Programmation des applications sur les serveurs WEB

Jean-Michel Ilié

IUT Paris 5 - département informatique

Intérêt de la programmation sur les serveurs HTTP

- Création dynamique de pages web (HTML, CSS …)
  Conception d'applications partagées où l'IHM est chargée sur un navigateur WEB.

- Accès à d'autres serveurs :
  - ressources partagées : SGBD, SGF
  - services : services WEB

- Traitements spécifiques
  - Adaptation aux versions des navigateurs
  - Authentification des clients
  - Prise en compte de l'état du serveur

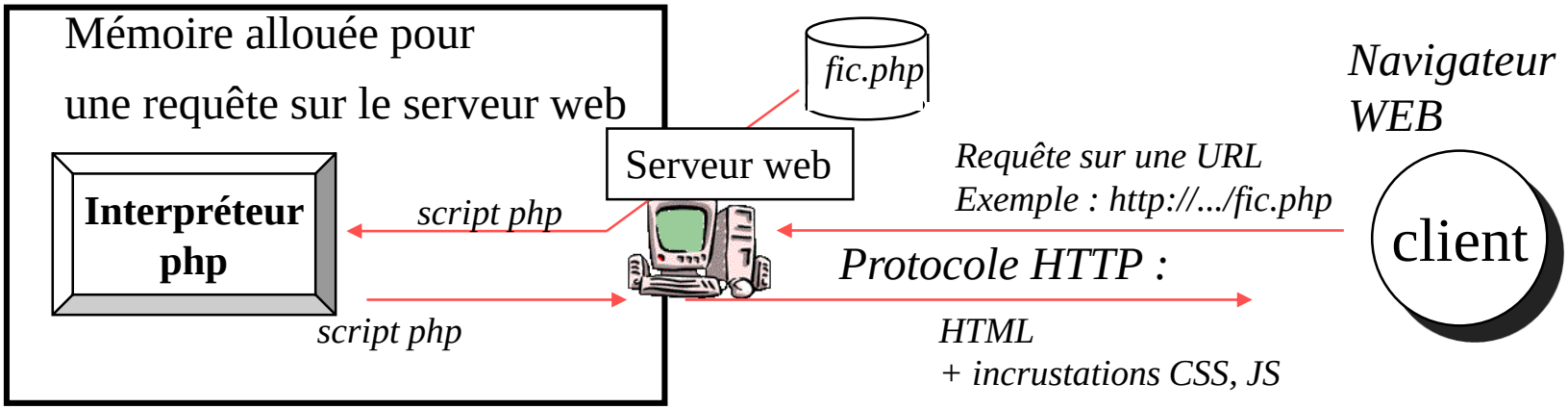
Cas classiques dans une page HTML:

- ***Sollicitation à un lien et chargement***
- ***Traitement d'un formulaire et réponse***

Langages de scripts côté serveur

- Langages interprétés
  - **ASP**.NET Active Server Page (Windows)
  - **PHP** 5 Personnage Home Page (Open Source)
  - **JSP** Java Server Page (Virtual machine)
- **script CGI : à partir de** tous langages compilés
  - fonctions d'interface d'accès à un serveur WEB
- **Servlet-**java sous une machine virtuelle Java
  - vision du serveur WEB avec une classe JAVA spécialisée

Interprétation des scripts sur le serveur WEB



- **Requête HTTP** issue du navigateur client : appel d'une URL (fichier html, php ...)
- **Traitement** : code classique (variables, instructions à réaliser sur le serveur)
  - Entrée : paramètres d'exécution, variables d'environnement du serveur web
  - Sortie : une spécification HTML à produire (print, echo, ...)
- **Résultat** : sortie du script envoyée vers le client qui a à fait la requête.

Programmation WEB à l'IUT

Programmation WEB côté serveur

- **langage PHP (semestre 3 A)**
- langage JAVA (semestre 3 B)

Programmation avancée des interfaces WEB (client riche)

- langages PHP et JAVASCRIPT (semestre 4 A)

Programmation WEB et PHP
plan

- Ecriture dynamique de pages HTML
- Conception des applications WEB
- Construction modulaire du code (paradigme MVC)
- Utilisation de framework MVC orienté-objet

Le langage PHP 5 (Personnage Home Page)

