

Introduction aux pages Web dynamiques

Nicolas Stroppa

stroppa@enst.fr
Brique AWEB

30 mars 2005

Plan

- 1 Introduction
 - Rappels sur le protocole HTTP, le langage HTML, et les URL
 - Les pages dynamiques : une nécessité
- 2 Principes et exemples de pages dynamiques
 - L'interaction Serveur ↔ Programme
 - Exemples d'utilisation
- 3 Suivi de connexion
 - Les champs cachés
 - Les cookies
 - Les sessions en PHP

Plan

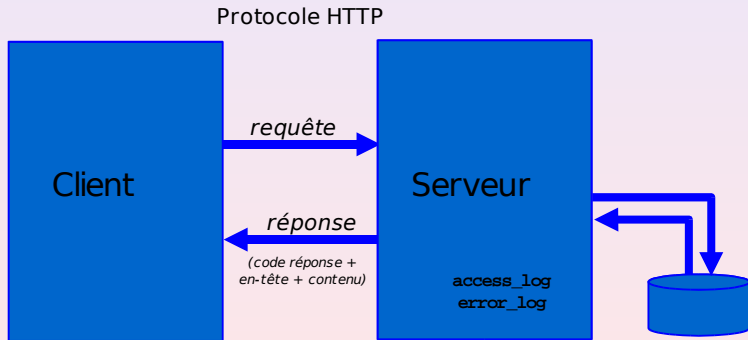
- 1 Introduction
 - Rappels sur le protocole HTTP, le langage HTML, et les URL
 - Les pages dynamiques : une nécessité
- 2 Principes et exemples de pages dynamiques
 - L'interaction Serveur ↔ Programme
 - Exemples d'utilisation
- 3 Suivi de connexion
 - Les champs cachés
 - Les cookies
 - Les sessions en PHP

Le Web

- Repose sur 3 standards :
 - Le protocole **HTTP** (diffusion de documents)
 - Le langage **HTML** (structuration des documents et navigation par liens hypertextes)
 - La spécification **URL** (localisation des documents)
- Doit sa naissance à Tim Berners-Lee (fin années 80)
- Est standardisé par le World Wide Web Consortium (W3C)
<http://www.w3.org/>

Le protocole HTTP

- Interaction entre un client et un serveur web



Le protocole HTTP - Quelques caractéristiques

- HTTP = HyperText Transfer Protocol
- HTTP est un protocole axé question/réponse (connexion unique, sans état, mode « pull »)
- Un échange HTTP = requête(s) de la part d'un client (e.g. demande de page HTML), puis réponse(s) du serveur.
- Les requêtes et réponses sont de la forme : en-tête + corps

Le protocole HTTP

- En-têtes (headers)
 - Requête : méthode, URL demandé, type de client, jeu de caractères attendu par le client, ...
 - Réponse : type du contenu, date de début de transfert, ...
- Méthodes principales
 - Commande **GET** : récupération d'un document
 - Commande **POST** : envoi de données à un programme

```
tricatel -> cours_aweb: telnet www.infres.enst.fr 80
Trying 137.194.160.20...
Connected to infres2-160.enst.fr.
Escape character is '^]'.
HEAD /index.html HTTP/1.0

HTTP/1.1 200 OK
Date: Fri, 05 Nov 2004 18:49:45 GMT
Server: Apache/1.3.22 (Unix) PHP/3.0.15 PHP/4.1.0
Connection: close
Content-Type: text/html

Connection closed by foreign host.
```

Les URL

- Les URL (**Uniform Resource Locator**) sont composées d'un « protocole » (`http`, `ftp`, `mailto`, `file`) d'un séparateur (`:`) et d'une chaîne propre au protocole (`//www.enst.fr`) \Rightarrow `http://www.enst.fr`
- Une URL permet d'identifier de façon unique un document sur Internet (protocole + chemin complet du fichier)

Le langage HTML

- Est un langage à balises (famille SGML)
 - Exemple : `<html><body>document</body></html>`
- Utilise des **liens hypertextes**, i.e. liens entre documents, à l'aide de la notion d'URL
 - Inclusion d'une image :
``
 - Lien vers un autre document :
`La page de Toto`
- Permet la navigation de page en page (web = toile).

Le langage HTML

Permet d'intégrer avec un site web à l'aide de formulaire de saisie.

