

TP 5 : La gestion d'état

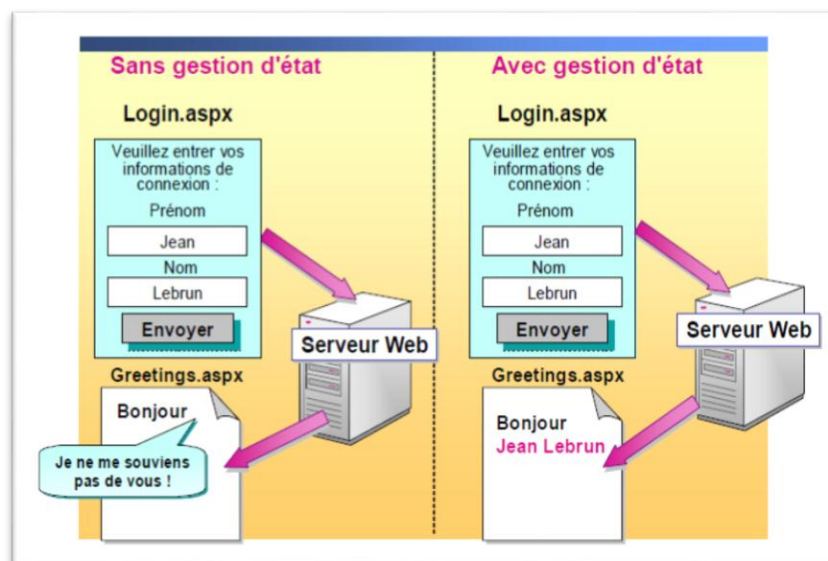
1. Définition :

Lors de l'affichage d'une page Web, il est possible de personnaliser cet affichage en fonction de l'utilisateur qui s'y connecte. Ainsi, on peut conserver des mots passés pour éviter des identifications trop fréquentes qui peuvent être lourdes à la longue. Il est également possible d'afficher les informations personnelles d'un utilisateur sur sa page d'accueil ou encore personnaliser le style du site selon son désir.

L'état est la capacité d'une application Web à conserver des informations utilisateur.

La connexion établie entre un utilisateur (l'ordinateur client) et un serveur Web est appelée une session. Les sessions peuvent couvrir plusieurs pages Web et sont suivies par la gestion d'état.

La gestion d'état est le processus qui vous permet de conserver les mêmes informations au sein de plusieurs demandes d'une même page Web ou de pages différentes.



ASP.NET offre les capacités de gestion d'état qui permettent d'enregistrer des informations sur le serveur, même lorsque l'utilisateur change de page, ce qui permet d'assurer la continuité des informations utilisateur (l'état) tout au long d'une visite sur un site Web.

Si l'état est conservé entre plusieurs pages, les informations d'origine fournies par les utilisateurs peuvent être réutilisées, ce qui signifie qu'ils ne doivent pas entrer les mêmes informations à chaque renvoi d'une page vers le serveur.

2. Type de gestion d'état :

ASP.NET offre deux types de gestion d'état qui permettent de conserver l'état lors des allers-retours vers le serveur. Le choix du type de gestion d'état dépend principalement de la nature de votre application Web.

Les deux types de gestion d'état sont les suivants :

- **Côté serveur**

Les options de gestion d'état côté serveur utilisent les ressources du serveur pour stocker les informations d'état. Ces options offrent une plus grande sécurité que celles côté client.

- **Côté client**

Les options de gestion d'état côté client n'utilisent pas les ressources du serveur pour stocker les informations d'état. Les options côté client offrent une sécurité minimale, mais accroissent les performances du serveur, car elles ne lui font pas appel pour conserver l'état.

Gestion d'état côté serveur	Gestion d'état côté client
État de l'application <ul style="list-style-type: none"> Les informations sont accessibles à l'ensemble des utilisateurs d'une application Web 	Cookies <ul style="list-style-type: none"> Le fichier texte enregistre des informations pour maintenir l'état
État de la session <ul style="list-style-type: none"> Les informations sont uniquement accessibles à l'utilisateur d'une session spécifique 	Propriété ViewState <ul style="list-style-type: none"> Conserve les valeurs entre plusieurs demandes pour la même page
Base de données <ul style="list-style-type: none"> Dans certains cas, utilisez la prise en charge de la base de données pour conserver l'état sur votre site Web 	Chaines de requête <ul style="list-style-type: none"> Informations ajoutées à la fin d'une URL

Ce chapitre traite uniquement la gestion d'état côté client. La gestion d'état côté serveur sera traitée dans le TP suivant.

