



Concepteur Développeur en Informatique

Développer des composants d'interface

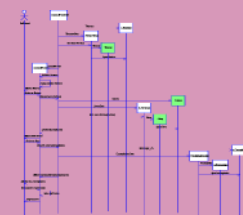
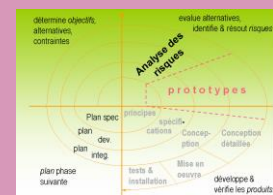
Présentation ASP Net – Objets intégrés

Accueil

Apprentissage

PAE

Evaluation



Localisation : U03-E05-S03

SOMMAIRE

| | |
|--|-----------|
| 1. Introduction..... | 3 |
| 2. Les principes fondamentaux d'une application Asp Net | 3 |
| 2.1. Les objets intrinsèques d'ASP Net | 3 |
| 2.1.1. Contexte de page et d'application | 4 |
| 2.1.2. Etat de l'application ApplicationState | 4 |
| Considérations sur l'état de l'application..... | 5 |
| Exemple d'utilisation de l'objet état de l'application | 6 |
| 2.1.3. Etat de Session | 7 |
| Considérations sur l'état de session | 7 |
| Exemple d'utilisation de l'objet état de session | 8 |
| Configurer une session..... | 10 |
| 2.1.4. Compléments sur la gestion des états de Session et d'Application | 13 |
| Exemple d'utilisation | 13 |
| 2.1.5. L'objet Response | 14 |
| Exemple de l'écriture d'un cookie | 14 |
| 2.1.1. Objet request | 18 |
| 1 ^{er} exemple pour récupérer les valeurs passées par les contrôles de formulaire | 18 |
| Déterminer les capacités du navigateur..... | 20 |
| Résultat | 20 |
| Obtenir la liste des variables du serveur | 20 |
| 2.1.2. Objet Trace | 21 |
| 3. Ce qu'il faut savoir à minima d'une page ASP.Net..... | 22 |
| 3.1. Le script serveur..... | 22 |
| </asp:textbox> | 22 |
| 3.1.1. Les directives de page | 23 |

1. Introduction

Ce document a pour objectif de vous présenter les bases du développement avec la technologie Asp.Net.

Les contrôles ASP.Net et le modèle de rendu de page seront abordés ultérieurement.

2. Les principes fondamentaux d'une application Asp Net

2.1. Les objets intrinsèques d'ASP Net

Si vous avez compris le modèle client/serveur Web, vous avez pu constater que les données ne sont accessibles que durant le temps de traitement de la requête. Chaque page est traitée comme une nouvelle demande et les informations de celle-ci ne sont plus disponibles ensuite. C'est le principe du mode **sans état** avec absence de session persistante.

Ce principe est adapté à des transactions simples : Par exemple l'utilisateur communique via un formulaire HTML des informations qui sont postées (méthode POST de HTTP) au serveur qui va se charger de les stocker dans la base de données. Il suffit pour cela de récupérer les valeurs transmises par la requête sous forme de paires clés/valeurs et de les insérer dans la base de données en invoquant les services d'ADO.

Mais les applications Web doivent pouvoir prendre en compte des transactions plus complexes. Prenons comme exemple une prise de commande en ligne : celle-ci ne peut être assurée par un échange unique entre le client et le serveur au travers d'une requête http. Elle correspond en fait à un ensemble de transactions effectuées, et un ensemble d'aller-retour entre le client et le serveur, au cours d'une même session, une session étant démarré lors de la première transaction (création du panier) et terminée sur la dernière transaction (validation du panier et paiement).

Les développeurs de serveurs Web ont donc implémenté des mécanismes permettant de conserver l'état de nos objets au-delà du temps de traitement d'une requête http.

Il existe deux objets de portée différente :

- L'état de session pour un utilisateur
- L'état de l'application pour conserver l'état de nos objets tout au long de la vie de l'application (entre démarrage et arrêt du serveur Web hébergeant l'application)

Lors de l'exécution d'une application Web, ASP.NET conserve des informations sur l'application en cours, sur chaque session utilisateur, sur la demande HTTP en cours, sur la page demandée, etc.

ASP.NET contient une série de classes permettant d'encapsuler ces informations de contexte.

Ces classes vous sont présentées dans le tableau page suivante.

Il convient de bien comprendre les conditions d'utilisation des objets issus de ces classes en fonction du contexte (scope) auquel on souhaite accéder.

2.1.1. Contextes de page, de session et d'application

| Nom de l'objet | Description | Classe ASP.NET |
|--------------------|---|-----------------------------|
| Response | Permet d'accéder au flux de sortie pour la page active. Cette classe peut être utilisée pour insérer du texte dans la page ou écrire dans des cookies. | HttpResponse |
| Request | Permet d'accéder à la demande de page active. Cette classe peut être utilisée pour lire les données envoyées par le navigateur : entêtes, cookies, certificat client, chaîne de la requête, données transmises. | HttpRequest |
| Context | Permet d'accéder à l'ensemble du contexte en cours (y compris l'objet de demande). Cette classe peut être utilisée pour partager des informations entre les pages. | HttpContext |
| Server | Présente des méthodes utilitaires qui permettent de transférer le contrôle entre les pages, d'obtenir des informations sur l'erreur la plus récente, d'encoder et de décoder du texte HTML, etc. Utile pour aiguiller les clients vers une page précise en fonction d'un contexte particulier. | HttpServerUtility |
| Application | Permet d'accéder à des méthodes et des événements d'application. Elle permet aussi de stocker des valeurs dans un cache d'application afin de les partager tout au long de la durée de vie de l'application. On parle ici d'état d'application ApplicationState . | HttpApplicationState |
| Session | Fournit des informations sur la session utilisateur en cours. Permet également d'accéder à un cache de session, que vous pouvez utiliser pour stocker des informations, et fournit les moyens de contrôler la gestion de la session. On parle ici d'état de session SessionState . | HttpSessionState |
| Trace | Fournit un moyen d'afficher les messages de diagnostic du suivi, créés par le système ou par vous-même, dans la sortie de page HTTP. | TraceContext |

2.1.2. Etat de l'application **ApplicationState**

L'état de l'application est un référentiel de données disponible pour toutes les classes d'une application ASP.NET.