- On englobe le formulaire dans :

```
<form method="méthode" action="action">  
formulaire</form>
```

- Un élément de saisie :

```
<input type="type" name="nom" size="taille" />  
(type = text, password, hidden, file, image, checkbox, radio,  
submit, reset)
```

- Une liste d'options :

```
<select name="nom"><option>option1</option>  
<option>option2</option></select>
```

Pourquoi des pages dynamiques ?

- Pages statiques (HTML)

- Avantages :

- pas de traitement par le client ou le serveur : contenu « valide », accès rapide
 - travail fait une fois pour toutes

- Inconvénients :

- contenu vieillissant très rapidement
 - pas d'interactivité possible

- Pages dynamiques (CGI, PHP)

- Avantages :

- interactivité
 - contenu pouvant facilement être mis à jour

- Inconvénients :

- problèmes de sécurité
 - accès plus lent (traitement des requêtes)
 - erreurs à l'exécution possibles

Pourquoi des pages dynamiques ?

- Pages statiques (HTML)

- Avantages :

- pas de traitement par le client ou le serveur : contenu « valide », accès rapide
 - travail fait une fois pour toutes

- Inconvénients :

- contenu vieillissant très rapidement
 - pas d'interactivité possible

- Pages dynamiques (CGI, PHP)

- Avantages :

- interactivité
 - contenu pouvant facilement être mis à jour

- Inconvénients :

- problèmes de sécurité
 - accès plus lent (traitement des requêtes)
 - erreurs à l'exécution possibles

Pourquoi des pages dynamiques ?

- Pages statiques (HTML)

- Avantages :

- pas de traitement par le client ou le serveur : contenu « valide », accès rapide
 - travail fait une fois pour toutes

- Inconvénients :

- contenu vieillissant très rapidement
 - pas d'interactivité possible

- Pages dynamiques (CGI, PHP)

- Avantages :

- interactivité
 - contenu pouvant facilement être mis à jour

- Inconvénients :

- problèmes de sécurité
 - accès plus lent (traitement des requêtes)
 - erreurs à l'exécution possibles

Pourquoi des pages dynamiques ?

- Pages statiques (HTML)
 - Avantages :
 - pas de traitement par le client ou le serveur : contenu « valide », accès rapide
 - travail fait une fois pour toutes
 - Inconvénients :
 - contenu vieillissant très rapidement
 - pas d'interactivité possible
- Pages dynamiques (CGI, PHP)
 - Avantages :
 - interactivité
 - contenu pouvant facilement être mis à jour
 - Inconvénients :
 - problèmes de sécurité
 - accès plus lent (traitement des requêtes)
 - erreurs à l'exécution possibles

Pourquoi des pages dynamiques ?

- Pages statiques (HTML)
 - Avantages :
 - pas de traitement par le client ou le serveur : contenu « valide », accès rapide
 - travail fait une fois pour toutes
 - Inconvénients :
 - contenu vieillissant très rapidement
 - pas d'interactivité possible
- Pages dynamiques (CGI, PHP)
 - Avantages :
 - interactivité
 - contenu pouvant facilement être mis à jour
 - Inconvénients :
 - problèmes de sécurité
 - accès plus lent (traitement des requêtes)
 - erreurs à l'exécution possibles

Deux types de dynamicité

- Dynamicité « côté client » : applets, JavaScript, ...
 - Avantages :
 - pas de traitement côté serveur
 - interactivité immédiate : animations, tests de validité des données à transmettre
 - Inconvénients :
 - problèmes de sécurité et de compatibilité
- Dynamicité « côté serveur » : CGI, PHP
 - Avantages :
 - programme indépendant du client
 - possibilité de traiter les formulaires
 - Inconvénients :
 - problèmes de sécurité toujours ...

Deux types de dynamicité

- Dynamicité « côté client » : applets, JavaScript, ...
 - Avantages :
 - pas de traitement côté serveur
 - interactivité immédiate : animations, tests de validité des données à transmettre
 - Inconvénients :
 - problèmes de sécurité et de compatibilité
- Dynamicité « côté serveur » : CGI, PHP
 - Avantages :
 - programme indépendant du client
 - possibilité de traiter les formulaires
 - Inconvénients :
 - problèmes de sécurité toujours ...

Deux types de dynamicité

- Dynamicité « côté client » : applets, JavaScript, ...
 - Avantages :
 - pas de traitement côté serveur
 - interactivité immédiate : animations, tests de validité des données à transmettre
 - Inconvénients :
 - problèmes de sécurité et de compatibilité
- Dynamicité « côté serveur » : CGI, PHP
 - Avantages :
 - programme indépendant du client
 - possibilité de traiter les formulaires
 - Inconvénients :
 - problèmes de sécurité toujours ...

