Bases de données et Interfaçage Web

Mohamed SIDIR

Université Virtuelle de Tunis

2006

Le Web et les Bases de Données

Connecter une base de données au World Wide Web, c'est mettre en place une passerelle entre un serveur http (serveur web) et un système de gestion de bases de données. Cette passerelle dite aussi interface conduit à traiter des techniques associées au monde de l'informatique lié à l'architecture client/serveur, mettant en jeux des langages de programmation orientés web, des protocoles de communication, des langages de manipulation de données, etc.

Ce cours essaiera de répondre à la question suivante :

Comment associer les principes de navigation hypertexte utilisant le protocole http à la gestion de données structurées afin de générer de l'information dynamique en format html, en fonction de demandes (requêtes http) exprimées par l'utilisateur(client) ! ?

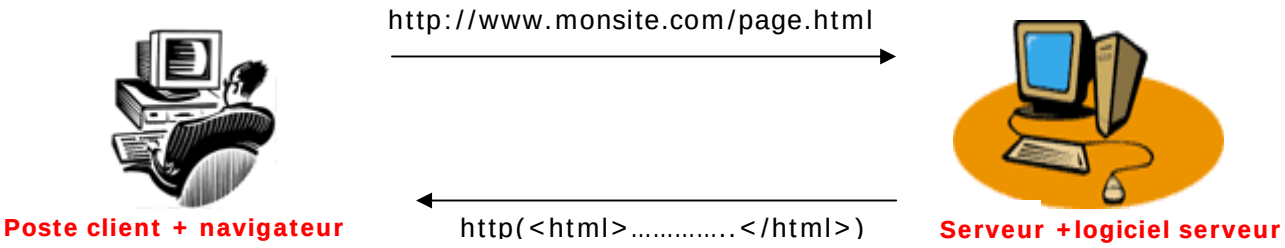
Page web : HTML statique

```
<HTML>
        <HEAD>
                <TITLE>exemple 1</TITLE>
        </HEAD>

        <BODY>
                <P>
                Do bee do bee do ...
                </P>
        </BODY>
</HTML>
```

L'exemple ci-dessus utilise une page web statique utilisant le code HTML.

Lorsque le client lance une requête http pour accéder à cette page (en cliquant sur un lien qui pointe sur cette page ou en tapant l'URL correspondant). Le serveur, quelque part sur le réseau Internet ou Intranet, renvoie le texte dans une enveloppe http en format HTML. Pour une requête http simple, la procédure est illustrée sur la figure suivante.



Les pages statiques sont donc des pages HTML invariables préparées à l'avance. Le serveur renvoie ces pages à l'utilisateur mais n'effectue aucune action particulière. Le code source de la page affichée par le navigateur sur le poste client est identique au code source de la page web installée sur le serveur.

Les avantages *du code HTML pur, statique* :

1- le code HTML est facile à comprendre, à corriger et à produire.
2- tous les navigateurs, en principe, sont capables de l'afficher correctement !!
3- les requêtes sont traitées rapidement par le serveur en utilisant moins de ressources.

Les inconvénients :

1- il est difficile de faire évoluer les pages
2- manque d'interactivité
3- les contenus ne sont pas personnalisés à la demande du client

pour ces raisons, le code HTML statique n'est plus à la mode et ne peut plus répondre aux exigences liées à la création, l'animation et la mise à jour d'un site web.

De nombreuses technologies complémentaires ont été développées pour répondre à ces limites. Du coté client (client side) : on trouve Javascript, Vbscript, Les feuilles de style en cascade ou le DHTML, XML, XSL, les applettes Java.
L'utilisation de scripts coté client distingués au navigateur ne doit pas laisser penser que l'on se trouve en présence de sites dynamiques.

Exemple :

```
<HTML>
        <HEAD>
                <TITLE>exemple 1</TITLE>
        </HEAD>

        <BODY>
                <P>
                Do bee do bee do ...
                </P>
        </BODY>
</HTML>
```

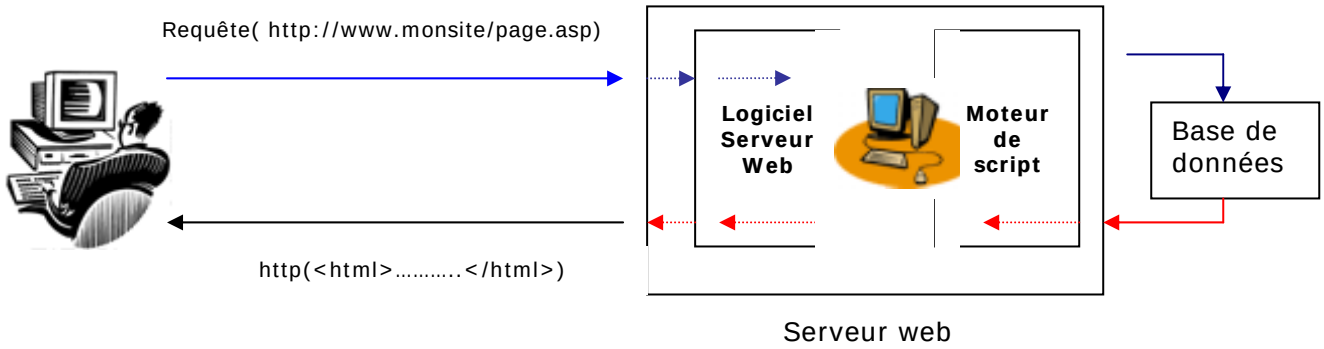
Equivalent

```
<HTML>
        <HEAD>
                <TITLE>exemple 1
                </TITLE>
        </HEAD>
        <BODY>
        <P>
        <SCRIPT LANGUAGE="JavaScript">
document.write("Do bee do bee do...");
        </SCRIPT>
                </P>
        </BODY>
</HTML>
```

Page dynamique : Script côté serveur

Les technologies utilisées autorisent la connexion aux bases de données. On note, à titre d'exemple, le CGI, ASP, PHP, IDC/HTX, etc.