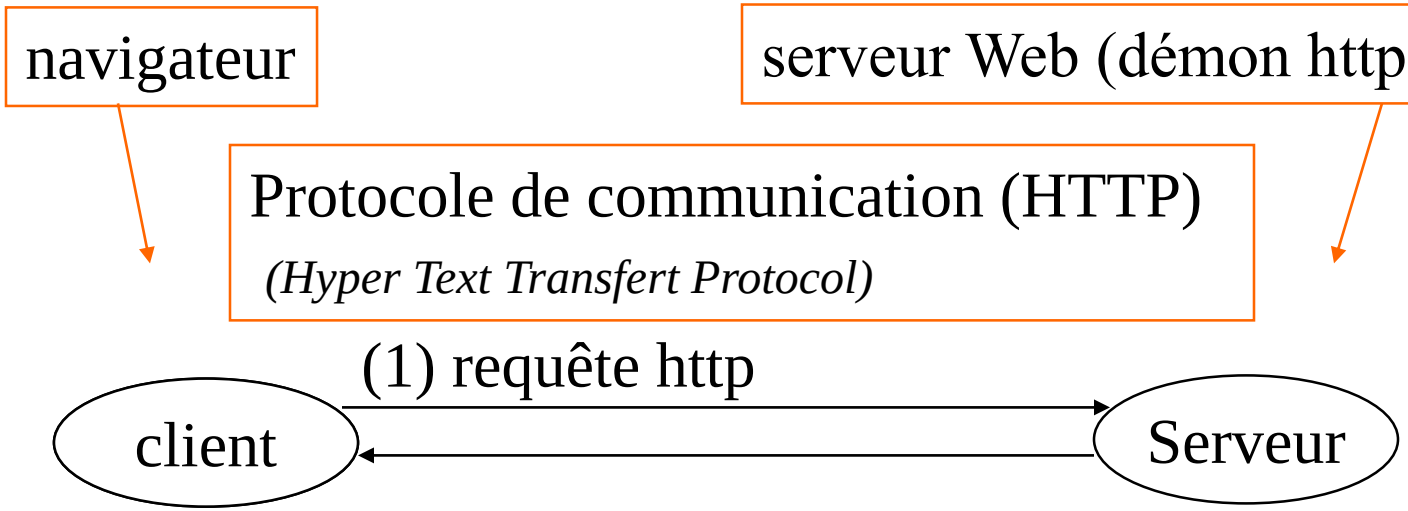
- Serveur **web APACHE** : serveur du monde libre développée pour toutes les plateformes systèmes (Linux, Windows ou service Windows : **module IIS**)
- **Plusieurs écritures compatibles**
  - Syntaxe proche du C -- 1ère approche très simple –
  - Syntaxe orientée objet
  - Utilisation possible des classes JAVA en complément des classes PHP

PHP
langage riche
et présent partout

- Intègre **par défaut les constructions HTML** en balises
- Propose **des fonctions spécifiques** pour les entrées sorties
  - manipulation d'**expressions régulières** (pro-format de chaîne de caractères)
  - indexation et **test aisés de parties** de chaînes
- **Nombreux composants (dll)** : *MySQL, ORACLE, SGF, XML-DOM, THREAD, ZIP*

Rappel : dialogue entre client (navigateur) et serveur WEB

http *requête/réponse de pages indépendante (mode non connecté)*



Rappel : l'URL comme désignation des ressources sur le WEB

URL **U**niform **R**esource **L**ocator

désignation d'une page ou de toute autre ressource sur internet (exécutable, fichier script...)
3 parties + [des arguments] (arguments=options ou paramètres optionnels).

http://www.lip6.fr/welcome.php

protocole

nom du serveur sur lequel est implanté le script

nom local de la page sur le serveur

L'URL précise :
- la page ou le service appelé
- sa localisation
- comment peut-on y accéder
- comment l'interpréter ?

Chaque (fichier de) script a une désignation unique - pas d'ambiguïté possible.

: // et / séparateurs

Exemples d'URL et d'URL paramétré

protocole"://"
[utilisateur[":"motdepasse]@](nom|adresseIP)[:port]
["/"chemin]["/"nomdefichier][#ancre][?paramètres]

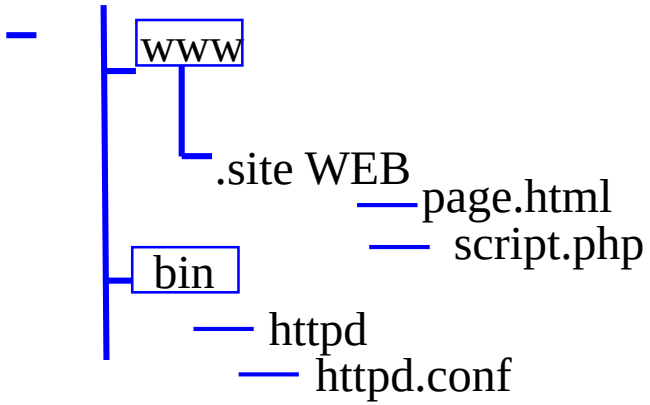
http://www.imag.fr/equipe/sirac/projet.php	*(avec nom du serveur)*
http://123.87.54.251/index.oho	*(adresse IP du serveur)*
http://www.altavista.com/query.exe?iut+paris5	(*paramètre, +: blanc*)
http://www.info.projet/search?nom=ilie&prenom=jm	*(paramètres, &: et)*
http://xenon.inria.fr:8080/hello.html	*(n° port de comm. du serveur)*
http://milo.ecoledoc.lip6.fr/index.html#annuaire	(ancre - pointeur interne)

Le format complet des URLs : norme RFC 1738 et 1808
(voir http://www.w3.org/Addressing/)

Fondement des serveurs WEB

- un répertoire contenant les sites WEB du serveur et leurs ressources (pages...)
- un exécutable implémentant le protocole HTTP pour l'accès aux pages

Exemples : exécutable httpd de Apache (libre), service IIS de MS windows.