Deux types de dynamicit 

- Dynamicit  « c t  client » : applets, JavaScript, ...
 - Avantages :
 - pas de traitement c t  serveur
 - interactivit  imm diate : animations, tests de validit  des donn es   transmettre
 - Inconv nients :
 - probl mes de s curit  et de compatibilit 
- Dynamicit  « c t  serveur » : CGI, PHP
 - Avantages :
 - programme ind pendant du client
 - possibilit  de traiter les formulaires
 - Inconv nients :
 - probl mes de s curit  toujours ...

Deux types de dynamicité

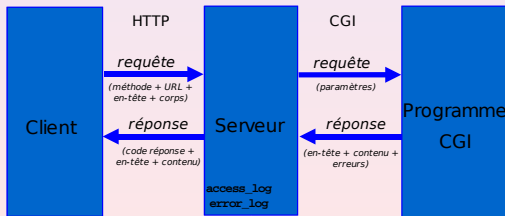
- Dynamicité « côté client » : applets, JavaScript, ...
 - Avantages :
 - pas de traitement côté serveur
 - interactivité immédiate : animations, tests de validité des données à transmettre
 - Inconvénients :
 - problèmes de sécurité et de compatibilité
- Dynamicité « côté serveur » : CGI, PHP
 - Avantages :
 - programme indépendant du client
 - possibilité de traiter les formulaires
 - Inconvénients :
 - problèmes de sécurité toujours ...

Plan

- 1 Introduction
 - Rappels sur le protocole HTTP, le langage HTML, et les URL
 - Les pages dynamiques : une nécessité
- 2 Principes et exemples de pages dynamiques
 - L'interaction Serveur ↔ Programme
 - Exemples d'utilisation
- 3 Suivi de connexion
 - Les champs cachés
 - Les cookies
 - Les sessions en PHP

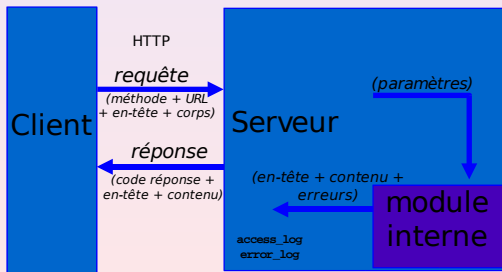
Serveurs HTTP et programmes CGI

- CGI définit une interface d'échange de données entre un serveur HTTP et des applications externes (API).
- L'application externe est un programme indépendant du serveur. Elle peut être théoriquement écrite dans n'importe quel langage.
 - Programmes compilés (C, Ada, ...)
 - Scripts (Perl, Python, Shell Unix, ...)
 - Dans la pratique, on a souvent recours à des bibliothèques



Serveurs HTTP avec modules internes

- Le serveur HTTP utilise un module/plugin interne : PHP, Perl, ASP
- Le serveur doit être configuré/compilé pour accepter et comprendre le langage. Ce sont donc des langages interprétés. Certains sont dédiés à la programmation dynamique de site web (PHP, ASP).



Fonctionnement de l'interface

- 1 Le client passe des arguments au serveur HTTP, sous la forme d'une chaîne de caractères représentant une liste de paires clé-valeur. Exemple :
`nom=Dupont&prenom=Jacques&age=47.`
- 2 Le serveur réceptionne les données et les transmet au programme/module.
- 3 Le programme :
 - s'exécute dans l'environnement du serveur (= peut accéder à certaines variables)
 - traite la requête et renvoie :
 - sur la sortie standard : l'en-tête + le contenu (php) / seulement le contenu
 - sur la sortie d'erreur : les éventuels messages d'erreurs
- 4 Le serveur :
 - renvoie la réponse (en-tête + contenu) au client
 - archive au besoin les messages d'erreurs

Fonctionnement de l'interface

- 1 Le client passe des arguments au serveur HTTP, sous la forme d'une chaîne de caractères représentant une liste de paires clé-valeur. Exemple :
`nom=Dupont&prenom=Jacques&age=47.`
- 2 Le serveur réceptionne les données et les transmet au programme/module.
- 3 Le programme :
 - 1 s'exécute dans l'environnement du serveur (= peut accéder à certaines variables)
 - 2 traite la requête et renvoie :
 - sur la sortie standard : l'en-tête + le contenu (php) / seulement le contenu
 - sur la sortie d'erreur : les éventuels messages d'erreurs
 - 3 Le serveur :
 - renvoie la réponse (en-tête + contenu) au client
 - archive au besoin les messages d'erreurs