Il est stocké en mémoire sur le serveur et plus rapidement accessible qu'une information stockée en base de données.

Contrairement à l'état de session, spécifique à une session mono-utilisateur, l'état de l'application s'applique à tous les utilisateurs et à toutes les sessions.

C'est donc un emplacement utile pour stocker de **petites** quantités de données souvent utilisées et qui restent identiques d'un utilisateur à un autre.

Considérations sur l'état de l'application

Lorsque vous utilisez l'état de l'application, vous devez tenir compte des considérations suivantes :

- **Ressources**

Parce qu'il est stocké en mémoire, l'état de l'application est très rapide comparé à l'enregistrement de données sur disque ou dans une base de données. Toutefois, stocker de grands blocs de données dans l'état de l'application peut remplir la mémoire serveur et obliger le serveur à paginer la mémoire vers le disque.

Plutôt que d'utiliser l'état de l'application, utilisez le cache d'ASP Net pour stocker d'importants volumes de données. Nous verrons comment utiliser l'objet **Cache** d'ASP dans ce document.

- **Volatilité**

L'état de l'application étant stocké dans la mémoire serveur, il est perdu chaque fois que l'application est arrêtée ou redémarrée.

- **Évolutivité**

L'état de l'application ne peut être partagé entre plusieurs serveurs servant la même application, comme dans une batterie (ferme) de serveurs Web, ou entre plusieurs processus de travail servant la même application sur le même serveur.

- **Accès concurrentiel**

L'état de l'application est libre de threads, ce qui signifie que ses données sont accessibles simultanément par un grand nombre de threads.

Si cela n'était guère prégnant dans le cas d'une application client lourd windows, il n'en est pas de même dans le contexte d'une application Web.

Il faut donc mettre à jour les données d'état de l'application de manière **thread-safe** en incluant une prise en charge intégrée de la synchronisation.

Utiliser les méthodes Lock et UnLock pour garantir l'intégrité des données en les verrouillant pour qu'elles ne puissent être écrites que par une seule source à la fois.

- **Événements**

L'état de l'application dispose de deux événements, start et end, qui vous permettront de gérer les phases d'initialisation et de terminaison de l'application. Nous verrons comment mettre en place des gestionnaires d'événements associés dans le fichier **Global.asax** qui sera abordé ultérieurement.

Exemple d'utilisation de l'objet état de l'application

Cet exemple présente le stockage d'une valeur dans une variable d'application.

Ces valeurs sont stockées sous la forme d'une paire de clés de type chaîne /valeurs de type objet. Il s'agit d'une collection de type **NameObjectCollectionBase**. Il est possible d'accéder aux éléments par l'index ou par la clé.

Vous pouvez obtenir l'ensemble des clés via `Keys` ou l'ensemble des valeurs via `staticObjects`.

Comme le figure l'exemple ci-dessous, il vous faudra convertir l'objet dans le type attendu.

Ici, nous souhaitons afficher dans un label le nombre de pages chargées depuis le démarrage de l'application.

Vous remarquerez l'usage de la méthode **Lock** sur l'objet `Application` associée à la page pour verrouiller l'accès à la valeur durant cette transaction.

```
protected void Page_Load(object sender, EventArgs e)
{
    Application.Lock();
    Application["CompteurPages"] =
        (Application["CompteurPages"] != null)
        ? ((int)Application["CompteurPages"])+1
        : 1;
    Application.Unlock();
    lblNombrePages.Text = |
        string.Format("Nombre de pages chargées depuis le démarrage {0}",
            Application["CompteurPages"]);
}
```

Résultat initial :

Nombre de pages chargées depuis le démarrage 4

Résultat après rafraichissement de la page :

Nombre de pages chargées depuis le démarrage 5

2.1.3. Etat de Session

L'état de session vous permettra de conserver des valeurs dans des variables dites de session pour chaque utilisateur de votre site Web et cela durant la durée de vie de leur session.

Considérations sur l'état de session

Lorsque vous utilisez l'état de session, vous devez tenir compte des considérations suivantes :

- **Conservation des valeurs** L'état de session ASP.NET vous permet de stocker et de récupérer des valeurs pour un utilisateur à mesure que ce dernier navigue dans les différentes pages ASP.NET d'une application Web. Le fait que HTTP soit un protocole sans état implique qu'un serveur Web traite chaque requête HTTP pour une page comme une demande indépendante.
Le serveur ne conserve donc aucune connaissance des valeurs variables utilisées dans les précédentes requêtes.
L'état de session ASP.NET identifie les demandes du même navigateur en tant que session dans une fenêtre à durée limitée et offre un moyen de rendre persistantes les valeurs variables pour la durée de cette session.

- **Variables de session** Les variables de session sont stockées dans un objet `SessionStateItemCollection` qui s'affiche via la propriété `HttpContext.Session`. Dans une page ASP.NET, les variables de session actuelles sont exposées à travers la propriété `Session` de l'objet `Page`.

Tout comme pour les variables d'application, la collection de variables de session est indexée par le nom de la variable et par un index.

Les variables de session sont créées par référence à leur nom. Vous n'avez pas besoin de déclarer de variable de session ni d'en ajouter explicitement à la collection.

- **Événements de session** Deux événements sont disponibles pour vous aider à gérer les sessions utilisateur.
L'événement **Session_OnStart** est déclenché lors du démarrage d'une nouvelle session, et l'événement **Session_OnEnd** est déclenché lorsqu'une session est abandonnée ou expire en fonction de la durée indiquée au niveau du serveur.
Les événements de session sont spécifiés dans le fichier `Global.asax` d'une application ASP.NET.
L'événement `Session_OnEnd` n'est pas pris en charge si la propriété de **session Mode** a une valeur différente de `InProc` (dans le même processeur que l'application), qui est le mode par défaut. Les variables de session peuvent être stockées sur un autre serveur ou dans une base de données.