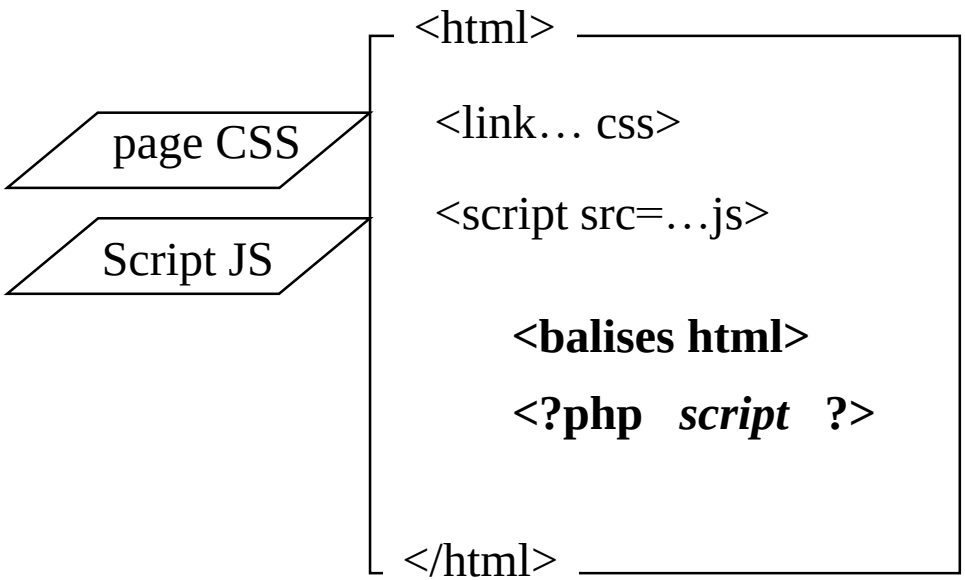


Le serveur WEB est généralement une application configurable (fichier httpd.conf pour apache)

- ***Port** d'écoute par défaut : 80 ou 8080*
- ***www :** répertoire racine par défaut pour les pages WEB*

Vue abstraite d'un script php construisant une page HTML

- Balise HTML classiques (HTML, LINK, …)
-Traitement spécial : **zones de script php-balisées**



Incrustation de scripts php

- Incrustation dans le HTML de scripts PHP encadrés par des balises

```
<?php ...  code interprété par PHP; ... ?>
```

(ou `<? ... ?>`

!!! mais dépend de l'administration du serveur)

- 2ème solution d'incrustation :

```
<script language="php" >
        ... code interprété par l'interpréteur PHP du serveur; …
</script>
```

Style d'écriture php

- Langage faiblement typé (déclaration des types possible mais non nécessaire)
- **variables** introduites par le symbole '**$**‘
  - déclarations globales, valables pour tout le script PHP
  - chaînes de car. explicites : entre quotes " ou '
  - concaténation : '**.**'
- Affichage en php :

  fonctions **echo** ("…")
  - *ou printf ("…") pour des formats %s, %d*

```
<html><body>
<?php  $user="JMI";  ?>

<h1>Bonjour Professeur
<?php
   echo ($user  . " !");
 ?>
</h1> </body></html>
```

Affichage echo : différence entre quotes simple et double

Les variables sont interprétées entre quotes "" (évite des concaténations)

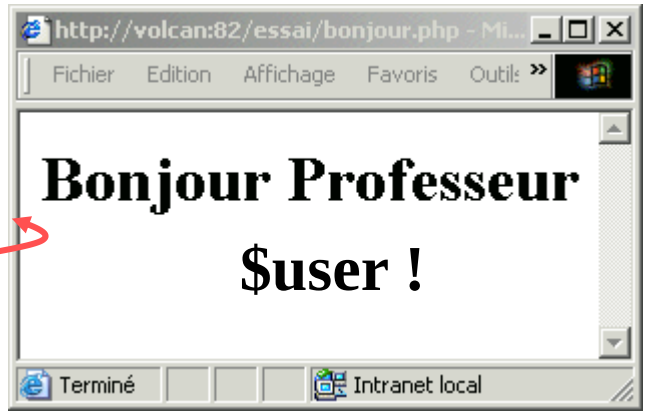
```
<?php
      echo ($user . " !");
?>
```

≡

```
<?php
      echo (" $user  !");
?>
```

Variables non interprétées entre quotes '

```
<?php
      echo (' $user  !');
?>
```



Affichage echo: quelques alternatives pour les balises

```
<tag nomAttribut="valAttr">
```

```
<?php
  echo ("<tag nomAttribut='valAttr'>");
?>
```

Quotes alternés

```
<?php
  echo ("<tag nomAttribut=\"valAttr\" ");
?>
```

Quotes échappés

Affichage sécurisé

```
<?php
    $v = htmlspecialchars($valAttr);
    echo ("<tag nomAttribut = $v >");
?>
```

```
<?php
    $v = htmlspecialchars($valAttr);
?>

<tag nomAttribut = "<?php
                        echo($v);
                    ?>"  >;
```

Affichage avec
échappement automatique
-> codage propre
des caractères html
*-> **évite les injections** de code*
exécutable malveillantes
un script js dans $valAttr
ne sera pas execute sur le navigateur.

