



Bases de la programmation web



Sommaire

- Principes généraux de la programmation Web
- Passage d'information via HTTP:
 - Paramètres CGI
 - Paramètres en mode `GET`
 - Paramètres en mode `POST`
 - Envois du serveur au client
 - *Cookies*
- Les formulaires HTML

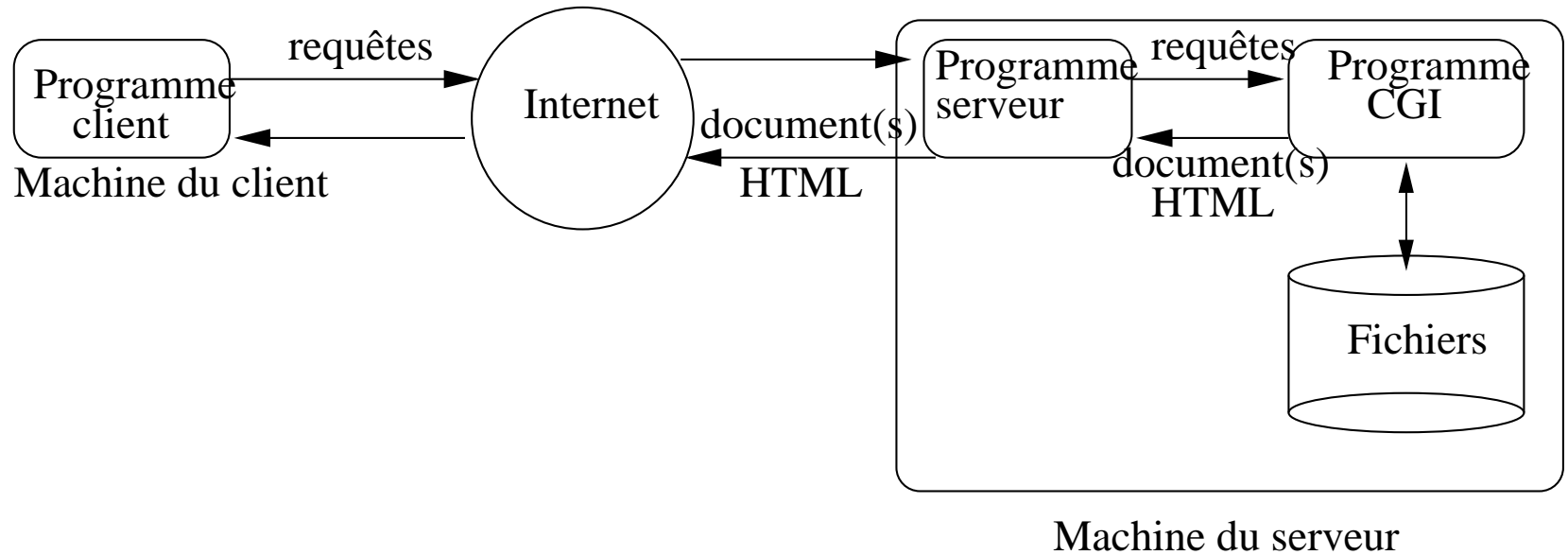


La programmation Web

Principe : on déclenche une action (un programme) sur la machine serveur

- On peut transmettre des informations au programme (typiquement saisies dans un formulaire)
- Le programme Web s'exécute sur le serveur
- À la fin de l'exécution le programme Web renvoie un document (le plus souvent une page HTML)
- Permet de créer **dynamiquement** des pages HTML : utilisé dans à peu près tous les sites Web professionnels.

Architecture programmation Web





Actions du programme Web

Lecture des paramètres

- transmis par le programme serveur

Traitement de la demande

- tout est permis ! Accès aux bases de données, génération d'images, ...

Production du document

- il est renvoyé au programme serveur, qui le transmet au navigateur



les paramètres HTTP

Reçus par le programme We. Essentiellement quatre catégories:

- *Paramètres CGI* : informent le serveur sur le client et le contexte;
- *Paramètres GET* : placés dans une URL
- *Paramètres POST* : placés dans le corps d'un message HTTP
- *Cookies* : paramètres transmis systématiquement, permettant de connaître l'historique des échanges avec un client.

