'est à dire jusqu'à moment de non activité du navigateur plus le timeout. Une session est un espace mémoire/disque sur le serveur associé à un et un seul utilisateur/navigateur. Cette session possède un identifiant unique généralement stocké au niveau du navigateur. Les variables de session sont stockées sur le serveur et sont accessibles par toutes les pages web associées à cette session (mais uniquement pour cette session). Cet accès sera valable tant que le navigateur n'aura pas été fermé et tant que la durée d'inactivité d'un utilisateur ne dépassera pas le timeout (personnalisable, voir page 9) associé à la gestion de session.

Comme pour le passage d'arguments dans l'url, les variables ne doivent pas être déclarées. Il faut donc faire attention à l'orthographe du nom de nos variables. Ainsi, ***Session["Username"]*** et ***Session["Usrname"]*** ne désignent pas la même variable et il est parfois très dur de trouver ce type d'erreur, ce qui est un gros inconvénient.

Revenons à notre page. Afin de ne pas surcharger le serveur et sa mémoire, nous allons considérer que si la variable de session est initialisée, c'est que l'utilisateur est bien loggué. Dans le cas contraire, son identification a été un échec.

Les traitements créées précédemment deviennent ainsi :

```
public void Check(object sender, EventArgs e)
{
   String[,] tab=
```

```
                {
                    {"Ditch", "aspnet"},
                    {"Test", "Test"}
                };

            int i=0;
            while (i<tab.Length/2) // Nb total d'elements / nombre de dimensions...
            {
                if (TbLogin.Text==tab[i,0].ToString() &&
TbPasswd.Text==tab[i,1].ToString())
                {
                    Session["Username"] = TbLogin.Text;
                    break;
                }
                i++;
            }
            Response.Redirect("MesInfos2.aspx");
        }
```

Et pour la page d'information :

```
        void Page_Load(object sender, EventArgs e)
        {
            La.Text = "Nous sommes passés par le Page_Load() pour vérification()";

            if (Session["Username"]!=null)
                La.Text += "<br>Login: " + Session["Username"].ToString();
            else
                La.Text + = "<br>Personne de connecté";
        }
```

On l'a dit, ces variables ne peuvent être accédées que pour une session. On peut cependant imaginer vouloir partager des informations entre les différents navigateurs clients, comme par exemple la liste des utilisateurs connectés. Ces dernières variables sont des *variables d'application*.

Les variables d'application

La syntaxe et l'utilisation des variables d'application sont les mêmes que pour les variables de sessions si ce n'est que ces variables sont accessibles par toutes les sessions, il est donc possible de les faire communiquer.

Un exemple simple et fréquent que l'on trouve au travers du net est le nombre d'utilisateurs connectés et identifiées. Ceci peut-être facilement gérable à l'aide des variables d'application. Effectivement, il suffit d'ajouter 1 lorsque l'identité de l'utilisateur est vérifiée.

Dans la fonction Check() de la page de Login, nous allons donc simplement ajouter ceci : **Application["NbUsers"]++;** ou de manière plus efficace (on vérifie que celle-ci a déjà été initialisée)

```
if (Application["NbUsers"]!=null)
   Application["NbUsers"]= Convert.ToInt32(Application["NbUsers"])+1;
else
   Application["NbUsers"] = 1;
```

Dans la page d'informations, nous allons récupérer cette valeur à l'aide de la syntaxe utilisée pour les variables de session. Nous ajoutons donc

```
if (Application["NbUsers"]!=null)
   La.Text += "<br>NbUsers: " + Application["NbUsers"].ToString();
else
   La.Text += "<br>NbUsers: 0!";
```

Ce qui donne …

Page MesInfos

Nous sommes passés par le Page_Load() pour vérification()
Login: Ditch
NbUsers: 2

… lors de la première connexion. Ouvrons une nouvelle instance d'un navigateur internet. Lors que l'on arrive sur la page MesInfos2.aspx, on voit directement que le nombre d'utilisateurs en ligne est bien de deux. Fermons un navigateur et regardons ce nombre. Il est toujours de deux. La valeur n'a pas été décrémentée.

Ceci est dû à plusieurs choses. La première raison est le fait qu'une session n'est pas terminée dès la fermeture de l'instance du navigateur. Effectivement, il y a un timeout. Par défaut, celui-ci est de 20 minutes. Il est modifiable dans le Web.Config à l'endroit désigné par :

```
<sessionState  …
      cookieless="false"
      timeout="20"
 />
```

Autres techniques

Certaines personnes utilisent les cookies comme moyen pour passer des informations d'une page à l'autre. Cette technique a l'inconvénient de ne pas pouvoir être contrôlée par le développeur. Effectivement, il est possible de désactiver les cookies dans les navigateurs, …

Il est donc déconseillé d'utiliser de telles techniques.