Écriture similaire
-> plus claire
***-> séparation du code** HTML*
et du code PHP

Affichage conditionnel des balises HTML

```
<?php   if (expression_bool) {
            echo ("code HTML");
            }
            else  {
            echo ("code HTML");
            }                                   ?>
```

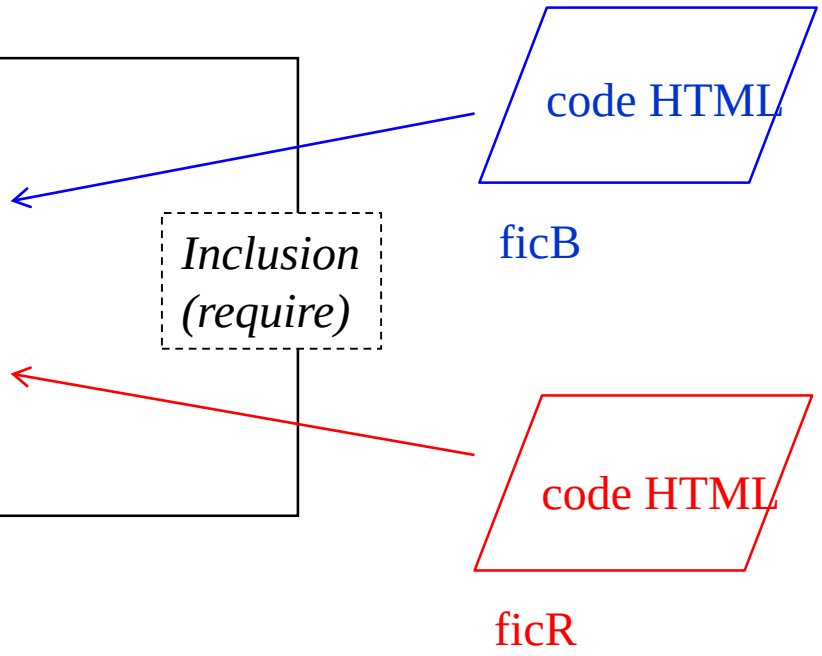
Exemple d'un IF
mais valable
pour toutes structures :
while, for ...

```
<?php                   if (expression_bool) {
?>
    ... code HTML sans echo ...
<?php                   } else  {
?>
    ... code HTML sans echo ...
<?php                               }
?>
```

Écriture similaire
mais plus claire :
-> Evite la 'lourdeur' de l'instruction echo()

Structure de contrôle PHP avec inclusion de fichiers HTML

```
<?php
    if ($bool) {
            require ("ficB");
            }
            else  {
            require("ficR");
            }
?>
```



Séparation propre du contrôle écrit en PHP et du HTML !

Note : les fichiers inclus sont quelconques et peuvent eux-mêmes contenir des incrustations php, inclusions…

Spécifier les fichiers secondaires du html : css ou script pour le navigateur

Basique :
hors balise PHP :

```
<link />
<script
   type='text/JavaScript'
            src='urlScript.js'>
</script >
```

Ces fichiers seront Chargés par le navigateur client.

ou, en dynamique entre balises PHP :

```
<?php   $nomfic=  'urlScript.js' ?>
<link />
< script type='text/JavaScript'  src=  "<?php
                                        echo $nomfic;
                                    ?> "
</script >
```

Lancement des scripts PHP via le navigateur

URL du script à lancer à partir de :

- la barre d'adresse du navigateur
- un lien dans une page HTML
- un formulaire dans une page HTML

Les URL peuvent avoir des paramètres

- intérêt : les valeurs des paramètres sont exploitables dans les scripts

Note : il est aussi possible de lancer un script en local sur le serveur

- spécifier le script en paramètre de lancement de l'interpreteur php.exe

Type de communication entre le navigateur et le serveur web

☞ Communication synchrone,
cas défaut :
- Liens (<a>)
- Formulaire (<form>)

☺ gestion automatique de la requête
☹ requête bloquante
(pas de requête simultanée : attente de rpse …)

☞ Communication asynchrone :
Lancements explicites de requêtes HTTP
(*via du code javascript exécuté sur le client)*

☺ gestion simultanée de plusieurs requêtes
☹ gestion explicite de l'historique
☹ contrôle explicite des transactions (arrêt)

Méthodes de transmission des requêtes du client vers le serveur

GET : un seul paquet
POST : plusieurs paquets
- url dans paquet d'entête

GET : paramètres du script transmis avec l'URL
(après un séparateur '?')
(limite: le paquet d'entête a une taille limité)

☞ GET http://serveur/action.exe?nomChps1=*val1*&nomChps2=*val2*

☞ POST http://serveur/action.exe | données | données

Astuce : même avec POST :
il est possible de rajouter
des paramètres
dans le paquet d'entête

données : - formulaires,
- fichiers (upload)

Exemple de lancement de fichier php à partir des liens HTML

Le clic sur un lien HTML provoque une transmission HTTP suivant la méthode GET (1 paquet).

```
<html><body>
<ul>
<li><a href="main.php?action=ajout&quant=un"> ajouter une fiche</a> </li>
<li><a href="main.php?action=affich&quant=un"> afficher une fiche</a> </li>
<li> <a href="main.php?action=affich&quant=tous" > afficher toutes fiches</a> </li>
</ul>
</body></html>
```

on invoque ici un script main.php,
avec 2 paramètres "action" et "quant"

Exemple de requête GET :: http://www..../**main.php?*action=affich&quant=tous***

Exemple de lancement à partir d'un formulaire

Un formulaire HTML est transmis suivant GET ou POST

```
<form     action="url SCRIPT"
                    method="get ou post">

    …… le formulaire (nomChamps-valeur)  et
                son bouton de soumission… ……
</form>
```