3. La gestion d'état côté client :

La façon la plus performante pour la navigation est la sauvegarde d'information côté client. Pour ce faire, nous avons à notre disposition divers outils que nous expliquerons au cours de cette partie. Voici une liste des outils que nous allons présenter :

View State	On utilise les <i>View State</i> pour traquer les valeurs dans des contrôles. Il est possible d'ajouter des valeurs personnalisées à nos <i>View State</i> .
Cookies	Les <i>Cookies</i> sont un bon moyen pour récupérer des données sur toutes les pages d'un même site Web. Les valeurs sont stockées du côté client par le navigateur Web.
Query Strings	On peut transmettre des données d'une page à l'autre en les faisant passer directement dans l'URL, on appelle cela le <i>QueryString</i> .

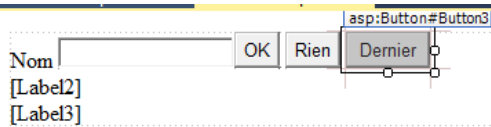
A. View State :

Vous avez sûrement remarqué que lorsque vous cliquez sur un *button* d'une page ASPX, les informations que vous avez rentré auparavant sont automatiquement conservées (sauf pour les éléments rajoutés en *CodeBehind*).

Par exemple si on met à jour un *label*, tous les rechargements de la page qui suivront conserveront cette dernière mise à jour. Ceci est possible grâce à la gestion d'état côté client d'ASP.NET et plus particulièrement aux *View State*.

La propriété *View State* nous fournit une collection d'objet permettant de conserver des valeurs entre plusieurs requêtes sur la même page. Quand une page ASP.NET est traitée, l'état courant de la page ainsi que de ses contrôles sont hachés en une chaîne de caractère puis sauvegardés dans la page un peu comme un champ caché en HTML (`<input type="hidden" />`).

Exemple :



```
protected void Button1_Click(object sender, EventArgs e)
{
    Label2.Text = "Votre nom est " + TextBox1.Text;
    ViewState.Add("Last", DateTime.Now.ToString());
    //Ajouter une entrée au Viewstate de la page.
}

protected void Button3_Click(object sender, EventArgs e)
{
    Label3.Text = ViewState["Last"].ToString();
    //Récupérer la valeur d'une entrée de ViewState de la page
}
```

4. Les cookies :

Les cookies constituent un moyen pratique de stockage des informations spécifiques à l'utilisateur dans les applications Web. Par exemple, lorsqu'un utilisateur visite votre site, vous pouvez utiliser des cookies pour stocker les préférences de celui-ci ou d'autres informations. Lorsque l'utilisateur revient sur votre site Web, l'application peut récupérer les informations stockées précédemment.

Les applications Web peuvent stocker des données de faible taille dans le navigateur client en utilisant les *Cookies*. Un *Cookie* est une petite quantité de données stockée soit dans un fichier texte dans le système de fichier client (si le *Cookie* est persistant) soit en mémoire dans les sessions du navigateur client (si le *Cookie* est temporaire). Le plus souvent, on utilise les *Cookies* pour identifier un utilisateur sur les différentes pages d'un même site Web. Ils permettent aussi d'afficher les informations relatives à cet utilisateur si on le désire.

Les *Cookies* sont le moyen le plus flexible et le plus fiable pour stocker des informations sur le poste client. Cependant, l'utilisateur peut supprimer ces *Cookies* à n'importe quel moment s'il le désire. Pour pallier à ce genre de problème il faut tout le temps tester la validité des *Cookies* et proposer une nouvelle authentification si nécessaire qui recréera un nouveau *Cookie* contenant les informations dont le site a besoin.

Les cookies sont stockés sur l'ordinateur client, et lorsque le navigateur demande une page, il envoie les informations contenues dans le cookie. Le serveur est autorisé à lire le cookie et à en extraire la valeur. Chaque cookie contient des informations sur le domaine dont il est issu. Un même domaine peut émettre plusieurs cookies.