Fonctionnement de l'interface

- 1 Le client passe des arguments au serveur HTTP, sous la forme d'une chaîne de caractères représentant une liste de paires clé-valeur. Exemple :
`nom=Dupont&prenom=Jacques&age=47.`
- 2 Le serveur réceptionne les données et les transmet au programme/module.
- 3 Le programme :
 - 1 s'exécute dans l'environnement du serveur (= peut accéder à certaines variables)
 - 2 traite la requête et renvoie :
 - sur la sortie standard : l'en-tête + le contenu (php : seulement le contenu)
 - sur la sortie d'erreur : les éventuels messages d'erreurs
- 4 Le serveur :
 - renvoie la réponse (en-tête + contenu) au client
 - archive au besoin les messages d'erreurs

Fonctionnement de l'interface

- 1 Le client passe des arguments au serveur HTTP, sous la forme d'une chaîne de caractères représentant une liste de paires clé-valeur. Exemple :
`nom=Dupont&prenom=Jacques&age=47.`
- 2 Le serveur réceptionne les données et les transmet au programme/module.
- 3 Le programme :
 - 1 s'exécute dans l'environnement du serveur (= peut accéder à certaines variables)
 - 2 traite la requête et renvoie :
 - sur la sortie standard : l'en-tête + le contenu (php : seulement le contenu)
 - sur la sortie d'erreur : les éventuels messages d'erreurs
- 4 Le serveur :
 - renvoie la réponse (en-tête + contenu) au client
 - archive au besoin les messages d'erreurs

Fonctionnement de l'interface

- 1 Le client passe des arguments au serveur HTTP, sous la forme d'une chaîne de caractères représentant une liste de paires clé-valeur. Exemple :
`nom=Dupont&prenom=Jacques&age=47.`
- 2 Le serveur réceptionne les données et les transmet au programme/module.
- 3 Le programme :
 - 1 s'exécute dans l'environnement du serveur (= peut accéder à certaines variables)
 - 2 traite la requête et renvoie :
 - sur la sortie standard : l'en-tête + le contenu (php : seulement le contenu)
 - sur la sortie d'erreur : les éventuels messages d'erreurs
- 4 Le serveur :
 - renvoie la réponse (en-tête + contenu) au client
 - archive au besoin les messages d'erreurs

Fonctionnement de l'interface

- 1 Le client passe des arguments au serveur HTTP, sous la forme d'une chaîne de caractères représentant une liste de paires clé-valeur. Exemple :
`nom=Dupont&prenom=Jacques&age=47.`
- 2 Le serveur réceptionne les données et les transmet au programme/module.
- 3 Le programme :
 - 1 s'exécute dans l'environnement du serveur (= peut accéder à certaines variables)
 - 2 traite la requête et renvoie :
 - sur la sortie standard : l'en-tête + le contenu (php : seulement le contenu)
 - sur la sortie d'erreur : les éventuels messages d'erreurs
- 4 Le serveur :
 - renvoie la réponse (en-tête + contenu) au client
 - archive au besoin les messages d'erreurs

Fonctionnement de l'interface

- 1 Le client passe des arguments au serveur HTTP, sous la forme d'une chaîne de caractères représentant une liste de paires clé-valeur. Exemple :
`nom=Dupont&prenom=Jacques&age=47.`
- 2 Le serveur réceptionne les données et les transmet au programme/module.
- 3 Le programme :
 - 1 s'exécute dans l'environnement du serveur (= peut accéder à certaines variables)
 - 2 traite la requête et renvoie :
 - sur la sortie standard : l'en-tête + le contenu (php : seulement le contenu)
 - sur la sortie d'erreur : les éventuels messages d'erreurs
- 4 Le serveur :
 - renvoie la réponse (en-tête + contenu) au client
 - archive au besoin les messages d'erreurs

Fonctionnement de l'interface

- 1 Le client passe des arguments au serveur HTTP, sous la forme d'une chaîne de caractères représentant une liste de paires clé-valeur. Exemple :
`nom=Dupont&prenom=Jacques&age=47.`
- 2 Le serveur réceptionne les données et les transmet au programme/module.
- 3 Le programme :
 - 1 s'exécute dans l'environnement du serveur (= peut accéder à certaines variables)
 - 2 traite la requête et renvoie :
 - sur la sortie standard : l'en-tête + le contenu (php : seulement le contenu)
 - sur la sortie d'erreur : les éventuels messages d'erreurs
- 4 Le serveur :
 - renvoie la réponse (en-tête + contenu) au client
 - archive au besoin les messages d'erreurs