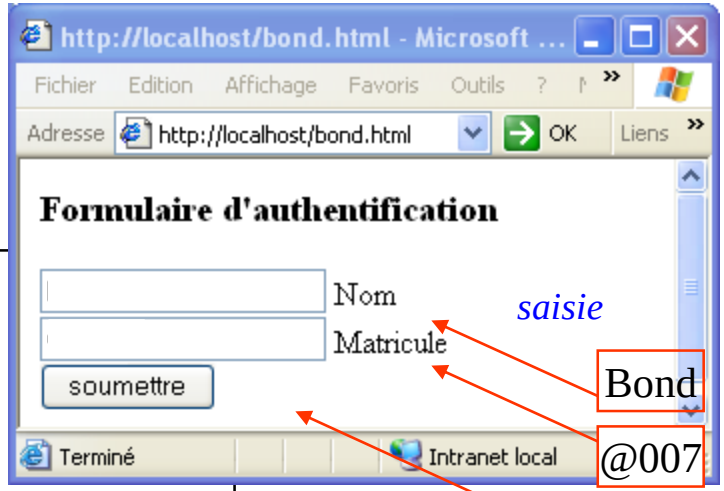
Attributs du formulaire :

action : url du script PHP ou d'un executable (script de type CGI) traitant les données saisies à partir du formulaire.

method : méthode de transfert des données

Exemple de formulaire avec la méthode GET

affichage du formulaire

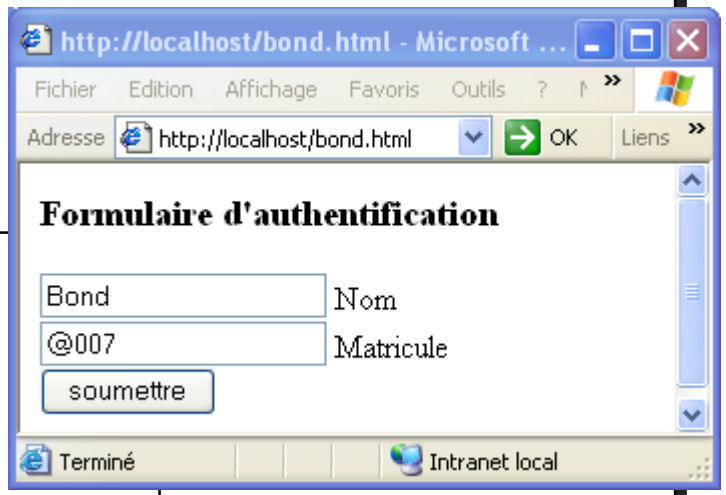



```
<html><body>
<h3> Formulaire d'authentification </h3>
<form action="ident.php" method="get">
   <input name="nom" type="text">     Nom     <br/>
   <input  name="num"    type="text"> Matricule     <br/>
   <input type= "submit"  value="soumettre">
</form></body></html>
```

authent.html

transmission GET :: http://www..../**ident.php?nom=Bond&num=@007**

Exemple de formulaire avec la méthode POST



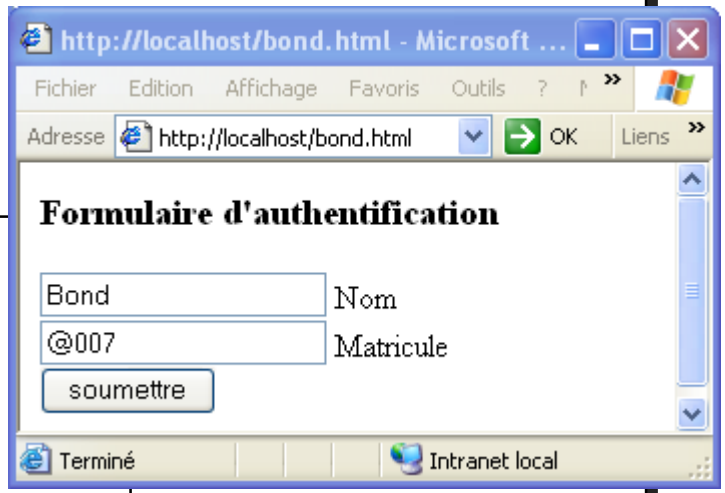

```
<html><body>
<h3> Formulaire d'authentification </h3>
<form action="ident.php" method="post">
   <input name="nom" type="text">     Nom     <br>
   <input  name="num"    type="text"> Matricule     <br>
   <input type= "submit"  value="soumettre">
</form></body></html>
```

authent.html

Transmission POST :: http://www..../**ident.php** ***nom=Bond&num=@007***

Exemple de formulaire paramétré

Cas d'un scripts pouvant rendre plusieurs services dont celui d'authentification
(voir cours sur la conception d'applications WEB)




```
<html><body>
<h3> Formulaire d'authentification </h3>
<form action="index.php?action=ident" method="post">
   <input name="nom" type="text">     Nom     <br>
   <input  name="num"    type="text"> Matricule     <br>
   <input type= "submit"  value="soumettre">
</form></body></html>
```

authent.html

Transmission POST :: http://www..../**index.php?action=ident** ***nom=Bond&num=@007***

GET vs POST

Avantage GET :

- efficace (un unique paquet), défaut
- transmission possible de paramètres

Inconvénient GET:

- données visibles dans la barre d'adresse avec l'url (non confidentielles)
- taille du paquet d'entête fixé (nombre de paramètres possibles limité)