Il existe deux types de cookies :

- Temporaires

Les cookies temporaires, appelés également cookies de la session ou cookies non persistants, sont stockés uniquement dans la mémoire du navigateur. Lorsque le navigateur est arrêté, ils sont supprimés.

- Persistants

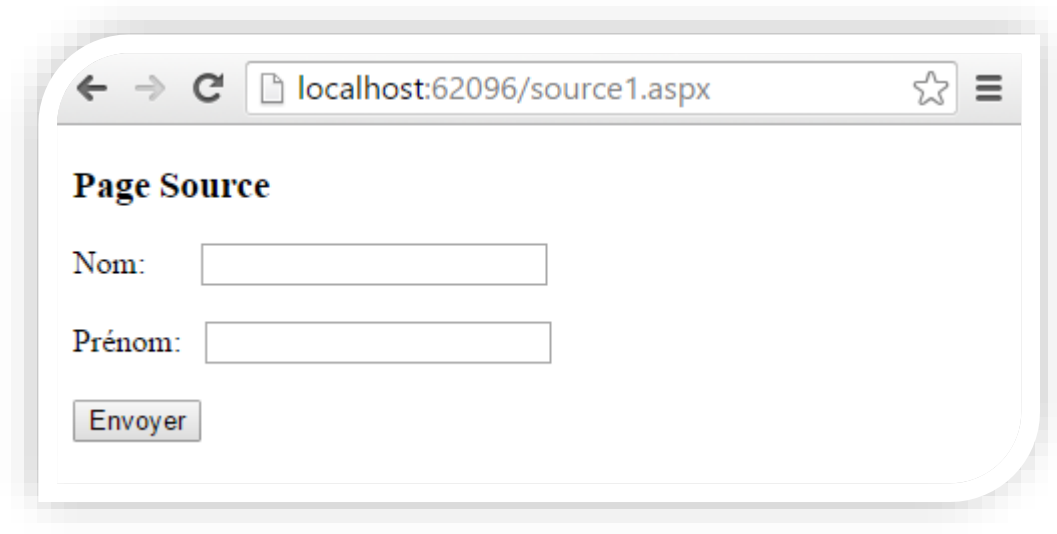
Les cookies persistants sont identiques aux cookies temporaires, mais ils possèdent une période d'expiration définie. Lorsqu'un navigateur demande une page qui crée un cookie persistant, il enregistre celui-ci sur le disque dur de l'utilisateur. Les cookies persistants peuvent résider pendant plusieurs mois, voire plusieurs années, sur l'ordinateur client.

Les cookies peuvent stocker une et ils sont moins sécurisés que les options de la gestion d'état côté serveur. Ils peuvent être en outre falsifiés. Les utilisateurs peuvent manipuler les cookies qui se trouvent sur leur ordinateur, et compromettre ainsi la sécurité ou entraîner l'échec de l'application qui dépend du cookie.

Exemple :

Dans cet atelier nous allons transférer les informations d'une page en utilisant une variable de cookie.

1. Créer une page source1.aspx contenant le même formulaire suivant :



2. Ajoutez le code suivant dans l'événement click du bouton *Envoyer* :

```
protected void Button1_Click(..){
    Response.Cookies["nom"].Value = txtNom.Text ;
    Response.Cookies["prenom"].Value = txtPrenom.Text ;
    DateTime now = DateTime.Now;
    Response.Cookies["nom"].Expires = now.AddHours(1);
    Response.Cookies["prenom"].Expires = now.AddHours(1);
    Response.Redirect("cible2.aspx")
}
```

3. Créer une deuxième page cible1.aspx content un Label
4. Ajouter le code suivant pour récupérer le nom et le prénom dans l'événement Load de la page :

```
protected void Page_Load(..){
    Label1.Text = "Bonjour " + Request.Cookies["nom"].Value + " " +
    Request.Cookies["prenom"].Value;
}
```

5. Exécuter la page source1.aspx et cliquer sur le bouton
6. Taper directement l'url de la page cible1.aspx dans le navigateur
7. Qu'est-ce que vous remarquez après le chargement de la page ?

5. Les Query Strings :

L'envoi de données à une page dynamique se fait par l'intermédiaire d'un formulaire HTML.