Exemple d'utilisation de l'objet état de session

Soit l'exemple suivant qui permet de stocker, dans un objet session, les propriétés prénom et le nom d'un utilisateur transmises par le biais d'une page HTML à un composant serveur.

Enregistrement Session

Entrez votre prénom :

Vincent

Entrez votre nom :

Bost

Enregistrer dans la session

Les données sont saisies dans le formulaire HTMML puis postées au composant EcrireSession.aspx. qui prend en charge l'enregistrement des informations transmises :

```

0 références
protected void Page_Load(object sender, EventArgs e)
{
    this.Session["Prenom"] = this.Request.Params["txtPrenom"];
    this.Session["Nom"] = this.Request.Params["txtNom"];
    this.Server.Transfer("AfficherSession.aspx");
}
  
```

Nous stockons sous la clé Prenom la valeur communiquée via le formulaire HTML. Nous faisons de même pour le Nom.

Nous transférons ensuite le contrôle à un troisième composant qui aura pour objectif de restituer les éléments stockés dans l'objet session.

Code du composant d'affichage :

```

0 références
protected void Page_Load(object sender, EventArgs e)
{
    lIDSession.Text = this.Session.SessionID;
    lPrenom.Text = this.Session["Prenom"].ToString();
    lNom.Text = this.Session["Nom"].ToString();
}
  
```


Résultat :

Affichage des infos session

ID Session :

Prénom :

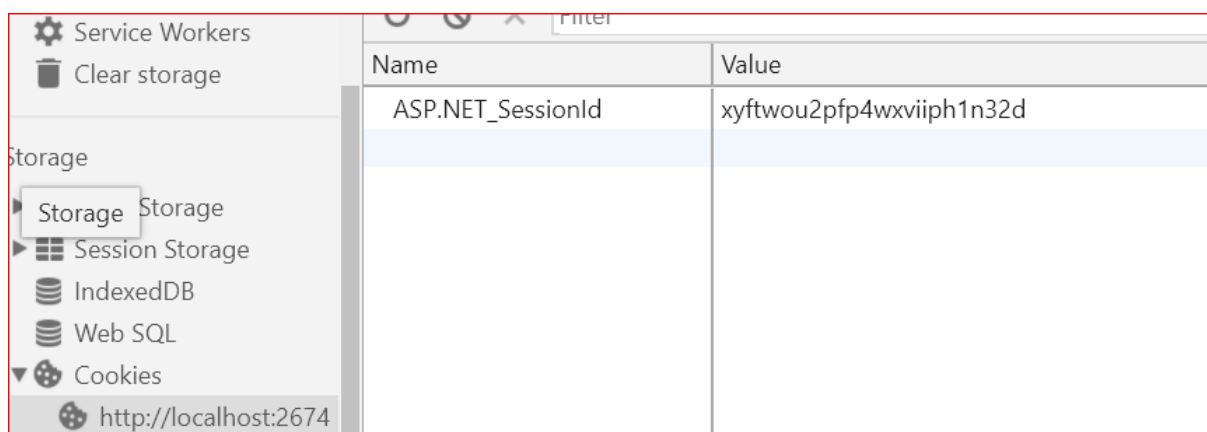
Nom :

Nous avons ainsi apporté la preuve que nous pouvions conserver les valeurs transmises au-delà de la durée de vie de la requête initiale.

Pour rappel les transactions effectuées :



Par défaut, la session de l'utilisateur est gérée grâce à la présence d'un **cookie de session**. Vous pouvez le vérifier en consultant les cookies présents sur le client :



Les cookies sont échangés dans toutes les transactions Requête/Réponse entre le client et le serveur. Nous pouvons aussi bien extraire les valeurs des cookies ou écrire dans ces derniers à partir du client comme du serveur

Cette première version pourrait être améliorée. Nous pourrions stocker les infos dans un objet Utilisateur doté de 2 propriétés Nom et Prénom. L'objet **session** **préserve les types**.

Configurer une session

Chaque session ASP.NET est identifiée à l'aide d'une chaîne de 120 bits (15*8) SessionID, unique générée par un composant système.

Elle est communiquée au navigateur puis renvoyée à l'application serveur par l'intermédiaire soit, d'un cookie, soit d'un URL modifié : l'approche choisie dépend des paramètres de configuration stockés dans le fichier Web.config, l'état de session exploite par défaut les cookies de session.

Session sans cookies

Pour que l'état de session soit exploitable, le client doit être en mesure de transmettre l'identification de session à l'application serveur ; Les applications ASP.Net définissent les paramètres spécifiques à la session dans la section **<SessionState>** du fichier de configuration.

La prise en charge des cookies dépend de la configuration de l'attribut cookieless. A noter : même si un navigateur prend en charge les cookies, l'utilisateur peut les avoir désactivés ; dans ce cas l'état de session ne va pas fonctionner correctement. Lorsque le support des cookies est désactivé, l'identificateur de session s'insère dans le contenu de la barre d'adresse du navigateur

Les valeurs de la propriété **cookieless** :

| Valeur | Description |
|-------------------------|--|
| AutoDetect | Utilise les cookies uniquement si le navigateur les prend en charge. |
| UseCookies | Les cookies sont utilisés pour maintenir l'identificateur de session que les navigateurs les prennent en charge ou non (option par défaut) |
| UseDeviceProfile | Orienté la décision en fonction des possibilités des navigateurs décrites dans la section profile dans le fichier de configuration |
| UseUri | Utilise systématiquement la chaîne de requête pour stocker un identificateur sans prendre en compte le fait de savoir si le navigateur ou le périphérique prend en charge les cookies. |

Les sessions sans cookies présentent des inconvénients :

- Elles déclenchent une redirection au démarrage de la session, et à chaque fois que l'utilisateur utilise une URL absolue
- Les données de session sont perdues lors de l'utilisation de liens vers des URL absolues
- L'identificateur de session étant visible dans la barre d'adresse, il n'est pas recommandé de stocker des informations essentielles sans utiliser SSL (Secure Sockets Layer) pour crypter les communications.