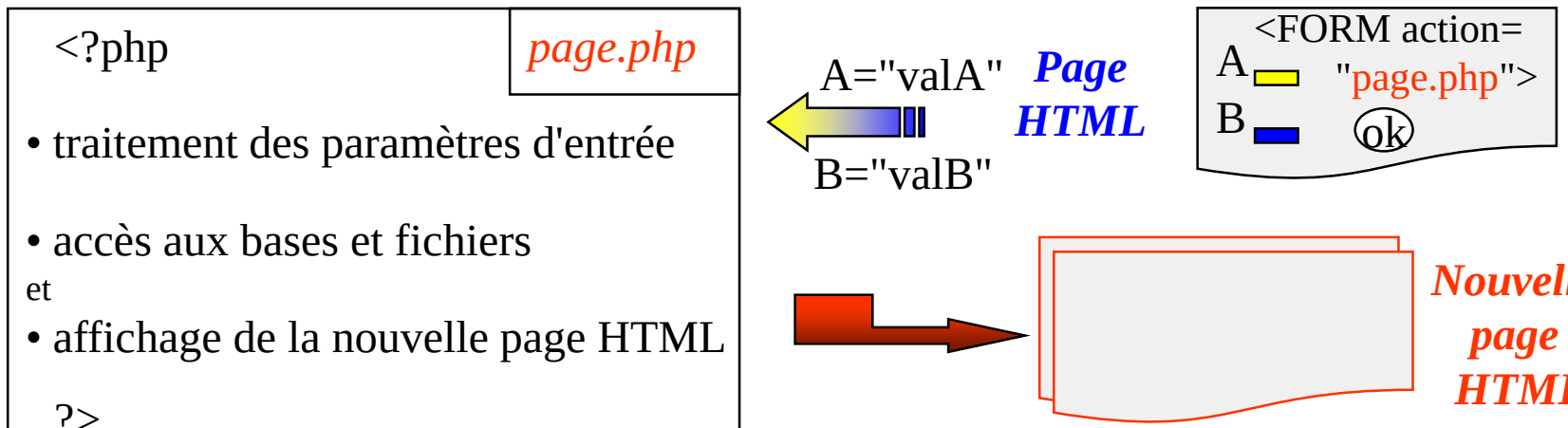
Avantage POST:

- données non visibles dans la barre d'adresse
- pas de limites dans la taille des données transmises

Traitement PHP des données de l'URL

Le script PHP est invoqué avec en entrée
les noms (attribut name) et valeurs des paramètres.
Son rôle possible est de produire une nouvelle page HTML

1ère page web statique ou créée dynamiquement



Traitement PHP des données de l'URL

Le script PHP est invoqué avec en entrée
les noms (attribut name) et valeurs des paramètres.
Son rôle possible est de produire une nouvelle page HTML

page html statique ou créée dynamiquement

Page HTML

<FORM action=
A
"page.php">
B
ok

A="valA"
B="valB"

<?php

page.php

- traitement des paramètres d'entrée
- accès aux bases et fichiers

et

- affichage de la nouvelle page HTML

?>

Nouvelle page HTML

Exploitation dans le script PHP des données d'entrée

pour un paramètre ou une donnée de nom 'nom'

Variables tableaux PHP spécialisées :

- $_GET['*nom*'] tout paramètre de l'URL ou données de formulaire GET
- $_POST['*nom*'] toutes données de formulaire POST
- $_REQUEST['*nom*'] paramètres GET, POST rassemblés en un seul tableau

Nom d'un champs (**name**) ou d'un paramètre de l'URL

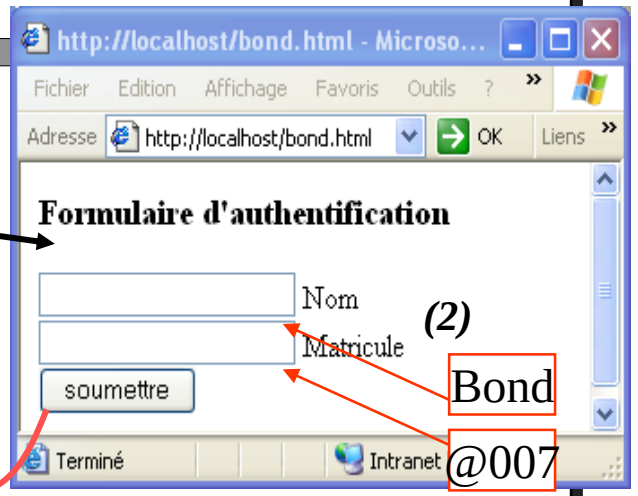
```
<?php
        $v  = $_GET['nomChamps'] ;
        echo ("champs : " . $v . "<br/>");
?>
```

Application au traitement d'un formulaire

Exemple simplifié : renvoi des valeurs saisies dans un formulaire

Form_authent.html

```
<html><body>
<h3> Formulaire d'authentification </h3>
<form action="retourSaisie.php" method="post">
   <input name="nom" type="text" />     Nom     <br />
   <input  name="num"    type="text" />    Matricule  <br />
   <input type= "submit"  value="soumettre">
</form></body></html>
```



(1)

(2)

(3)

retourSaisie.php

```
<html><body>
<h3>   Bienvenue M. <?php echo ($_POST['nom']);  ?>
</h3>
Nous pouvons traiter vos informations : <br/>
<?php    echo ("nom : "   . $_POST['nom'] . "<br/>\n");
         echo ("matricule : "   . $_POST['num'] . "<br/>\n"
?>   </body></html>
```

(4)

Script récursif incluant l'affichage du formulaire et ses traitements

Un tel script distinguera successivement **plusieurs étapes temporelles**, à invoquer séparément **via le navigateur**

Intérêt : l'affichage et les traitements du formulaire sont réalisés par un unique script.