La figure ci-dessous schématise la transmission des données dans le cas des scripts coté serveur.



Le travail le plus important d'écriture de script coté serveur consiste à connecter le serveur web au serveur de données dont l'architecture ou le type peuvent être différents. (une seule machine peut être serveur web et serveur de données !).

Du client vers le serveur

L'utilisateur remplit un formulaire en ligne ou clique sur un lien dynamique. Les données sont envoyées au serveur web. Ce dernier est géré par un logiciel serveur appelé aussi serveur httpd

Quelques serveurs httpd

- IIS (Internet Information Serveur) est un serveur Web/Ftp/Gopher
- Apache : logiciel libre, donc gratuit et dont le code source est ouvert donc disponible.
- HTTPD, le serveur de Bob Denny
- WebSite
- NCSA httpd
- CERN httpd
- Serveur Web Personnel Server (PWS)
- .....

Le serveur httpd réagit en transmettant les données à un script en vue de leur traitement, en se connectant et puis interrogeant une base de données.

Du serveur vers le client :

Les pages web sont assemblées à partir des résultats produits par le serveur de données et renvoyées à l'utilisateur dans une enveloppe http, en format HTML.
Lorsque les données arrivent sur le poste client, le navigateur fait de son mieux pour les afficher ...

Architecture Web / Bases de données

Il existe plusieurs modèles d'architecture web couplés aux bases de données :

- L'accès aux bases de données via les scripts CGI (Common Gateway Interface)

- l'accès aux bases de données via les API (Application Programming Interface = Interface de programmation d'application)
- l'accès aux bases de données via les Middleware
- ...

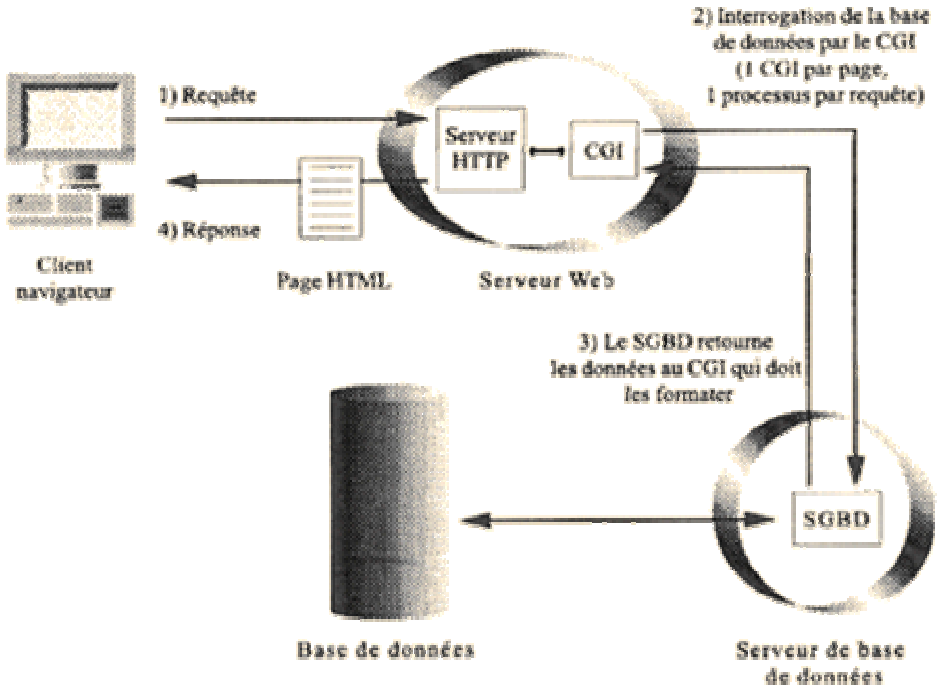
Il y a donc plusieurs approches dont le choix dépend du serveur web et du système d'exploitation utilisés.

Common Gateway Interface (CGI)

La première technique utilisée est le standard multiplateforme *CGI (Common Gateway Interface)* : programme écrit en shell, C ou le Perl. Ce dernier reste le langage typique de gestion des requêtes formulées par une page web. Le CGI est peut être une application exécutable qui offre d'autres possibilités que la récupération de données. Il permet de tirer parti d'autres fonctionnalités du système d'exploitation.

Le serveur web envoie les informations du navigateur web sous la forme de chaînes de caractères, et l'application CGI renvoie une chaîne de contenant le code HTML de la page à retourner au navigateur.

La figure ci-dessous schématise la transmission des données dans le cas des scripts coté serveur utilisant le script CGI.



Source : *Journal Internet Professionnel*, n° 9, mai 1997.