Les développeurs travaillent avec un seul objet, l'objet Session, grâce au module HTTP **SessionStateModule**, qui supervise le processus de récupération et d'enregistrement de l'état de session.

Lorsqu'il est invoqué, le module HTTP lit la valeur de l'attribut mode dans la section **<SessionState>** du fichier de configuration pour connaître le mode de gestion de la collection paire/valeurs.

Valeurs de l'attribut mode.

| Valeur | Description |
|--------------------|---|
| Custom | L'état de session est stocké dans une base de données personnalisée. (ASP.NET 2.0) |
| InProc | L'état de session est stocké dans la mémoire du processus de travail ASP.NET. (valeur par défaut) |
| Off | L'état de session est désactivé. |
| SQLServer | L'état de session utilise une base de données SQL Server out-of-process pour stocker les informations d'état. |
| StateServer | Les valeurs de toutes les sessions sont sérialisées et stockées dans la mémoire d'un processus système séparé, qui peut aussi être exécuté sur une autre machine. |

L'option InProc offre les meilleurs résultats en terme de rapidité d'accès : toutefois, il est important de ne pas utiliser trop de stockage mémoire ce qui pourrait diminuer les performances du serveur.

Si une solution hors processus est envisagée, il est nécessaire de sérialiser les valeurs des variables conservées au niveau de la session. Il convient de vérifier l'impact en matière de performances des mécanismes de sérialisation.

A cette occasion, nous pouvons découvrir le fichier de configuration ASP.NET. Tout comme pour les applications Windows, les propriétés de configuration sont conservées dans un fichier de configuration XML.

Le .NET Framework définit un ensemble d'éléments qui implémentent des paramètres de configuration et le schéma de configuration ASP.NET contient des éléments qui contrôlent le comportement des applications Web ASP.NET.

Les paramètres de configuration par défaut sont spécifiés dans les fichiers Machine.config et Web.config situés dans le répertoire

%SystemRoot%\Microsoft.NET\Framework\NuméroVersion\CONFIG\.

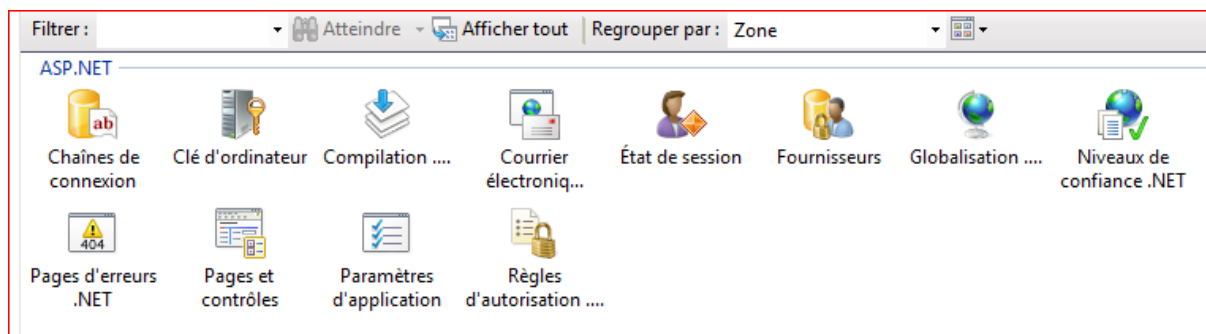
Les valeurs sont héritées par les sites et les applications enfants.

S'il existe un fichier de configuration dans une application ou un site enfant, les valeurs héritées n'apparaissent pas, mais elles peuvent être substituées et sont accessibles à l'API de configuration.

Du fait de la complexité des sections de configuration, il est préférable de les modifier en recourant aux outils spécifiques mis à votre disposition plutôt que d'envisager de modifier directement le fichier de configuration.

ASP Net Objets intégrés

Modification des paramètres par le biais de l'interface au niveau de IIS 7.



État de session

Paramètres de mode d'état de session

☐ Non activé

☒ Dans le processus

☐ Personnalisé

☐ Serveur d'état

Chaîne de connexion :

tcpip=loopback:42424 Créer...

Délai d'attente (en secondes) :

10

☐ SQL Server

Chaîne de connexion :

data source=localhost;Integrated Security=SSPI Créer...

Délai d'attente (en secondes) :

30

☐ Activer la base de données personnalisée

Paramètres de cookies

Mode :

Utiliser les cookies

Nom :

ASP.NET_SessionId

Délai d'attente (en minutes) :

20

☐ Régénérer l'ID de session ayant expiré

2.1.4. Compléments sur la gestion des états de Session et d'Application

Lorsque vous créez un projet de type Application Web un fichier Global.asax est ajouté au projet. Ce fichier est également connu sous le nom de fichier d'application ASP.NET.

Il est facultatif et contient le code pour répondre aux événements de niveau application ou session déclenchés par ASP.NET ou par les modules HTTP.

Il réside toujours dans le répertoire racine d'une application ASP.NET.

Au moment de l'exécution, Global.asax est analysé et compilé en une classe .NET Framework générée de manière dynamique, dérivée de la classe de base `HttpApplication`.

ASP.NET est configuré de sorte que toute demande d'URL directe concernant le fichier Global.asax soit automatiquement rejetée. Ce fichier ne peut en aucun cas être consulté par des utilisateurs externes.