Comment transmettre des paramètres

- Les paramètres GET et POST peuvent être placés dans des *ancres*, ou saisis dans un *formulaire*.
- Les arguments sont fournis dans une *chaîne de caractères* au format
`nom1=val1&nom2=val2&...`
- Deux méthodes principales:
 - GET : la chaîne est transmise dans l'URL
 - POST : la chaîne est transmise dans le corps du message HTTP

POST est préférée, car elle évite les très longues URL



Les paramètres CGI

Ils sont envoyés systématiquement au programme Web.
Voici les principaux.

- `SERVER_NAME` : nom du serveur
- `HTTP_USER_AGENT` : nom du client
- `REQUEST_METHOD` : GET ou POST
- `QUERY_STRING` : la chaîne des paramètres
- `CONTENT_LENGTH` : longueur de la chaîne

Exemple: `HTTP.php`

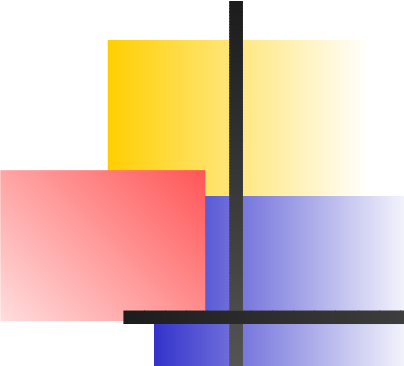


Action du programme

Trois phases:

- Décryptage des paramètres (facile avec PHP!)
- Traitement (accès BD, calculs, etc)
- Transmission du résultat au client Web (*stdout*)

Peut être n'importe quel script (shell, Perl) ou programme exécutable (C, C++)



Programme d'affichage des paramètres

C'est le programme *HTTP.php* : il affiche tous les paramètres reçus.

- `HTTP.php`
- `HTTP.php?parametre=valeur`
- `HTTP.php?nom=rigaux&prenom=philippe`

Autre exemple : un calcul de loyer

On dispose d'un programme qui sait calculer des loyers :

- on lui passe en mode GET le montant à emprunter, le taux et la durée en mois ;
- il récupère les données et lance le calcul ;
- enfin il affiche en retour le tableau d'amortissement.

Exemple d'appel au programme:

`Simul.php?montant=10000&taux=5&duree=24`

Mieux: avec un formulaire (`SimulSimple.html`)

Formulaires HTML:

<FORM>

```
<FORM ACTION=' Simul.php '  
METHOD=' GET ' NAME=' monForm ' >
```

- ACTION est l'URL du script ou du programme à déclencher (en principe, sur le serveur)
- METHOD est GET ou POST (meilleur)
- NAME est le nom du formulaire (utile pour les contrôles JavaScript)
- ENCTYPE est le mode d'encodage des informations. Utile pour transférer des fichiers

Exemple : un formulaire HTML (ExForm1.html)

```
<HTML><HEAD>
<TITLE>Simulation financière</TITLE>
</HEAD><BODY bgcolor="WHITE">

Entrez le montant, le taux et la durée :

<FORM ACTION="HTTP.php" METHOD="GET">

Mt : <INPUT TYPE=TEXT SIZE=20 NAME="mtfin"><BR>
Taux : <INPUT TYPE=TEXT SIZE=20 NAME="taux"></>

Durée : <SELECT NAME='duree'>
        <OPTION VALUE="6">6 mois</OPTION>
        <OPTION VALUE="12">1 an</OPTION>
        <OPTION VALUE="24">2 ans</OPTION>
</SELECT></TD>

<INPUT TYPE=SUBMIT VALUE="Exécuter">
</FORM></BODY></HTML>
```

Champs de formulaire :

< INPUT >

Très général: saisie de texte, ou choix dans des listes.