Un programme CGI peut être écrit dans de nombreux langages. La seule condition est en fait, que le langage choisi puisse être exécuté sur une ligne de commande sans faire appel à un autre programme. On utilise sous UNIX les langages PERL, C, C++,shel,

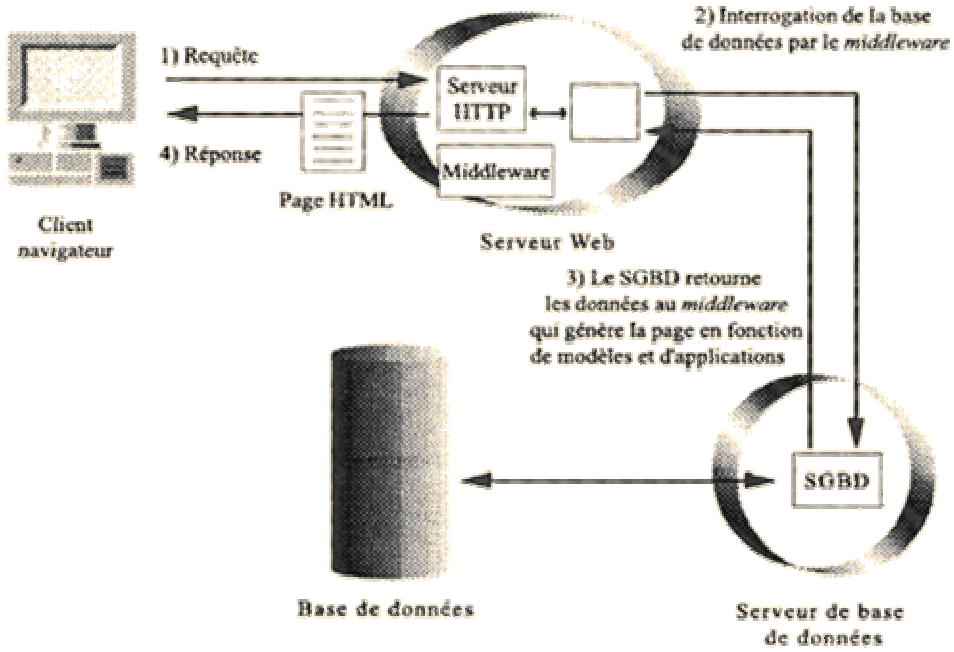
Fotran, Pascal, sous windows on utilise le C, le C++, visual Basic et sur le Macintosh, essentiellement AppleScript. Si les scripts sont écrits dans un langage de programmation qui demandent à être compilés (C, C++, Fortran, Pascal…), les fichiers sources se trouvent généralement dans le répertoire /cgi-src, mais les fichiers compilés sont dans le répertoire /cgi-bin. Si les scripts sont écrits dans un langage de programmation directement interprétable (PERL, shel UNIX, AppleScript,…), ils doivent se trouver dans le répertoire /cgi-bin. Dans tous les cas (surtout sous UNIX), il faut vérifier que les fichiers possèdent bien les permissions d'exécution.

Accès aux bases de données via les middlewares

Un middleware est un traducteur qui met en relation deux programmes essayant de changer des informations. Utiliser cette technique consiste à développer une couche logiciel entre l'application et le réseau.

Exemple :

Le SGBD Sybase utilise un middleware, nommé Adaptive Server



Source : *Journal Internet Professionnel*, n° 9, mai 1997.

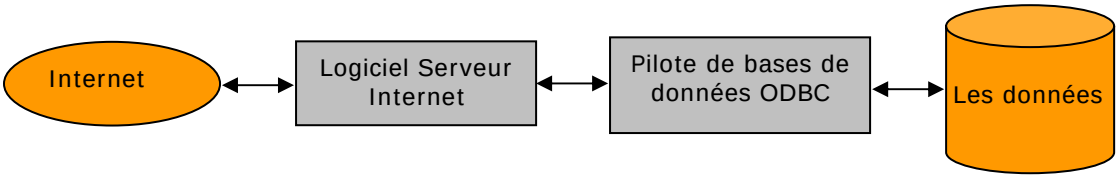
Accès aux bases de données via les API

Cette technologie est plus récente, il existe deux formes d'API auxquelles se conforment les principales bases de données : NSAPI de Netscape et ISAPI (Internet Server Application Programming Interface). Ce mode d'accès est très utilisé, actuellement, mais il reste très lié aux types de serveurs http et aux bases de données. En effet, les ISAPI sont une interface propre à Microsoft. Si on utilise IIS (Internet Information Serveur) de windows NT ou un autre serveur compatible. ISAPI permet d'utiliser un ensemble de fonctions utilisables depuis la plupart des langages de programmation. Ces fonctions permettent de récupérer les informations en provenance du navigateur et de lui transmettre en retour des pages générées dynamiquement. ISAPI se connecte aux bases de données en utilisant une connexion spéciale.

Cette connexion a besoin d'une combinaison matériel-logiciel entre la base de données et le monde extérieur. Elle utilise une couche de traduction adéquate permettant les mouvements de données entre Internet et la base de données.
Pour mettre en place physiquement cette connexion, la machine serveur doit disposer d'un logiciel de connexion à l'Internet qui supporte la connexion ODBC (IIS, website, PWS, ...).

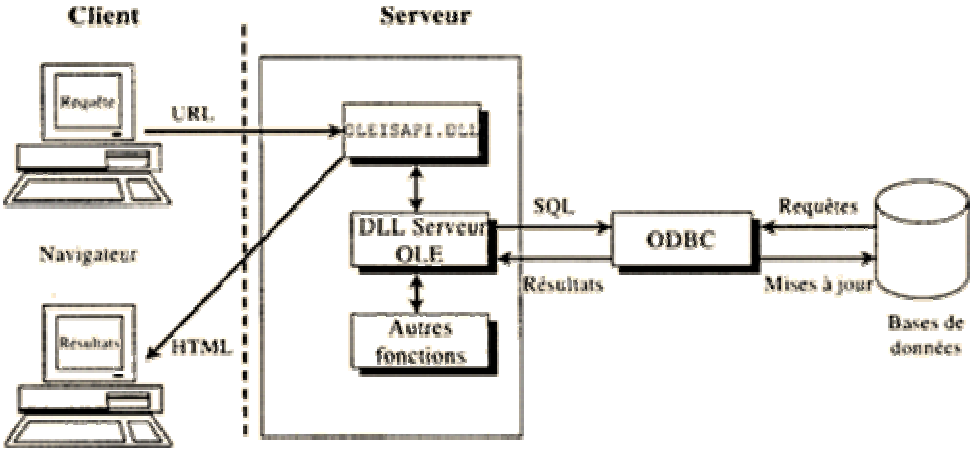
Couche ODBC

Open **D**ata**b**ase **C**onnectivity est une interface standard d'accès à des bases de données hétérogènes (oracle, sql Server, DB2, dbase,...) .
La communication s'effectue via un pilote (driver) qui effectue le lien entre un moteur de script et le SGBD.