Ce fichier comporte dès sa finition 4 gestionnaires d'événement associés aux événements :

- Ouverture de session utilisateur `SessionOnStart`
- Fermeture de session `SessionOnEnd`
- Démarrage de l'application Web `ApplicationOnStart`
- Terminaison de l'application Web `ApplicationOnEnd`

Exemple d'utilisation

Dans cet exemple nous allons compter le nombre de sessions ouvertes durant le cycle de vie de l'application. Nous programmons donc ces opérations sur l'événement `onStart` de l'objet `Session` et incrémentons une variable d'état d'application. Nous implémentons ces opérations dans le gestionnaire ad hoc du fichier `global.asax`.

```
void Session_Start(object sender, EventArgs e)
{
    // Code qui s'exécute lorsqu'une nouvelle session démarre
    Application.Lock();
    Application["CompteurSessions"] =
        (Application["CompteurSessions"] != null)
        ? ((int)Application["CompteurSessions"]) + 1
        : 1;
    Application.Unlock();
}
```

2.1.5. L'objet Response

D'une manière synthétique, cet objet permet d'écrire dans le flux destiné au client en réponse à une demande (requête) et contrôler les informations transmises au client.

Vous pouvez donc par son intermédiaire écrire dans le flux (Stream) et gérer la mise en tampon (buffer) du flux. Son utilisation doit être limitée à des opérations bien spécifiques.

L'objet Response permet aussi, comme illustré précédemment, de rediriger le client vers une autre page par l'invocation de sa méthode Redirect.

L'objet Response nous permettra aussi d'écrire dans un espace mémoire disponible tant côté client que serveur, le cookie.

Les cookies sont accessibles via la propriété Cookies qui est une collection de paire de clés/valeurs de type chaîne.

L'intérêt d'un cookie réside dans le fait que les valeurs déposées par un site dans celui-ci sont transmises à chaque requête et fournies dans chaque réponse.

Je vous propose de nous arrêter quelques instants sur ces mécanismes.

Exemple de l'écriture d'un cookie

Si le recours à la technique des cookies est très fréquent, cette technique n'en est pas moins limitée du fait du contrôle qu'exerce l'utilisateur de l'application sur ces derniers.

Les cookies sont souvent utilisés à des fins de traçabilité et d'analyse des actions réalisées par l'utilisateur.

Par exemple, lorsqu'un utilisateur demande une page du site, l'application transmettra, en plus de la page, un cookie contenant la date et l'heure. Le navigateur de l'utilisateur récupérera cette page plus le cookie, qu'il stockera dans un dossier du disque dur local.

Ultérieurement, si l'utilisateur demande à nouveau une page de votre site, le navigateur recherchera sur le disque dur local un cookie associé à l'URL saisie. Si le cookie existe, le navigateur l'enverra à votre site en même temps que la demande de page.

L'application pourra alors déterminer la date et l'heure auxquelles l'utilisateur a visité le site pour la dernière fois. Vous pourrez éventuellement utiliser cette information pour afficher un message sur l'écran de l'utilisateur ou vérifier une date d'expiration.

Les cookies sont associés à un site Web, et non à une page spécifique, afin que le navigateur et le serveur échangent des informations de cookie quelle que soit la page demandée par l'utilisateur sur votre site.

Votre application ne peut donc accéder qu'aux cookies dont elle est dépositaire. Il est possible de restreindre la vue d'un cookie à un dossier de l'application.

La plupart des navigateurs prennent en charge des cookies jusqu'à 4 096 octets. Cette limite relativement basse fait que les cookies sont indiqués pour stocker de petites quantités de données.

Les navigateurs limitent également le nombre de cookies que votre site peut stocker sur l'ordinateur de l'utilisateur (une vingtaine).

ASP Net Objets intégrés

Une autre limitation aux cookies est la possibilité pour les utilisateurs de paramétrer leur navigateur de manière à ce qu'il refuse ceux-ci. Il faudra donc vous assurer que l'utilisateur les accepte. En tous cas, vous ne pouvez pas envisager d'opérations critiques associées à des cookies.

A noter : Les utilisateurs sont presque obligés d'accepter les cookies de session, cookies non persistants, dont la durée est celle de la session. Pourquoi ? Et bien le mécanisme de gestion d'état de session fonctionne de base sur ce principe : l'id session donné par le serveur est stocké dans un cookie temporaire et c'est la valeur de l'ID stocké dans ce cookie qui permet de reconnaître l'utilisateur au cours de la session et donc de récupérer les variables qui lui sont associées.

A noter : La propriété Cookies de l'objet Request.browser n'indique pas si les cookies sont activés. Elle indique seulement si le navigateur actuel prend en charge les cookies de façon inhérente. Nous y reviendrons lors de l'étude de l'objet request.

1^{ère} partie sur le chargement de la page. Le cookie est créé s'il n'existe pas déjà

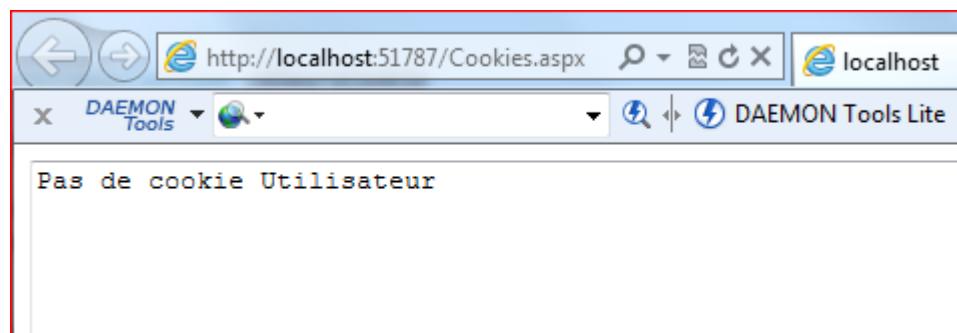
```
if (Request.Cookies["Utilisateur"] == null)
{
    HttpCookie cookie = new HttpCookie("Utilisateur");
    cookie.Values["Nom"] = "Bost";
    cookie.Values["Prenom"] = "Vincent";
    cookie.Values["DateDerniereVisite"] = DateTime.Now.ToString();
    cookie.Expires = DateTime.Now.AddDays(30);

    Response.Cookies.Add(cookie);
    txtCookie.Text = "Pas de cookie Utilisateur";
}
```

Il s'agit ici d'un cookie dont la valeur est une collection de paires clés/valeurs de type chaîne. 3 propriétés sont stockées : le nom, le prénom et la date de dernière visite de l'utilisateur.