Chaque élément du formulaire doit posséder un nom unique, de telle façon que la valeur associée à l'élément forme avec le nom de celui-ci une paire nom/valeur du type :

Nom_de_l_element=valeur

L'ensemble des paires nom/valeur sont séparées par des esperluettes (le caractère « & »). Ainsi, l'envoi d'un formulaire crée une chaîne de la forme :

champ1=valeur1&champ2=valeur2&champ3=valeur3

L'envoi de cette chaîne se fera différemment selon que la méthode utilisée pour l'envoi du formulaire est GET ou POST.

- **L'objet Request**

Pour manipuler l'ensemble des informations envoyées par un formulaire, on va utiliser l'objet *Request*.

Le rôle de l'objet *Request* est de permettre de récupérer la requête HTTP envoyée par le client au serveur.

Cet objet possède deux collections pour le traitement des formulaires :

- La collection *QueryString* permet de récupérer la chaîne de requête envoyée par la méthode GET, c'est-à-dire les paires clés/valeurs présentes dans l'URL après le point d'interrogation. Prenons par exemple l'URL suivante :

`http://localhost/tp2/test.aspx?id=12&nom=Ahmed`

La chaîne de requête est ainsi la suivante :

`id=12&nom=Ahmed`

id et *nom* sont ici des champs (parfois appelés clés) et leurs valeurs respectives sont *12* et *Ahmed*

Voici la syntaxe de la collection *QueryString* :

`Request.QueryString["Champ"]`

Le paramètre *champ* désigne la variable que l'on désire récupérer. Il s'agit d'un paramètre obligatoire.

Ainsi, la récupération de la variable *nom* dans l'exemple ci-dessus se fera grâce à l'instruction suivante :

`Request.QueryString["nom"]`

- La collection *Form* permet de manipuler les données envoyées par un formulaire en utilisant

la méthode POST.

La collection *Form* permet de récupérer la valeur associée à un champ par la syntaxe suivante :

```
Request.Form["Champ"]
```

- **L'objet Response**

Le rôle de l'objet *Response* est de permettre de créer la réponse HTTP qui va être envoyée au navigateur, c'est-à-dire la page Web demandée par le client.

Pour envoyer du texte au navigateur dans un code ASP, il suffit d'utiliser la méthode *write* de l'objet *Response*, voici un exemple simple montrant comment utiliser cette méthode :

```
Response.write("Votre nom : " & nom)
```

Pour faire une redirection vers une page, on utilise la méthode *Redirect*

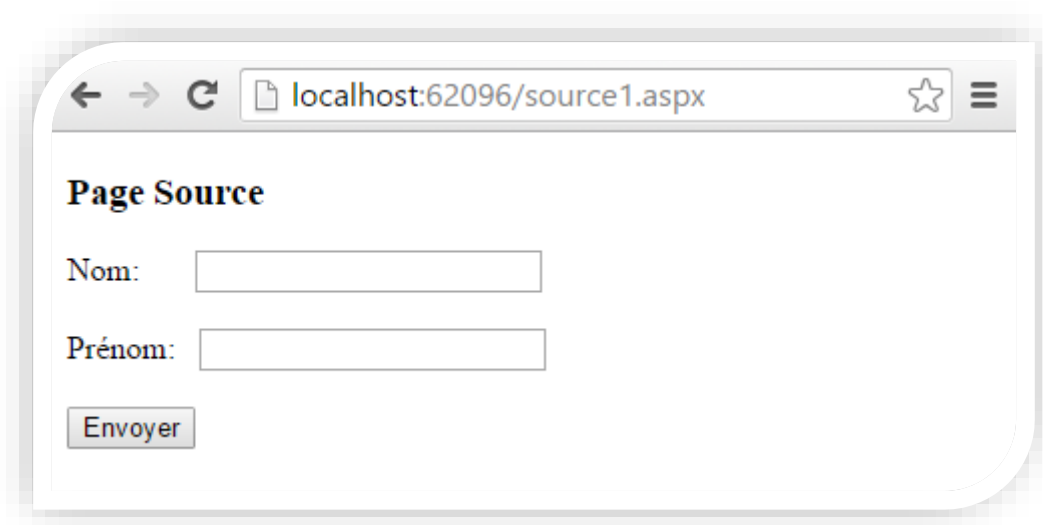
Exemple :

```
Response.Redirect("AutrePage.aspx")
```

Les objets *Request* et *Response* possèdent aussi la collection **Cookies** pour manipuler les cookies d'une application.