L'attribut `TYPE` vaut :

- `TEXT` pour les chaînes de caractères
- `HIDDEN` pour les champs cachés
- `CHECKBOX` pour un choix multiple
- `RADIO` pour un choix exclusif
- `SUBMIT` pour déclencher l'action
- `FILE` pour transmettre un fichier

Champs HIDDEN et SUBMIT (ExSubmit.html)

```
<HTML><HEAD>
  <TITLE>Balise INPUT-SUBMIT</TITLE>
</HEAD><BODY bgcolor='white'>

<FORM ACTION='HTTP.php' METHOD='GET' >
  <P>
  <INPUT TYPE='HIDDEN'
        VALUE='valeur' NAME='champ' >

  <P>
  <INPUT TYPE='SUBMIT'
        VALUE='Pas le choix !' >

</FORM>

</BODY></HTML>
```

Champs de saisie de texte (ExInput.html)

```
<HTML><HEAD>
  <TITLE>Exemple de la balise INPUT</TITLE>
</HEAD><BODY bgcolor='white'>
  <FORM ACTION='HTTP.php'
    METHOD='GET' >
    <P>
    Texte :   <INPUT TYPE='TEXT'
              SIZE=20 NAME='texte' > <P>
    Nombre : <INPUT TYPE='TEXT'
              SIZE=4  NAME='nombre' VALUE='
    <P>
    <INPUT TYPE=SUBMIT VALUE='Allons-y' >
  </FORM>
</BODY></HTML>
```


Boutons radio (ExRadio.html)

```
<HTML><HEAD>
  <TITLE>Exemple de la balise INPUT-RADIO</TITLE>
</HEAD><BODY bgcolor='white'>
<FORM ACTION='HTTP.php' METHOD='GET'>
  <P>
    Le beurre : <INPUT TYPE='RADIO'
                  VALUE='1' NAME='choix'> <B
    L'argent du beurre : <INPUT TYPE='RADIO'
                          VALUE='2' NAME='choix'>
  <P>
    <INPUT TYPE=SUBMIT VALUE='Allons-y'>
</FORM>
</BODY></HTML>
```

Transfert de fichiers

```
<HTML><HEAD>
  <TITLE>Exemple de la balise INPUT-FILE</TITLE>
</HEAD><BODY bgcolor='white'>
  <FORM ACTION='HTTP.php'
    METHOD='POST' ENCTYPE='multipart/form-data'
  <P>
Fichier:<INPUT TYPE='FILE' SIZE=20 NAME='file'
  <P>
  <INPUT TYPE=SUBMIT VALUE='Allons-y'>
</FORM>
</BODY></HTML>
```

ExFile.html

Menus déroulant

```
<HTML><HEAD>
  <TITLE>Balise INPUT-SELECT</TITLE>
</HEAD><BODY bgcolor='white'>

<FORM ACTION='HTTP.php' METHOD='GET' >
  Choix :
  <SELECT NAME='choix' >
    <OPTION VALUE='1'>Saucisson
    <OPTION VALUE='2'>Rillettes
    <OPTION VALUE='3'>Paté
  </SELECT>
  <P>
  <INPUT TYPE=SUBMIT VALUE='Allons-y' >
</FORM>

</BODY></HTML>
```

ExChSelect.html

Champ <TEXTAREA>

```
<HTML><HEAD>
  <TITLE>Balise INPUT-SUBMIT</TITLE>
</HEAD><BODY bgcolor='white'>

<FORM ACTION='HTTP.php' METHOD='GET' >
  <P>
    <TEXTAREA NAME='champ' COLS='30' ROWS='4' >
      Entrez votre texte
    </TEXTAREA>
  <BR>
  <INPUT TYPE=SUBMIT VALUE="C'est parti">
</FORM>

</BODY></HTML>
```

ExTextarea.html



Quelques limites du CGI

- Passage des paramètres : pénible
- Pas d'intégration avec HTML
- Lancement d'un programme à chaque requête !
- Mode totalement déconnecté : pas de mémoire des requêtes précédentes

