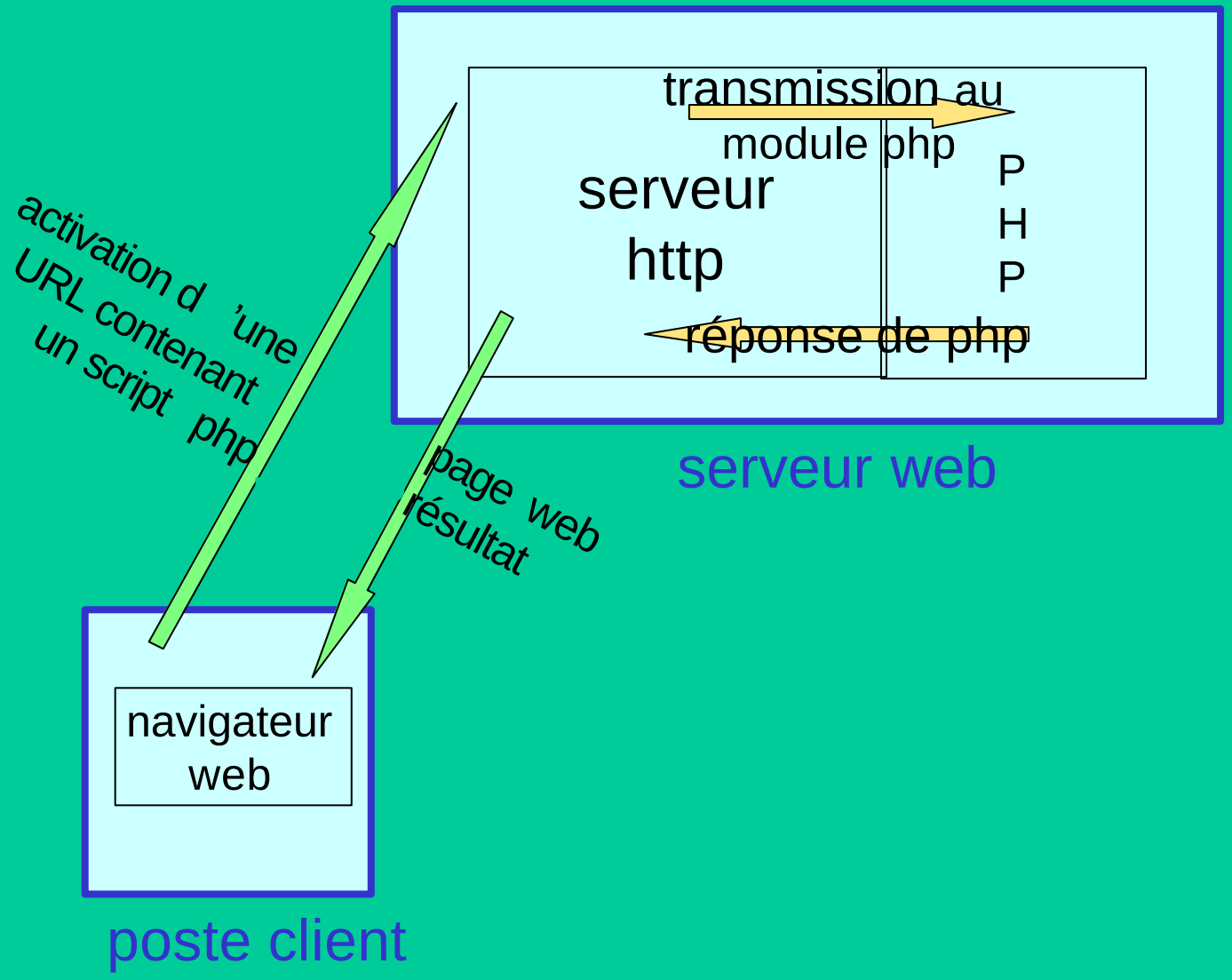
PHP

principe des pages web dynamiques en PHP



généralités

- une page php (Hypertext PreProcessor http://www.php.net) est un fichier d'extension .php contenant du code php entre <? et ?> et éventuellement du code HTML.
- le fichier
  - réside sur le serveur web
  - n'est pas transféré vers le navigateur (sécurité,confidentialité)
  - est interprété côté serveur par un module PHP
- l'exécution du script « écrit » la page qui sera transmise au navigateur via le serveur web
- le module php « tourne » dans l'environnement des serveurs web qui le supportent (IIS, Apache, donc la majorité) ou éventuellement en cgi

premier exemple

accessible à partir de http://www.info.univ-angers.fr/pub/pn/PHP/index.html

```
<HTML>
<HEAD>
<TITLE>page bonjour par php</TITLE>
</HEAD>
<BODY BGCOLOR="white">
<?
print("bonjour <FONT COLOR=\"red\"> VOUS</FONT><BR>\n");
setlocale ("LC_TIME", "fr");
print("chez moi nous sommes le");
print(strftime("%A %d %B %Y et il est %H h %M min %S sec\n"));
?>
<BR>
Ce bonjour vous était donné par php.<BR>
<BR>
<A HREF="index.html">retour</A>
</BODY>
</HTML>
```