Exemple :

1. Créer une page *source1.aspx* contenant le formulaire suivant :



The screenshot shows a web browser window with the address bar displaying 'localhost:62096/source1.aspx'. The page content is titled 'Page Source' and contains a form with two text input fields labeled 'Nom:' and 'Prénom:'. Below these fields is a button labeled 'Envoyer'.

2. Ajoutez le code suivant dans l'événement click du bouton *Envoyer* :


```
protected void Button1_Click(..){

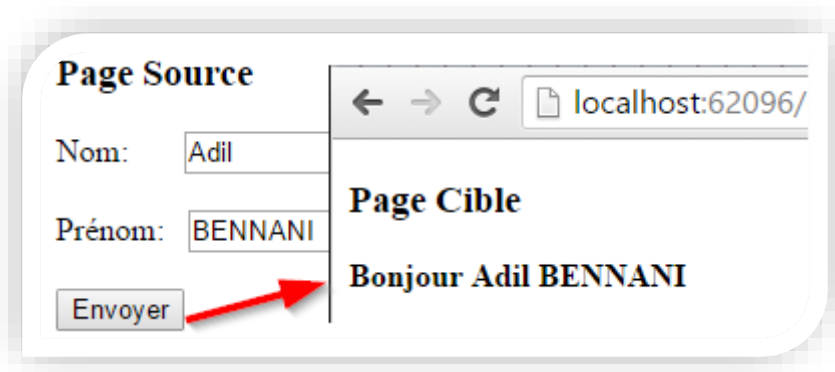
    Response.Redirect("cible1.aspx?nom=" + txtNom.Text + "&prenom=" +
txtPrenom.Text);
}
```

3. Créer une deuxième page cible1.aspx content un Label
4. Ajouter le code suivant pour récupérer le nom et le prénom dans l'événement Load de la page :

```
protected void Page_Load(..){

Label1.Text = "Bonjour " + Request.QueryString["nom"] + " " +
Request.QueryString["prenom"];
}
```

5. Exécuter la page source1.aspx et cliquer sur le bouton



6. Modifier le code la page source1.aspx par :

```
protected void Button1_Click(..){

Response.Redirect("cible1.aspx");
}
```

7. Modifier le code de l'événement Load de la page cible1.aspx par :

```
protected void Page_Load(..){

Label1.Text = "Bonjour " + Request.Form["txtNom"] + " " + Request.Form["txtPrenom"];
}
```

8. Exécuter la page `source1.aspx` et remarquer que les données saisies ne sont pas affichées dans la page `cible1.aspx` même si le formulaire est envoyé par la méthode POST.
9. Modifier le code la page `source1.aspx` par :

```
protected void Button1_Click(..){  
  
    Server.Transfer("cible1.aspx");  
}
```

10. Qu'est-ce que vous remarquez ?
11. Taper directement l'url de la page `cible1.aspx` dans le navigateur
12. Qu'est-ce que vous remarquez après le chargement de la page?

Exercices :

Exercice 1 : Formulaire d'authentification avec redirection

L'objectif de cet exercice est de protéger une page (qui contient des informations confidentielles) par un mot de passe.

a- Créer un formulaire (*login.aspx*) qui demande à l'utilisateur la saisie de son pseudo et son mot de passe et qui appelle la page protégée (*secret.aspx*).

L'accès au contenu de la page (*secret.aspx*) ne sera autorisé que si le pseudo et le mot de passe sont corrects.

Si l'utilisateur demande la page *secret.aspx* directement, il devra être redirigé vers la page d'authentification en lui affichant un message d'erreur. Utilisez les méthodes `response.redirect(url_page)` et `server.transfer(url_page)` qui permettent de rediriger l'utilisateur vers la page spécifiée en paramètre.

Vous devez donc créer 2 pages web :

- **login.aspx** : contient un simple formulaire d'authentification.
- **secret.aspx** : contient les informations confidentielles mais ne les affiche que si on lui donne le mot de passe.

Exercice 2 : Formulaire d'authentification avec cookies

Améliorer l'exercice précédent en ajoutant une case à coché « Se souvenir de moi » au niveau du formulaire d'authentification afin de permettre à un utilisateur déjà authentifié de demander la page *secret.aspx* directement.