Il est possible de se connecter directement à l'API d'Internet Information Serveur en créant une bibliothèque de liens dynamiques (DLL – Dynamic Link Library) au standard Windows. Cependant, on peut utiliser la technologie ActiveX (OLE), référencée sous le nom de OLEISAPI (API OLE d'Internet Server), ce qui nécessite la création d'un serveur OLE Automation. Un serveur OLE Automation est essentiellement une collection d'objets programmables en vue d'exécuter certaines tâches. On peut utiliser Visual Basic pour créer des DLL de type serveur OLE.

La figure ci-dessous schématise la transmission des données dans le cas des scripts coté serveur utilisant l'application OLEISAPI.



OLEISAPI a ouvert la création de pages dynamiques au moyen d'une DLL ActiveX compilée. Toutefois, la technique de transmission de données et de mise en oeuvre à

laquelle elle a recours n'est performante et efficace que pour des taches mineures et les opérations effectuées sur les intranets (contrairement aux sites Internet à volumes élevés). Par ailleurs, là aussi, chaque modification apportée à la page ou marquage exige une nouvelle compilation de la DLL.

Conclusion

Perl fut le premier langage de script côté serveur. Depuis, d'autres ont été mis au point : **ASP** (**A**ctive **S**erver **P**age), **IDC** (**I**nternet **D**atabase **C**onnector, voir cours de Mr Cochard), **PHP** (**P**ersonnel **H**ome **P**age),…
Il existe également des langages conçus pour certains types d'utilisateurs : par exemple, **TCL** facilite les calculs mathématiques complexes dans le domaine des sciences.

Introduction aux Active Server Pages

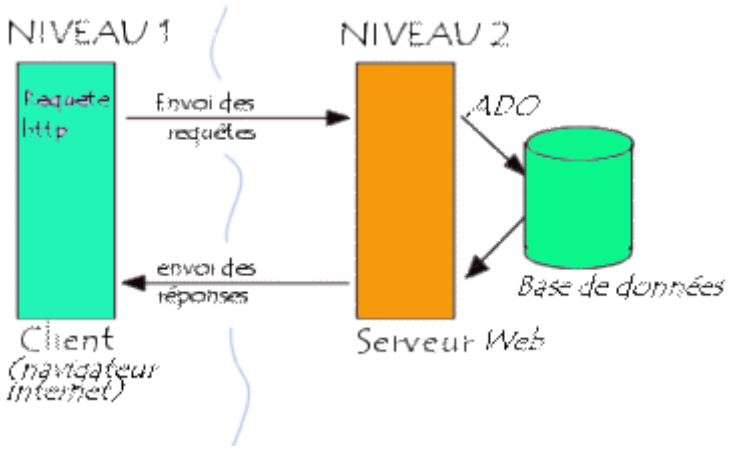
Section 1 : Présentation des Active Server Pages

Les **ASP** (*Active Server Pages*) sont un standard Microsoft permettant de développer des applications Web interactives. Une page web ASP aura donc un contenu dynamique, pouvant être différent selon certains paramètres (des informations stockées dans une base de données, les préférences de l'utilisateur,...) alors que la page web "classique" (dont l'extension est *.htm* ou *.html*) affichera continuellement la même information.

Il s'agit en réalité d'un langage de script, interprété, et exécuté du côté du serveur (au même titre que les scripts CGI,...) et non du côté client (les scripts écrits en Java script ou les applets Java s'exécutent dans le navigateur de la personne connectée à un site).

Les ASP sont intégrables au sein d'une page Web en HTML, à l'aide de balises spéciales **< % ... % >**. Le code compris à l'intérieur de ces balises est interprété par le serveur, le résultat (en code HTML) est renvoyé au navigateur du client.

Ainsi, les Active Server Pages s'inscrivent dans une architecture 3-tier, c'est à dire qu'un serveur supportant les ASP peut servir d'intermédiaire entre le navigateur du client et une base de données. L'accès à cette base de données est donc transparent, grâce à la technologie **ADO** (*ActiveX Data Object*). Celle-ci fournit en effet les éléments nécessaires à la connexion au système de gestion de bases de données, et à la manipulation des données grâce au langage SQL.



Section 2 : Les objets de base des Active Server Pages

Les Active Server Pages sont basées sur des objets manipulés par le serveur et permettant de réaliser diverses applications. Les 6 objets de base sont:

- **Application** : il contient les informations en cours, les différentes variables.
- **Request** : il sert à récupérer les informations envoyées au serveur par un formulaire
- **Response** : il sert à envoyer les réponses au client (le navigateur)
- **Server**: il contient les informations propres au serveur
- **Session** : il permet de conserver des informations d'une page à l'autre

Section 3 : Caractéristiques des Active Server Pages

A l'origine, les ASP ont été conçues pour fonctionner sur le serveur Web de Microsoft intitulé **Microsoft IIS** (*Internet Information Server*). Ce serveur web, mis au point par Microsoft en 1996, a l'avantage d'être gratuit, et fonctionne sous Microsoft Windows NT.