« Hello world! » avec un CGI en Perl

```
#!/usr/local/bin/perl

print "Content-type: text/html\n";
print "\n";
print "<html>\n";
print "<head><title>Hello world !!!</title></head>\n";
print "<body>\n";
print "Hello world?\n";
print "</body></html>\n";
```

Pour installer : mettre le fichier source en lecture et exécution pour tous (en réalité l'utilisateur apache) et ajouter l'extension .cgi

« Hello world! » avec un CGI en C

```
#include <stdio.h>
#include <stdlib.h>

void main (int argc, char ** argv) {
    printf("Content-type: text/html\n");
    printf("\n");
    printf("<html>\n");
    printf("<head><title>Hello world !!!</title></head>\n");
    printf("<body>\n");
    printf("Hello world?\n");
    printf("</body></html>\n");
}
```

Attention : dans ce cas, il faut compiler et faire l'édition de liens pour obtenir l'exécutable !

« Hello world! » avec le module PHP

```
<html>
<head><title>Hello world !!!</title></head>
<body>
<?php echo "Hello world?\n"; ?>
</body></html>
```

- Le code PHP est à l'intérieur des balises <?php et ?>.
- Il n'y a pas besoin de rajouter l'en-tête
- Pour installer, il suffit que le fichier ait l'extension .php

Méthodes d'envoi des paramètres

- **GET** : données codées dans l'URL. Exemple :
`http://www.toto.com/form1.cgi?nom=Dupont&prenom=Jacques`
 - Les données sont stockées dans la variable `QUERY_STRING`
 - Avantage : facile à déboguer
 - Inconvénient : taille des données limitée
- **POST** : données dans le corps de la requête.
 - Les données sont stockées dans la variable `BUFFER`
 - Avantage : taille illimitée (précisée dans la variable `CONTENT_LENGTH`)
 - Inconvénient : debug plus délicat

En PHP, il y a création automatique des variables portant le nom des clés. (Ici, la variable **nom** sera automatiquement créée avec pour valeur **Dupont**.)

Un exemple de formulaire

```
<html>
<head><title>Exemple de formulaire</title></head>
<body>

<form method="post" action="script.cgi">
<input type="radio" name="action" value="name">
Afficher mon nom
<input type="radio" name="action" value="with_email">
Afficher mon nom et mon e-mail<br />

Nom : <input type="text" name="name" size="30"><br />
E-mail : <input type="text" name="email" size="30"><br />

<input type="submit" value="OK">
<input type="reset" value="RAZ">
</form>
</body></html>
```

Script Python traitant le formulaire

```
#!/usr/local/bin/python

import cgi

print """Content-type: text/html

<html>
<head><title>Résultats du formulaire</title></head>
<hr><h1>Résultats du formulaire</h1><hr>"""

# Récupération des résultats dans un dictionnaire
fields = cgi.SvFormContentDict()
```

Les valeurs des champs du formulaire sont associées aux noms dans le dictionnaire obtenu (objet Python).

Script Python traitant le formulaire

```
# Extraction des champs
name = email = radio = None
if fields.has_key('name'):
    name = fields['name']
if fields.has_key('email'):
    email = fields['email']
if fields.has_key('action'):
    radio = fields['action']

# Générer la fin de la page HTML
print "<ul>"
if name:
    print "<li>Nom : %s</li>" %(name)
if (email) and (radio == 'avec_email'):
    print "<li>E-mail : %s</li>" %(email)
print "</ul>"
print "<hr>"
print "</body></html>"
```

Script PHP traitant le formulaire

```
<html>
<head><title>Résultats du formulaire</title></head>
<hr><h1>Résultats du formulaire</h1><hr>
<ul>

<?
if isset($name) {
    echo "<li>Nom : $name</li>";
}

if (isset($email) and $radio == 'avec_email') {
    echo "<li>E-mail : $email</li>";
}
?>

</ul><hr></body></html>
```

Plan

- 1 Introduction
 - Rappels sur le protocole HTTP, le langage HTML, et les URL
 - Les pages dynamiques : une nécessité
- 2 Principes et exemples de pages dynamiques
 - L'interaction Serveur ↔ Programme
 - Exemples d'utilisation
- 3 **Suivi de connexion**
 - Les champs cachés
 - Les cookies
 - Les sessions en PHP

Le problème

- Le protocole HTTP n'est pas connecté \Rightarrow impossible de garder une réelle liaison entre le client et le serveur.
- Inadapté aux « sessions » client/serveur dans lesquelles on doit garder une trace de l'échange (identification, historique, liaison avec une BD...).