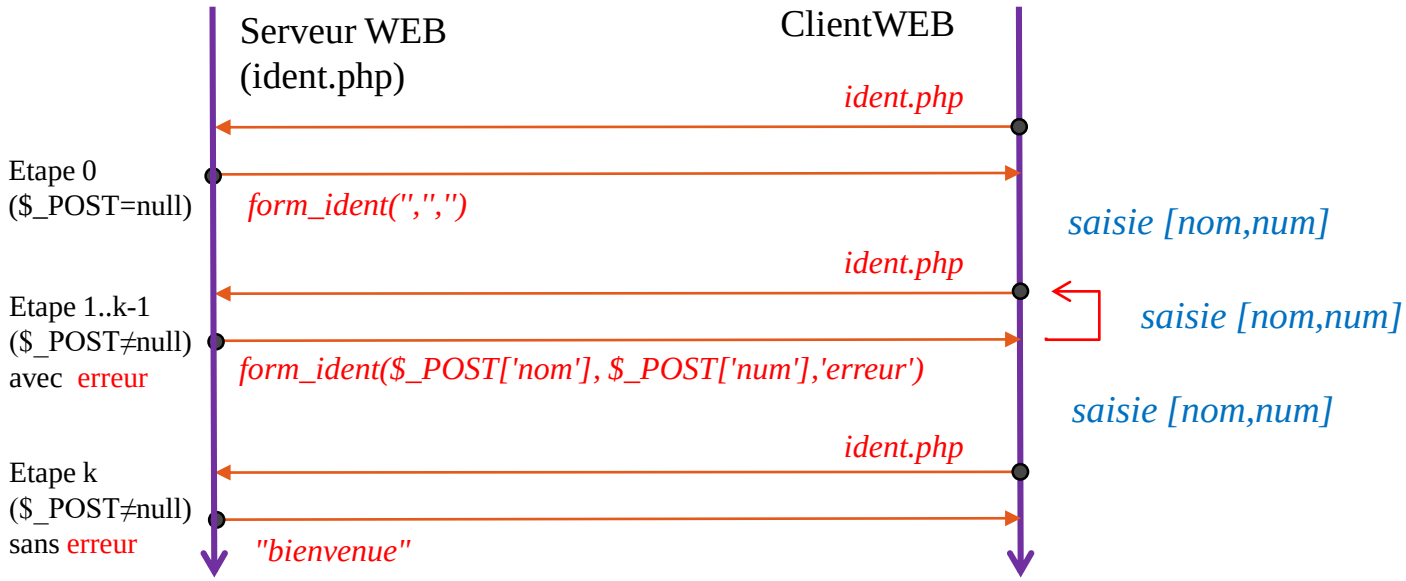
- Etape 1 : **Emettre** le formulaire vierge
- Etape 2 : **Traiter** le formulaire
  - Cas erroné : **Ré-émettre** le formulaire avec ses saisies et une information sur l'erreur.

Différencier les étapes 1 et 2 en testant la présence ou l'absence de données saisies

exple : if ***count($_POST)*** *== 0* *//test global pour un formulaire POST*

autre : if ***isset($_POST['unChamps'])*** *//ou sur un champs du formulaire*

Schéma d'exécution du script récursif



Exemple de script récursif

//ident.php

```
<?php

if  (count($_POST)==0)
     form_ident() ;
     else {
          $nom =$_POST['nom']);
          $num=$_POST['num']);
        if  (! verif_ident($nom,$num)) {
              $msgErr ="erreur de saisie";
              form_ident($nom,$num,$msgErr) ;
         }
        else  { echo ("ok, bienvenue"); }
     }
?>
```

Appels de la fct form_ident

vérification dans la base de données

La fonction form_ident() affiche une page HTML contenant le formulaire et re-affiche les valeurs des champs du formulaire quand ils ont été saisis, en utilisant $_POST.

//suite de ident.php

```
<?php  function form_ident($nom='',$num='',$msg='') {
?>
      <html><body>
       <form action="ident.php" method="post">
           nom : <input name="nom"
                    value="<?php echo $nom; ?>" />
           matricule : <input name= "num"
                    value="<?php echo $num; ?>" />
              <input type= "submit" value= "ok" />
        </form>
         <div id ="m"> <?php echo $msg; ?> </div>
        </body></html>
<?php   }   ?>
```

Exemple alternatif : inclusion de page HTML paramétrée (template)

ident.php

```
<?php
 $nom=  isset($_POST['nom'])?($_POST['nom']):'';
 $num=  isset($_POST['num'])?($_POST['num']):'';
 $msg='';                    Tests compacts pour affectations

if  (count($_POST)==0)                          Inclusion
        require ("ident.tpl") ;                 de fichier
         else {
              if  (! verif_ident($nom,$num)) {
                 $msg ="erreur de saisie";
                 require ("ident.tpl") ;
              }
              else { echo ("ok, bienvenue"); }
         }
?>
```

Précaution : Le fichier template contient des variables PHP (affichage) qui doivent être **renseignées avant** le "require" d'inclusion du template.

ident.tpl

```
<html><body>
<form action= "ident.php" method= "post" >
   nom : <input name="nom"
            value="<?php echo $nom ?>" /> <br/>
   matricule : <input name="num"
            value="<?php echo $num?>" /> <br/>
   <input type= "submit" value= "ok" />
</form>
<div id ="m"> <?php echo $msg; ?> </div>
</body></html>
```