Aujourd'hui, cette technologie est disponible sur d'autres serveurs web que celui de Microsoft. Il a d'abord été porté sur le serveur Netscape FastTrack par Chili!Software, puis sur d'autres serveurs dont Apache, avec le module ***Apache::ASP***, ce qui rend possible la création de sites Web utilisant la technologie des ASP sur de nombreuses plate-formes (Unix,Linux,PowerPC,...).

Les ASP peuvent donc être programmés dans différents langages (Visual basic, Perl, Langage C++,Java,...), ce qui augmente les possibilités offertes par ces ASP.

Section 4 : Interprétation du code par le serveur

- Le serveur reconnaît qu'il s'agit d'un fichier ASP grâce à son extension .asp
- Le serveur lit le fichier asp
- Le serveur exécute les instructions : il "passe" en mode ASP dès qu'il rencontre une balise indiquant que les lignes qui suivent sont en code ASP.
- Le serveur transmet les instructions rencontrées à l'interpréteur.

- L'interpréteur exécute l'instruction puis envoie les sorties éventuelles à l'interpréteur
- A la fin du script, le serveur transmet le résultat au client (le navigateur)

Le code ASP stocké sur le serveur n'est donc **jamais** visible directement par le client puisque dès qu'il en demande l'accès, le serveur l'interprète.

Section 5 : Implantation au sein du code HTML

ASP se présente d'une certaine façon comme une extension du langage HTML (au même titre que les **SSI** (*Server Side Include*, des commandes imbriquées dans le code HTML interprétées par le serveur)). Afin de définir les scripts inclus dans le code HTML et interprétés par le serveur, ASP définit une nouvelle balise (ou tag) HTML: **<% %>.** A l'intérieur de ces balises, on trouve des scripts écrits dans un langage pouvant être du:

- VBScript
- JavaScript
- Jscript
- Perl
- Java
- C++
- ...

Section 6 : Présentation des objets ASP

Dans sa version 3.0, ASP est architecturé autour de 6 objets internes, qui comprennent des méthodes permettant d'effectuer les principaux traitements sur les données. Ces objets constituent ce que l'on appelle le modèle objet, et sont:

* L'objet **Application** qui représente le site.

* L'objet **Session** qui représente l'utilisateur. Il permet de conserver les données relatives à l'utilisateur d'une page du site à une autre

* L'objet **Response** qui représente le résultat à afficher sur le navigateur.

* L'objet **Request** qui permet de traiter les informations en provenance du client par l'intermédiaire de formulaires. Il permet de récupérer les valeurs des champs de requête issus du formulaire du navigateur.

* L'objet **Server** qui représente le serveur. Il permet d'en gérer les paramètres, ainsi que d'instancier les objets utilisateurs.

* L'objet **ObjectContext** qui désigne la transaction courante. Il sert à gérer les traitements de la transaction.

Section 7 : La structure d'un objet ASP

Les objets ASP constituent l'essentiel du moteur de scripts ASP. Ce sont les principaux éléments regroupant les propriétés (valeurs) et les méthodes (traitements) utilisables dans les scripts.

En réalité un objet est composé de trois types d'entités:

- **Les collections**: ce sont des structures de données (une sorte de tableau) contenant un ensemble de valeurs repérées par une clé. Chaque objet peut donc contenir plusieurs collections de variables. Une valeur d'une collection d'un objet est accessible par la syntaxe suivante:

  objet.colllection("clé")

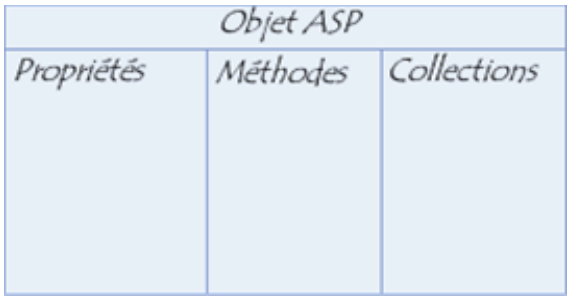
- **Les propriétés**: ce sont des valeurs spécifiques directement accessibles. On accède à une propriété d'un objet ASP par la syntaxe:

  objet.propriete

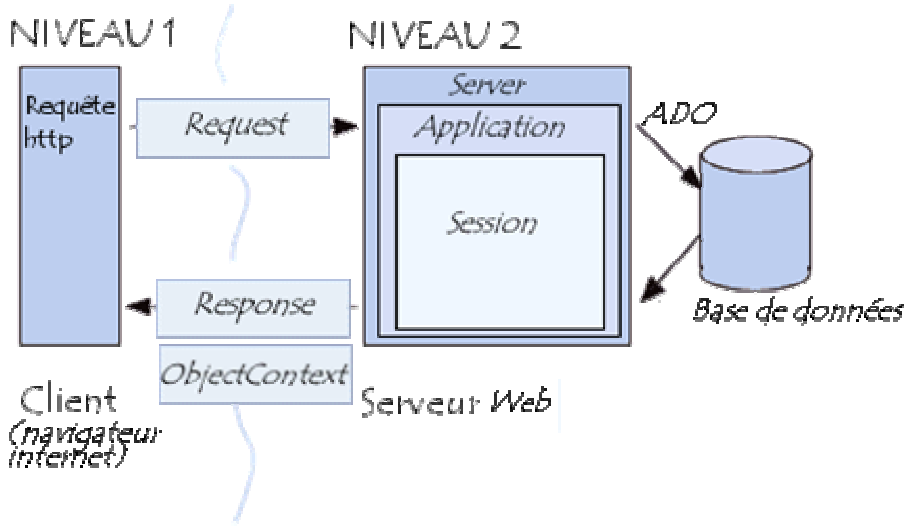
- **Les méthodes**: ce sont des fonctions standards associées à un objet, permettant de manipuler des valeurs passées en argument. La syntaxe d'une méthode ressemble donc à ceci:

  objet.methode(arguments)

Voici la façon par laquelle il est possible de représenter un objet ASP:



La manipulation des propriétés et des méthodes des objets internes permet la récupération d'informations sur la requête ainsi que la création de la réponse HTTP. De cette façon le modèle objet peut être représenté conceptuellement de la façon suivante dans la communication client/serveur:



Lors d'une transaction Client/serveur (c'est-à-dire à l'appel d'une page ASP par l'intermédiaire de son URL ou bien de l'envoi d'un formulaire HTML), un objet ***Request*** est créé. Il contient les informations sur la requête HTTP.