Améliorations avec des langages récents (PHP, ASP, JSP)



Langages de scripts

PHP, ASP, JSP : de nombreux points communs

- Code inclus dans du texte HTML
- Interpréteur inclus dans le serveur
- Instructions exécutées par le serveur
- Décodage automatique des variables provenant du client

Dernier problème : les échanges client/serveur en mode déconnecté

Petit aperçu de PHP

(ExFormFilm.html)

```
<HTML> <HEAD>
<TITLE>HTML avec PHP</TITLE></HEAD>
<BODY bgcolor='white'>
```

J'ai reçu les données suivantes :

```
<TABLE BORDER=3>
<?php
    echo " <TR><TD>Titre:<TD>{$_POST['titre']} ";
    echo " <TR><TD>Année:<TD>{$_POST['annee']} ";
    echo " <TR><TD>Genre:<TD>{$_POST['genre']} ";
    echo " <TR><TD>Pays:<TD>{$_POST['pays']} ";
    echo " <TR><TD>Réalisateur:<TD>{$_POST['reali
    echo " <TR><TD>Résumé:<TD>{$_POST['resume']} "
?>
</TABLE></BODY></HTML>
```



Les avantages

Facilite énormément la programmation CGI :

- Tous les paramètres de la requête HTTP sont automatiquement décodés.
- La création de l'en-tête pour la réponse est simplifiée
- Tout ce qui est HTML « en dur » peut être conservé tel quel

Tout n'est pas rose : il faut maîtriser plein de langages (HTML, JavaScript, CSS, PHP, SQL)

Essentiel : les cookies !

- le serveur envoie au navigateur une variable ayant un nom et une valeur
 - MonServeur dit à MonNavi « stocke la variable `MaVariable` avec la valeur `100` pendant 2 jours »
- Le navigateur transmet ensuite systématiquement la variable au serveur qui l'a créée
 - MonNavi transmet à MonServeur la variable sous la forme `MaVariable=100`