Conception d'application WEB : la notion de services

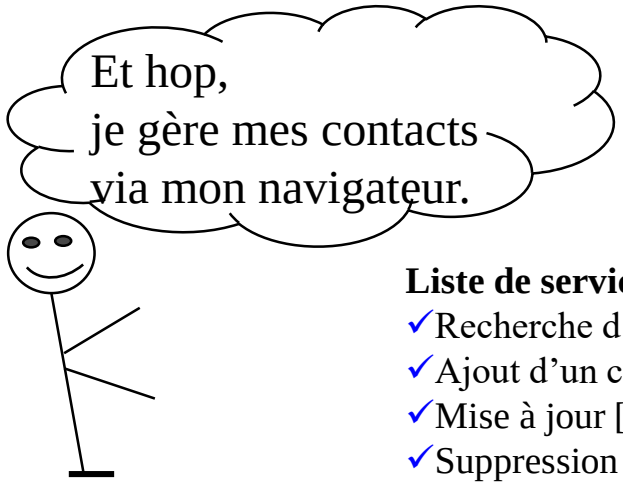
Modèle d'applications WEB orientées services

- Une application WEB est **structurée en services applicatifs codés sous forme de scripts PHP.**

- **L'IHM est définie sous forme de pages WEB HTML** permettant de sélectionner les services, via de liens ou des formulaires
- **Les URL des liens et des boutons de soumission peuvent être paramétrés :**
  - identifiant du service si un script traite plusieurs services
  - valeurs de données à traiter.
- **Les pages HTML de l'IHM sont construites dynamiquement** chacune comme une sortie d'un script PHP.

Exemple d'applications WEB utilisant des liens et formulaires

- Soit, une application de **gestion de contacts**
- On trouve en BD, les informations de chaque contact
- Chaque contact est référencé par un identifiant

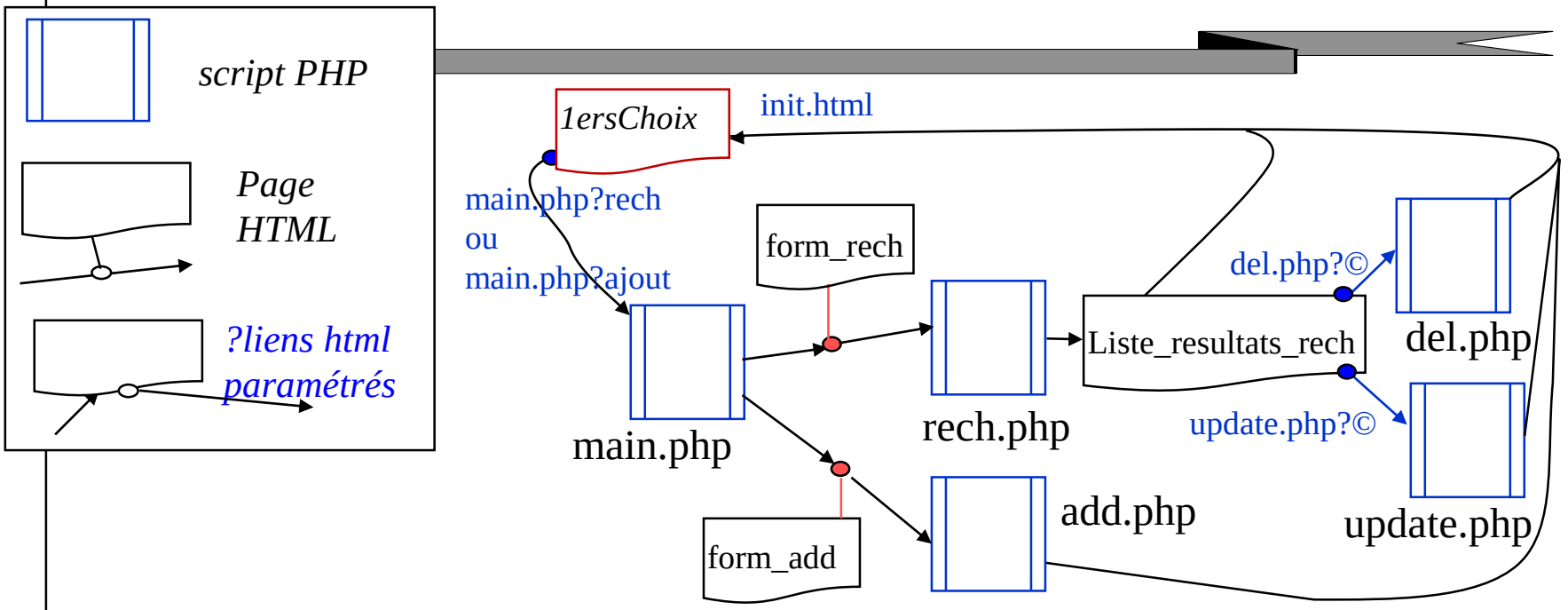


use case

Liste de services de mon application :

- ✓Recherche d'une liste de contacts
- ✓Ajout d'un contact
- ✓Mise à jour [c]
- ✓Suppression [c]

Enchaînement des services et des écrans de l'IHM



• *Ici, init.html est la page principale de l'application : elle contient 2 liens paramétrés pour enclencher via main.php une recherche ou un ajout.*

• *Chaque ligne du résultat d'une recherche correspond aux informations d'un contact, et affiche deux liens paramétrés avec l'identifiant du contact, pour une suppression ou une mise à jour.*