L'objet ***Server*** contient les informations concernant l'état du serveur et fournissant des méthodes pouvant être utilisées dans les script.

L'objet ***Application*** sert à stocker les informations utilisées lors de l'exécution des scripts.

L'objet ***Session*** sert plus spécifiquement à conserver les informations sur l'utilisateur lors du passage d'une page ASP à une autre. L'objet

ObjectContext est un objet plus spécifique, utilisé lors de transactions gérées par le logiciel *MTS* (*Microsoft Transaction Server*).

Présentation de 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.

L'objet **Response** permet en réalité de manipuler l'ensemble des informations à destination du navigateur du client, comme par exemple l'écriture de cookies...

Les constituants de l'objet Response

L'objet Response possède une seule collection, mais aussi de nombreuses propriétés et méthodes:

Objet Response		
Collections	**Methodes**	**Propriétés**
Cookies	Buffer	AddHeader
	CacheControl	AppendToLog
	CharSet	BinaryWrite
	ContentType	Clear
	Expires	End
	ExpiresAbsolute	Flush
	isClientConnected	Redirect
	Status	Write
	PICS	

La majorité des propriétés et des méthodes de l'objet Response correspondent à des champs de la réponse HTTP.

L'envoi de données au navigateur

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

```
<%@ LANGUAGE="VBSCRIPT" %>
<HTML>
<HEAD>
<TITLE>Exemple de script ASP</TITLE>
</HEAD>
<BODY>
```

```
<% Response.write("coucou") %>

</BODY>
</HTML>
```

Ce script est totalement inutile dans la mesure ou un simple fichier HTML pourrait donner le même résultat. L'intérêt de cette méthode est de pouvoir gérer des variables ainsi que des chaînes:

```
<%@ LANGUAGE="VBSCRIPT" %>
<HTML>
<HEAD>
<TITLE>Exemple de script ASP</TITLE>
</HEAD>
<BODY>

<% for i=1 to 10

Response.write("Compte a rebours: " & 10 - i & "<br>")
%>

</BODY>
</HTML>
```

La méthode **Write()** gère la conversion des données entrées en paramètre en chaîne de caractères. Pour envoyer des données binaires, il est possible d'utiliser la méthode **BinaryWrite()**.

L'envoi de cookies au navigateur

En dehors de la possibilité d'envoyer une page Web au navigateur du client, l'objet **Response**, permet aussi de lui envoyer des **cookies**, c'est-à-dire de stocker des données dans un fichier du client sous la forme de paires nom/valeur. L'objet Response fournit la collection **cookies** pour effectuer les opérations d'envoi de cookies au navigateur. Le protocole HTTP permet de spécifier les valeurs de cookies dans les en-têtes http. L'envoi de cookies au navigateur dans la page ASP doit donc se faire avant tout envoi dans le corps de la réponse.

Pour stocker une valeur dans un cookie appelé *CCMCookie*, il suffit d'utiliser la commande suivante:

```
<%@ LANGUAGE="VBSCRIPT" %>
<% response.cookies("CCMCookie") = "Valeur" %>
```

Pour stocker plusieurs valeurs associées à des index dans un cookie appelé *CCMCookie*, il suffit d'utiliser la commande suivante:

```
<%@ LANGUAGE="VBSCRIPT" %>
<% response.cookies("CCMCookie")("Index1") = "Valeur1"
<% response.cookies("CCMCookie")("Index2") = "Valeur2"
<% response.cookies("CCMCookie")("Index3") = "Valeur3"
...
%>
```

En réalité, le cookie ainsi créé n'aura qu'une durée limitée à celle de l'utilisation du navigateur ; il sera donc effacé à la fermeture de celui-ci. Pour y remédier, il suffit de définir la propriété *expires*, qui définit la "date limite de péremption du gâteau".

```
<% response.cookies("CCMCookie").expires = #24/12/2000#
```

La collection **cookies** possède d'autres propriétés que *expires*:

- *domain* définit le nom du serveur pour lequel les valeurs du cookie sont accessibles
- *path* définit le chemin sur le serveur pour lequel les valeurs du cookie sont accessibles
- *secure* permet d'indiquer que le cookie ne peut être envoyé que lors d'une connexion sécurisée (SSL,S-HTTP,...)

Section 8 : Présentation de l'objet Request

Le rôle de l'objet **Request** est de permettre de récupérer la réponse HTTP envoyée par le navigateur au serveur, c'est-à-dire la page Web demandée par le client.

Les constituants de l'objet Request

L'objet **Request** possède une seule méthode et propriété et de nombreuses collections:

Objet Request		
Collections	**Methode**	**Propriété**
ClientCertificates	TotalBytes	BinaryRead
Cookies		
Form		
QueryString		
ServerVariables		

La majorité des propriétés et des méthodes de l'objet Request correspondent à des fonctions ou propriétés permettant de manipuler les champs de la requête HTTP.