Suivi de connexion

Trois méthodes :

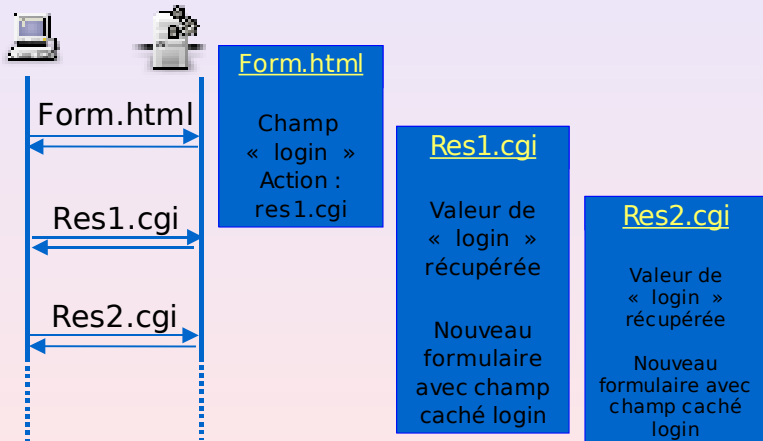
- Avec des *champs cachés*. Uniquement pour une suite d'étapes bien précises. Exemple :

```
<input type="hidden" name="page" value="1">
```
- Avec des *cookies*
 - Information fournie par le serveur mais stockée par le client
- Avec une « *session* » *PHP*
 - On récupère un identifiant par champ caché ou cookie
 - Toutes les autres variables liées à l'identifiant sont récupérables de façon transparente.

Les champs cachés

- Un champ permet de transmettre au serveur une variable de manière transparente.
- Le serveur récupère tous les champs fournis par le formulaire.
- **Problème** : Pour une variable donnée, on ne connaît pas le type de champ qui lui était associé dans le formulaire, i.e. on détecte les champs uniquement par leur nom.
- Objectif des champs cachés : récupérer une variable, la traiter, et la réintroduire à nouveau dans l'affichage d'un nouveau formulaire.

Les champs cachés (2)



Cookies

- Un cookie est composé de :
 - Un nom et une valeur : **id=123**
 - Un domaine : **nom_du_serveur** (défaut : celui de la page)
 - Un path : arborescence du site (par défaut « / »)
 - Une date d'expiration : date, nombre de secondes (par défaut la durée de la session)
- Envoi d'une information du serveur vers le client :
 - Transmise dans l'en-tête HTTP de réponse
 - Exemple : **Set-Cookie: message=coucou; expires=Tue, 08-Jan-2002; Version=1;**
- Envoi d'une information du client vers le serveur :
 - A chaque fois que l'URL requise par le client correspond au domaine et au chemin d'un cookie présent chez le client
 - L'information est récupérée à la requête suivante (variable HTTP_COOKIE)

Cookies en PHP

- **Recevoir** : En PHP, un cookie est reçu de la même façon que dans le cas d'un formulaire : une variable du nom du cookie est créée automatiquement.
- **Envoyer** : Pour envoyer, on utilise la fonction `setcookie`.

```
setcookie("nom du cookie", "valeur", ...)
```

```
<? setcookie(name, "Dupont"); ?>  
<html>...
```

Attention : Erreur,
l'en-tête est déjà fini !

```
<? setcookie(name, "Dupont"); ?>  
<html>...
```

Classe « session » en PHP

- PHP 4 et 5 : inclus dans la distribution de base, via des fichiers stockés sur disque
- On utilise un identifiant de session via un champ caché ou un cookie. Transparent via `session_start` qui gère un cookie tout seul. Une fois la session ouverte, on sauve/charge toute variable de façon transparente.

La suite...

- L'avenir du web est dynamique côté client et serveur.
- Au niveau serveur : PHP et serveurs d'applications (Tomcat, Websphere, etc.)
- Au niveau client : clients riches (Javascript, XUL, XAML, Actionscript, etc.)

Références



Site de référence :

<http://www.php.net>



Rasmus Lerdorf, Kevin Tatroe

Programming PHP.

O'Reilly, 2002.



David Sklar

Learning PHP 5.

O'Reilly, 2004.



GNU/Linux magazine Hors-Série n 20

PHP 5

Sept./Oct. 2004.