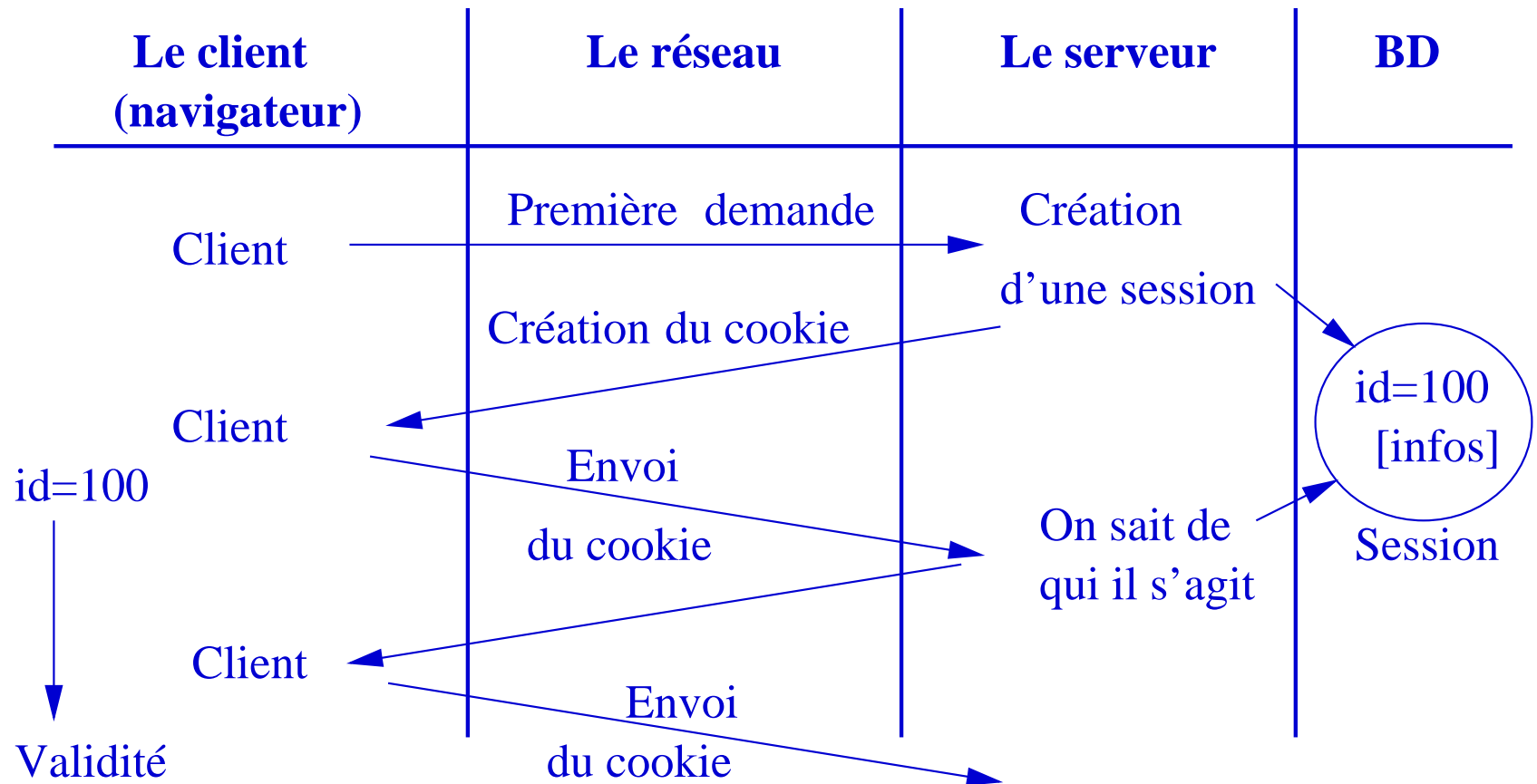


Cookies et sessions web

Les cookies sont **essentiels** pour établir une continuité dans le dialogue client/serveur

- Quand un client se connecte la première fois : le serveur lui associe un identifiant de session
- Le serveur transmet cet identifiant au client sous la forme d'un cookie
- On conserve un historique des actions du client, associé à l'identifiant de session
- Quand le client se connecte à nouveau : on sait l'identifier grâce au cookie.

Session web



L'en-tête Set-Cookie

Il comprend les éléments suivants :

- `nom=valeur` : définition du cookie
- `expires` : date de fin de validité
- `domain` : le domaine pour lequel le cookie est visible.
Par défaut : nom du serveur qui crée le cookie
- `path` : racine des répertoires pour lesquels le cookie est visible. `/'` par défaut
- `secure` : indique que le cookie doit être transmis en mode sécurisé.



Exemple d'un en-tête avec cookie

Set-Cookie:

```
MonCookie=200;
```

```
expires=Mon,24-Dec-2006
```

```
12:00:00 GMT;
```

```
path=/;
```

```
domain=dauphine.fr
```

Tout est optionnel, à l'exception de la première directive.

Script pour cookies

(ExFormCookie.html)

```
<?php
    Header ("Set-Cookie: $nom=$valeur; "
            . "expires=Mon, 24-Dec-2005 "
            . "12:00:00 GMT");
?>
<HTML><HEAD><TITLE>Liste des cookies</TITLE>
</HEAD><BODY bgcolor='white'>
Liste des cookies:<BR>
<?php
    while (list($nom, $valeur)
            =each($HTTP_COOKIE_VARS))
        echo  "<B>Nom</B> : $nom <B> "
            . "Valeur</B> : $valeur<BR>\n" ;
?>
```



Quelques conclusions

Quelques problèmes de la programmation web :

- Pas de connexion : on ne sait jamais vraiment avec qui on dialogue.
- Sécurité : tout le monde peut envoyer n'importe quoi à un script.
- Architecture : comment intégrer un module web à une application plus généraliste ?
- Référencement : comment *faire savoir* ce qu'on propose sur un site.