La réception de données

Pour comprendre comment utiliser l'objet **Request**, il est nécessaire de connaître la manière par laquelle les données sont envoyées au navigateur grâce à la requête HTTP. Pour cela, il faut se référer à la section "formulaire HTML".

Les formulaires HTML se créent à l'aide de la balise **<FORM>** contenant des boutons, des champs, des listes et/ou des cases à cocher, repérés par des noms auxquels sont associées des valeurs, fonction de la saisie des utilisateurs, puis d'un bouton de soumission du formulaire qui envoie l'ensemble des informations au script indiqué en tant qu'attribut ***Action*** de la balise ***FORM*** selon la méthode **GET** ou **POST**. 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 suivant que la méthode utilisée pour l'envoi du formulaire pouvant être **GET** ou **POST**.

- **la méthode GET** permet d'envoyer les éléments du formulaire au travers de l'URL du script, en ajoutant l'ensemble des paires nom/valeur à l'URL du script, séparé de celui-ci par un point d'interrogation, ce qui donne un URL du type:

  http://nom_du_serveur/Dfoad/Formation.asp?champ1=valeur1&champ2=valeur2

  Toutefois, la longueur de la chaîne URL étant limitée à 255 caractères, les informations situées au-delà de cette limite seront perdues. De plus, cela crée une URL surchargée dans la barre d'adresse d'un navigateur et peut dévoiler des informations sensibles comme un mot de passe...

- **la méthode POST** est une bonne alternative à la méthode GET. Cette méthode code les informations de la même façon que la méthode GET (encodage URL et paires nom/valeur) mais elle envoie les données à la suite des en-têtes HTTP, dans un champ appelé *corps de la requête*. De cette façon la quantité de données envoyées n'est pus limitée, et est connue du serveur grâce à l'en-tête permettant de connaître la taille du *corps de la requête*.

La collection QueryString

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

```
<%@ LANGUAGE="VBSCRIPT" %>
<% Request.QueryString("Champ") %>
```

La collection Form

Alors que la collection **QueryString** permet de récupérer de façon simple les données envoyées au script ASP par l'intermédiaire de l'URL (c-est-à-dire par la méthode GET), la collection **Form** permet de manipuler les données envoyées par un formulaire utilisant la méthode POST. La récupération de la valeur associée à un champ s'effectue par la syntaxe suivante :

```
<%@ LANGUAGE="VBSCRIPT" %>
<% Request.Form("Champ") %>
```

La collection Cookies

La collection **Cookies** permet de récupérer les valeurs d'un cookie, c'est-à-dire un fichier présent sur le disque du client contenant des données envoyées par le serveur (un cookie peut être créé grâce à l'objet Response).
L'accès aux données d'un cookie se fait de la manière suivante:

```
<%@ LANGUAGE="VBSCRIPT" %>
<% Variable = Request.Cookies("NomCookie")("Element") %>
```

Il est possible de parcourir l'ensemble des Cookies par le script suivant:

```
<%@ LANGUAGE="VBSCRIPT" %>

<% For Each Element in Request.Cookies %>

<%
Response.write(Element + " = ")
Response.write(Request.Cookies(Element)) %>
<%BR%>
<% Next %>
```

Section 9 : La collection ServerVariables

La collection **ServerVariables** de l'objet *Request* contient les en-têtes HTTP de la requête, pouvant parfois donner des informations très utiles sur les visiteurs ou le navigateur du client.

La syntaxe permettant de récupérer ces en-têtes est la suivante:
Request.ServerVariables("NOM-EN-TETE")

Voici les principaux en-têtes utiles:

Nom de l'en-tête	**Description**
ALL_HTTP	Type de méthode utilisée par le client (ie *POST* ou *GET*)
CONTENT_TYPE	Type de contenu du corps de la requête (par exemple *text/html*). Voir types MIME
METHOD	Type de méthode utilisée par le client (ie *POST* ou *GET*)
REFERER	URL du lien à partir duquel la requête a été effectuée
REMOTE_ADDR	Adresse IP du client
HTTP_ACCEPT_LANGUAGE	Langage attendu par le browser (anglais par défaut)
HTTP_USER_AGENT	Chaîne donnant des informations sur le client, comme le nom et la version du navigateur, du système d'exploitation

Pour compter les visiteurs d'un site par exemple, il peut être intéressant de stocker l'adresse IP de ces derniers et de compter le nombre d'adresses IP stockées et différentes chaque jour. Voici le code permettant de stocker dans une variable l'adresse IP d'un visiteur, et qui l'affiche:

```
<%@ LANGUAGE="VBSCRIPT" %>
<%
IP = Request.ServerVariables("REMOTE_ADDR")
Response.write("Votre adresse IP est : " + IP) %>
<%BR%>
```

Active Serveur Pages (ASP)
Travaux Pratiques

Exemple 1 : page dynamique

```
<HTML>
<head> <Title>Un exemple d'une page Asp</title>
<Body BGColor= Wheat>
<p><font Face = verdana size=3>le serveur Web accueillant à <%=Time%>
le <%=Date%></font></P>
</body>
</html>
```

:

Exemple 2 : Première page ASP avec un script VbScript

```
<%@ LANGUAGE="VBScript" %>
<HTML>
<HEAD>
<TITLE>Exemple d'une page asp</TITLE>
</HEAD>

<BODY>
<%For x=1 to 6 %>
<FONT FACE="ARIAL" SIZE=<%=x%>>
Bonjour tout le monde - ASP vous salut ! - taille police : <%=x%>
</FONT>
<br><br>
<%Next%>
</BODY>
</HTML>
```

objet Request.

Cet objet permet de récupérer des informations.