La date d'expiration est fixée. Ce cookie sera résident sur le disque si le client accepte ce type de cookie...

1^{ère} fois



2^{ème} fois après rafraichissement de la page lorsque le cookie existe

```
else
{
    System.Text.StringBuilder chaineAffichee
        = new System.Text.StringBuilder();

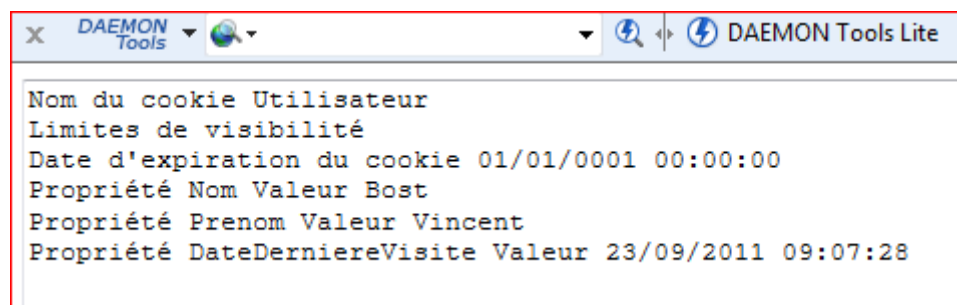
    HttpCookie cookie = Request.Cookies["Utilisateur"];

    chaineAffichee.Append(string.Format("Nom du cookie {0} \n",
        cookie.Name));
    chaineAffichee.Append(string.Format("Limites de visibilité {0} \n",
        cookie.Domain));
    chaineAffichee.Append(string.Format("Date d'expiration du cookie {0} \n",
        cookie.Expires));

    System.Collections.Specialized.NameValueCollection valeursCookie
        = cookie.Values;
    for (int i = 0; i < valeursCookie.Count; i++)
    {
        chaineAffichee.Append(string.Format("Propriété {0} Valeur {1} \n",
            valeursCookie.AllKeys[i], valeursCookie[i]));
    }

    txtCookie.Text = chaineAffichee.ToString();
}
```

Vous remarquerez le traitement de la collection des valeurs du cookie. Il s'agit d'une collection de type clés/valeurs chaîne.
Nous faisons appel ici à l'objet Request pour extraire (lire) les valeurs.



```
Nom du cookie Utilisateur
Limites de visibilité
Date d'expiration du cookie 01/01/0001 00:00:00
Propriété Nom Valeur Bost
Propriété Prenom Valeur Vincent
Propriété DateDerniereVisite Valeur 23/09/2011 09:07:28
```

2.1.1. Objet request

Cet objet permet de lire les valeurs transmises dans la requête http par le client. Cet objet était très usité dans les précédentes versions d'ASP avant la mise en place des mécanismes de gestion des états entre deux pages et la mise en œuvre de contrôles serveur. Nous verrons ces mécanismes dans la suite du cours.

Vous pourrez néanmoins toujours l'utiliser pour récupérer les valeurs d'un formulaire HTML mais aussi pour récupérer des informations transmises dans les entêtes http comme la famille et la version du navigateur.

1^{er} exemple pour récupérer les valeurs passées par les contrôles de formulaire

Sur ce formulaire, deux boîtes de texte txtNom et txtPrenom

Le script qui traitera les données transmises est indiqué au niveau de l'attribut action de l'élément form et la méthode http de soumission est get :

```
<body onload="affecterGestionnaires();">
  <form id='frmCoordonnees' action="Formulaire.aspx" method="get" enctype="multipart/form-data">
    <fieldset id='fsCoordonnees'>
      <legend>Merci de renseigner ces informations</legend>
      <p>
        <label for='txtNom'>
          Nom :</label>
        <input class="inputText" id="txtNom" name="txtNom" type="text" />
        <span id="M_Nom" class="Message" >* le nom doit être renseigné</span>
      </p>
      <p>
        <label for='txtPrenom'>
          Prénom :</label>
        <input class="inputText" id="txtPrenom" name="txtPrenom" type="text" />
        <span id="M_Prenom" class="Message">* le prénom doit être renseigné</span>
      </p>
      <p>
        <input id='btnEnvoyer' class="inputText" onclick="envoyer()" value="Envoyer" type="button" />
      </p>
    </fieldset>
  </form>
</body>
```

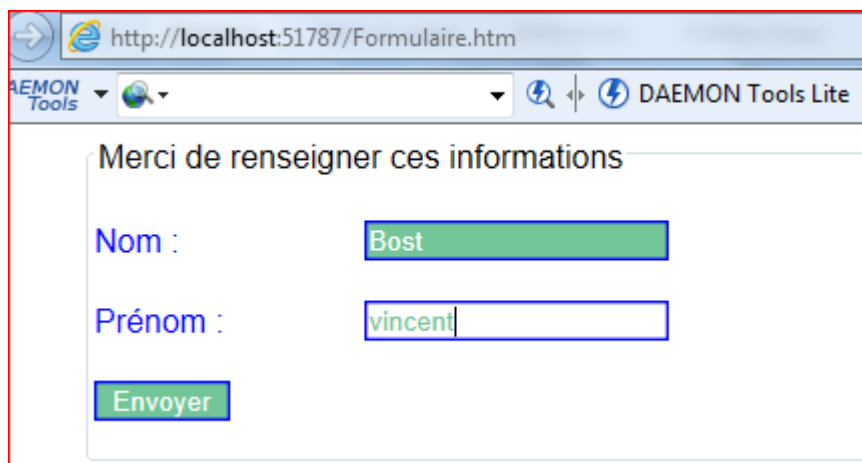
Au niveau du script de traitement des données.

En fonction de la méthode RequestType utilisée, les valeurs transmises sont récupérées dans la collection QueryString pour la méthode GET ou Form pour la méthode POST.

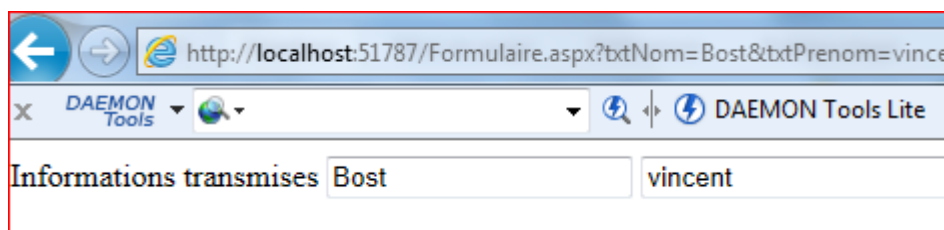
Lorsque vous utilisez la méthode GET, les données sont transmises dans l'URL.

```
protected void Page_Load(object sender, EventArgs e)
{
    switch (this.Request.RequestType)
    {
        case "GET":
            txtNom.Text = this.Request.QueryString["txtNom"];
            txtPrenom.Text = this.Request.QueryString["txtPrenom"];
            break;
        case "POST":
            txtNom.Text = this.Request.Form["txtNom"];
            txtPrenom.Text = this.Request.Form["txtPrenom"];
            break;
    }
}
```

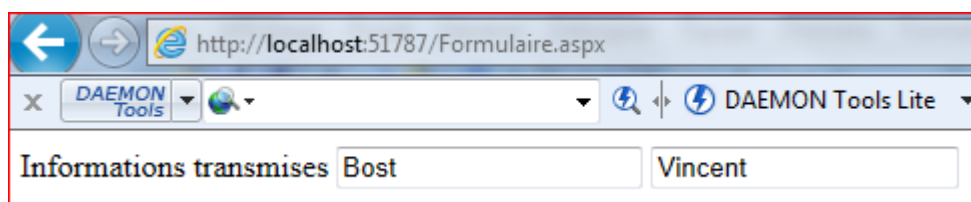
Résultat : page HTML



Résultat : page ASP.Net méthode GET



Résultat : page ASP.Net méthode POST



Les données transmises ne figurent plus dans l'URL. La méthode POST est préférable à la méthode GET plus limitée. On réservera la méthode GET au chargement de page via l'URL lorsque les liens sont générés dynamiquement.

Exemple l'article Offre CDI du catalogue AFPA des Formations

http://www.afpa.fr/formations/les-offres-de-formation-et-vae/formation-diplomante/fiche/9952/objectif/concepteur_developpeur_informatique.html

Pour des besoins d'indexation et de performance, les pages html sont générées et stockées sous cette forme au niveau du serveur.

Autres illustrations d'utilisation de l'objet Request

Déterminer les capacités du navigateur

Peut être utile à la préparation de la réponse.

Ces informations concernent le type de navigateur et non l'instance de navigateur de l'utilisateur. Si le type de navigateur peut interpréter le JavaScript cela ne signifie pas pour autant que l'utilisateur accepte que des scripts JavaScript soient interprétés par son navigateur...

Les capacités du navigateur sont extraites de l'objet Request.Browser

Capacités du Navigateur : Parmi les principales propriétés

```
private string extraireCapacitesNavigateur(HttpBrowserCapabilities navigateur)
{
    string capacites = string.Empty;
    capacites += string.Format("Type = {0} \n", navigateur.Type);
    capacites += string.Format("Nom = {0} \n", navigateur.Browser);
    capacites += string.Format("Version = {0} \n", navigateur.Version);
    capacites += string.Format("OS = {0} \n", navigateur.Platform);
    capacites += string.Format("Cookies supportés = {0} \n", navigateur.Cookies);
    capacites += string.Format("JavaScript supporté = {0} \n",
        navigateur.EcmaScriptVersion.ToString());
    capacites += string.Format("version DOM {0} \n",
        navigateur.W3CDomVersion.ToString());
    capacites += string.Format("Applets Java = {0} \n", navigateur.JavaApplets);
    capacites += string.Format("Activex = {0} \n", navigateur.ActiveXControls);
    return capacites;
}
```

Résultat

```
Type = IE9
Nom = IE
Version = 9.0
OS = WinNT
Cookies supportés = True
JavaScript supporté = 3.0
version DOM 1.0
Applets Java = True
Activex = True
```

Obtenir la liste des variables du serveur

Parmi ces informations, vous trouverez notamment :

- Le type du navigateur : HTTP_USER_AGENT
- La liste des cookies : HTTP_COOKIE
- Adresse logique du document : PATH_INFO
- Adresse physique de celui-ci : PATH_TRANSLATED
- Adresse IP du visiteur : REMOTE_ADDR
- Le nom du serveur : SERVER_NAME
- ...

Les variables du serveur de la collection de paires de clés/valeurs request.ServerVariables

```
private string extraireCaracteristiquesServeur(
    System.Collections.Specialized.NameValueCollection variablesServeur)
{
    string serveur = string.Empty;
    for (int i = 0; i < variablesServeur.Count; i++)
    {
        serveur += string.Format("Propriété : {0} \n", variablesServeur.Keys[i]);
        String[] proprietes = variablesServeur.GetValues(variablesServeur.Keys[i]);
        for (int j = 0; j < proprietes.Length; j++)
        {
            serveur += string.Format("Valeur {0} : {1} \n", j + 1, proprietes[j]);
        }
    }
    return serveur;
}
```

Un extrait du résultat obtenu

```
Propriété : INSTANCE_META_PATH
Valeur 1 :
Propriété : LOCAL_ADDR
Valeur 1 : 127.0.0.1
Propriété : PATH_INFO
Valeur 1 : /Informations.aspx
Propriété : PATH_TRANSLATED
Valeur 1 : E:\CDI\CDI-U03\CDI-U03-E05\CDI-U03-E05-S03\05 -
EC\SupportCours\SupportCours\Informations.aspx
Propriété : QUERY_STRING
Valeur 1 :
Propriété : REMOTE_ADDR
Valeur 1 : 127.0.0.1
```

2.1.2. Objet Trace

Il s'agit d'un objet utilisé en mode debug pour vous permettre de mettre au point vos pages. Un exemple de ce qui peut être affiché en mode trace.