Exemple 3 : Passer des variables à une autre page web par un formulaire

Saisie d'un formulaire dans une page html

```
<HTML>
<HEAD>
<TITLE>Exemple 3 d'une page asp</TITLE>
</HEAD>

<BODY>
<br><Br><br><Br>
```

```
<FORM METHOD="POST" Action = "Form_reponse.asp">
<P Align="CENTER"> Nom : <INPUT TYPE ="TEXT" NAME="nom"></P>
<P Align="CENTER"> Prénom : <INPUT TYPE ="TEXT" NAME="prenom"></P>
<P Align="CENTER"> <INPUT TYPE ="SUBMIT"></P>
</FORM></BODY></HTML>
```

Page ASP nommée **Form_reponse.asp**, permettant d'afficher les données saisies dans le formulaire précédent

```
<%@ LANGUAGE="VBScript" %>
<HTML>
<HEAD>
<TITLE>Script Form_reponse</TITLE>
</HEAD>

<BODY>
<P>Votre nom est : <%=Request.form("nom")%></P>
<P>Votre prénom est : <%=Request.form("prenom")%> </P>
</BODY></HTML>
```

Exemple 4 :

Passer des variable à une autre page Web par l'URL (lien hypertexte)

a) Le passage de variable dans l'URL de la page

```
<HTML>
<HEAD>
<TITLE>Script URL</TITLE>
</HEAD>
<Body>
<Br><Br><Br><Br>
<P align="center">
<A Href ="URL_rep.asp?prenom=philippe&nom=Deschamp&pays=France">Cliquer sur ce lien
</A> pour passer des variable vers la page suivante</P>
</body></html>
```

b) Le programme de récupération des variables passées dans l'URL de la page

```
<%@ LANGUAGE="VBScript" %>
<HTML>
<HEAD>
<TITLE>URL_rep</TITLE>
</HEAD>

<BODY><br><Br><Br>

<P ="CENTER">La variable nom contient :
<%=Request.queryString("nom")%></P>

<P Align=center>La variable prénom contient :
<%=Request.queryString("prenom")%></P>
```

```
<P Align=center>La variable Pays contient :
<%=Request.queryString("pays")%></P>

</BODY></HTML>
```

Accéder à une base de données avec ASP

L'intérêt principal des pages dynamiques sur un site web (ou sur un intranet), est de fournir un accès aux bases de données. Un des composants de l'ASP est conçu pour répondre à ce besoin. Il s'agit de ADO « ActiveX Data Objects » = objet de données activeX, permet d'accéder à n'importe quel système de gestion de base de données pour lequel existe un pilote ODBC (Open DataBase Connectivity).
nous allons illustrer son utilisation par des exemples concrets.

La page Asp contient :

- le nom de la source de données définie pour le pilote ODBC.
- La ou les requête(s) en SQL.
- Objet(s) de donnée(s) activeX

Pour tous ces exemples, nous allons utiliser une base de données réalisée sous MS Access et nommée Entreprise.mdb ; pour simplifier, nous supposerons que cette base de données, pour les premiers exemples, est réduite à une seule table, nommée **personnels** dont la structure est :

Nom du champ	Paramètres
N°	Numeroauto
Nom	Texte
Prenom	texte
Age	Numéro
Adres_email	texte

Les connecteurs ODBC, le DSN Système

- Les connecteurs ODBC (Open DataBase Connectivity) permettent d'établir une liaison avec une base de données.
- Connexion au DSN système : cette connexion sert à définir le nom de la source de données systèmes, c'est à dire le nom du fichier de base de données qui pourra être interrogé directement sans passer par le logiciel qui a été utilisé pour créer la base (Access ou SQL Server).

Connexion au DSN système

```
<%@ LANGUAGE="JSCRIPT" %>
<%
conn = Server.CreateObject("ADODB.Connection");
conn.open("base");

sql = ("SELECT enregistrement1, enregistrement2,... FROM Personnels ")
res= conn.execute(sql);
%>
```

TP1 : Afficher le contenu d'une table dans une page ASP

Afficher le contenu de la table « table1 » dans un tableau de la forme :

Nom	Prénom	Age	Email	Téléphone

Pour afficher un enregistrement dans une page asp utilisez le script suivant :

<%=res("enregistrement")%>

TP2 : Trier les données d'une table dans une page ASP

Pour trier les enregistrements dans une page asp, il faut utiliser dans le Script de la requête la condition (WHERE condition)

TP3 : Ajouter des enregistrements dans une table avec ASP

L'ajout d'enregistrements dans une base de données par le web constitue une étape essentielle. Elle permet principalement deux choses d'un intérêt évident : la mise à jour du site par l'administrateur ou par toute personne autorisée, et la contribution des visiteurs à la vie de votre site (rédaction d'une petite annonce, forum de discussion….)

1- Création d'une page html avec un formulaire de saisie

Nom :	Nom de ce champ de saisie = « nom »
Prénom :	Nom de ce champ de saisie = « prenom »
Adresse email :	Nom de ce champ de saisie = « adres_email »

Dans une page ASP pour actualiser une base de donnée, on utilise la commande **INSERT**

Exemple : insert.asp

```
<%@ LANGUAGE="JSCRIPT" %>
<%
conn = Server.CreateObject("ADODB.Connection");
 conn.open ("base");

sql="INSERT INTO table1 (Nom, prenom, adres_email) ";
sql=sql+ " VALUES ('"+Request.Form("Nom")+"','"+Request.Form("prenom")+"',
'"+Request.Form("adres_email")+"');";
rs = conn.execute(sql);
%>
<html>
<head>
<title>insertion</title>
</head>
<body>La base de données est bien actualisée !</body></html>
```