Votre ID Session est : z0rxyzf1y35hyjsydogoozid

Votre Prénom est : vincent

Votre Nom est : bost

Détails de la demande

ID de session:

Heure de la demande:

Encodage de la demande:

z0rxyzf1y35hyjsydogoozid

26/09/2011 08:46:45

Unicode (UTF-8)

Ty

En

Informations de traçage

Catégorie

Message

À partir des premiers

aspx.page

Begin PreInit

aspx.page

End PreInit

4,02716043338563E-05

aspx.page

Begin Init

9,04136999260108E-05

aspx.page

End Init

0,000126342288106216

aspx.page

Begin InitComplete

0,000140160975867833

aspx.page

End InitComplete

0,000162665695936753

3. Ce qu'il faut savoir à minima d'une page ASP.Net

3.1. Le script serveur

Une page ASP.Net (suffixée « .aspx ») peut mélanger du code destiné à être exécuté sur le client (html et scripts client) à du code destiné à être exécuté sur le serveur.

Ce script serveur est distingué du reste de la page par des balises particulières et peut être placé dans un fichier séparé (suffixée « .aspx.cs » en C#). Nous verrons par la suite qu'il doit, de préférence, être codé dans la page C# associée.

| Variantes de script | Balises | Exemples |
|-------------------------------------|---|--|
| Script serveur en ligne | <% ... %> | <% Response.Write(time);%> |
| Affichage d'une expression en ligne | <%= ... %> | <%= time %> |
| Fonction script serveur | <script language="c#" runat=server> ... </script> | < script language="c#" runat=server > public string GetDate() { ... } </script> |
| Directive de paramétrage de la page | <%@ Page ... %> | <%@ Page Language="C#" AutoEventWireup="true" CodeFile="Default.aspx.cs" Inherits="_Default" %> |
| Uniquement en conception ... | | |
| Contrôle serveur HTML | Exemple de contrôle « HtmlInputText » | <INPUT type="text" id="Text1" name="Text1" runat="server"> |
| Contrôles serveur Web | Exemple de contrôle « TextBox » | <asp:TextBox id="TextBox1" runat="server"> </asp:TextBox> |
| Expression de liaison de données | <%# ... %> | <asp:textbox id=txtNom runat="server" Text=' <%# DataBinder.Eval(ds21, "Tables[TEntrepise].DefaultView.[0].En_No m")%> '> </asp:textbox> |

Exemple

```
<body>
  <% string enteteTableau;

  enteteTableau =
  "<table class='tbFormulaire'><thead><tr><td>Propriété</td><td>Valeur</td></tr></thead>";
  |
  HttpContext requeteCourante;
  HttpRequest requete;
  HttpResponse reponse;
  requeteCourante = HttpContext.Current;
  requete = requeteCourante.Request;
  reponse = requeteCourante.Response;
  reponse.Write(enteteTableau);
  %>
```

3.1.1. Les directives de page

```
<%@ Page Language="C#" AutoEventWireup="true"  
CodeBehind="FrmSession.aspx.cs" Inherits="SupportCours.FrmSession" %>
```

Elles sont introduites par une balise particulière @ Page.

Elles déterminent les caractéristiques spécifiques de la page utilisées par le compilateur et l'analyseur de requêtes.

Voici les quelques valeurs basiques qu'il est nécessaire de connaître

AutoEventWireup

Indique si les événements de la page sont auto-connectés. true si l'auto-connexion d'événement est activée ; sinon, false. La valeur par défaut est true.

CodeBehind

Spécifie le nom du fichier compilé qui contient la classe associée à la page. Cet attribut n'est pas utilisé au moment de l'exécution.

Cet attribut est utilisé pour les projets d'application Web. L'attribut CodeFile est utilisé pour les projets de site Web. Pour plus d'informations au sujet des types de projet Web dans Visual Studio, consultez Projets d'application Web et projets de site Web.

ClassName

Chaîne qui spécifie le nom de classe de la page compilée dynamiquement lors de la demande de la page.

Cette valeur peut être n'importe quel nom de classe valide et peut inclure un nom de classe qualifié complet.

Si aucune valeur n'est spécifiée pour cet attribut, le nom de classe de la page compilée est basé sur le nom de fichier de la page et utilise l'espace de noms par défaut ASP.

Si une valeur est spécifiée pour l'attribut ClassName sans un espace de noms complet, l'espace de noms ASP est utilisé, associé au nom de classe spécifié pour créer un nom de classe qualifié complet.

Une autre page peut référencer le nom de classe assigné à la première page en utilisant la directive @ Reference.

Inherits

Définit une classe code-behind pour la page à hériter.

Peut correspondre à toute classe dérivée de la classe Page. Cet attribut est utilisé avec l'attribut CodeFile, qui contient le chemin d'accès au fichier source de la classe code-behind. L'attribut Inherits respecte la casse lors de l'utilisation de C# comme langage de page et ne respecte pas la casse lors de l'utilisation de Visual Basic comme langage de page.

Si l'attribut Inherits ne contient pas d'espace de noms, ASP.NET vérifie si l'attribut ClassName contient un espace de noms. Le cas échéant, ASP.NET essaie de charger la classe référencée dans l'attribut Inherits à l'aide de l'espace de noms de l'attribut ClassName. (Cela suppose que l'attribut Inherits et l'attribut ClassName utilisent tous deux le même espace de noms.)