fichier bonjour.php

```
<HTML>
<HEAD>
<TITLE>page bonjour par php</TITLE>
</HEAD>

<BODY BGCOLOR="white">

bonjour <FONT COLOR="red"> VOUS</FONT><BR>
chez moi nous sommes le jeudi 20 mars 2003 et il est 13 h 43 min 12 sec
<BR>
Ce bonjour vous était donné par php.<BR>

<BR>
<A HREF="index.html">retour</A>
</BODY>
</HTML>
```

page web reçue par le navigateur

éléments de syntaxe

- ; est un séparateur d'instructions
- { … } regroupe des instructions dans un bloc unique
- // commentaire sur une seule ligne
- /* commentaire
  sur plusieurs
  lignes */
- déclaration de variable : $X = expression
  - la variable est du type de l'expression, mais ce type peut évoluer au fur et à mesure des affectations successives (langage faiblement typé)
  - la casse (majuscule/minuscule) est prise en compte par le langage (X et x sont deux variables différentes), mais pour les fonctions la casse est ignorée
- affectation : $X = expression

types de données

- entier :
  - $N = 5;
- décimal :
  - $X = 7.123; ou $Y=1.45e10;
- texte :
  - $NOM = "toto"; ou $NOM = 'toto';
  - $PHRASE = "bonjour $NOM"; donne "bonjour toto" car la variable incluse dans la chaîne est interprétée (ce n'est pas le cas si la chaîne est délimitée par ' et ')
- pas de type logique spécifique
  - la valeur FALSE est représentée par 0 ou "" (la chaîne vide) et toutes les autres valeurs sont interprétées à TRUE
- pas de type caractère unique
  - un caractère unique est représenté par une chaîne de longueur 1

paramètres de formulaires

- les paramètres d'un formulaire envoyés depuis le navigateur vers le serveur (par post ou get) sont accessibles directement par leur nom dasn le script qui les reçoit
- exemple de l'addition accessible à partir de http://www.info.univ-angers.fr/pub/pn/PHP/index.html

```
Additionneur : entrez un nombre dans chaque case et cliquez sur =.
<FORM ACTION="addition.php" METHOD=POST>
<INPUT TYPE=text NAME="N1" SIZE="10">
+
<INPUT TYPE=text NAME="N2" SIZE="10">
<INPUT TYPE=submit VALUE="=">

</FORM>
```

formulaire html

```
<HTML>
<HEAD>
 <TITLE> résultat addition</TITLE>
</HEAD>
<BODY BGCOLOR="white">
<H1>Résultat de l'addition</H1><BR>
<?
// on calcule le résultat
$R=$N1+$N2;
// on écrit le résultat dans la page web
print("$N1 + $N2 = $R");
?>
<BR>
<A HREF="index.html">retour</A>
</BODY>
</HTML>
```

addition.php

opérateurs

- arithmétiques
  - +, -, *, / (division entières si les 2 opérandes sont entiers), % (modulo), ++ (incrémentation), -- (décrémentation)
- de comparaison
  - == (égalité), != (différence), <, >, <=, >=
- logiques
  - && (et), || (ou), ! (non)
- sur des chaînes de caractères
  - . (concaténation)

structures de contrôle

- conditionnelle

```
if (condition) {
        instructions
}
```

- alternative

```
if (condition) {
        instructions1
}
else {
        instructions2
}
```

- alternative complexe

```
if (condition) {
        instructions1
}
elseif (condition) {
                instructions2
        }
        else {
                instructions3
        }
```

- cas

```
switch (expression) {
        case val1 : instructions1; break;
        …
        case valn : instructionsn; break;
        default : instructions par défaut;
}
```

est équivalent à

```
if (expression==val1) { instructions1 }
else  if (expression==val2) { instructions2 }
          else …
                    ...
                    else { instructions par défaut }
```

- boucle *pour i=valdéb croissant jusqu'à valfin*

```
for ($i=valdéb; $i<=valfin; $i++) {
        instructions
}
```

- boucle pour "générale"

```
for (initialisation; condition; mise à jour) {
        instructions
}
```

la boucle s'arrête quand la condition n'est plus respectée

- boucle tant que

  ```
  while (condition) {
        instructions
  }
  ```

  la boucle s'arrête quand la condition n'est plus vérifiée
- boucle répéter jusqu'à

  ```
  do {
        instructions
  }
  while (condition)
  ```

  la boucle s'arrête quand la condition n'est plus vérifiée (au moins une itération)

fonctions

- function nom(liste de paramètres) {
  corps de la fonction
  return résultat; // éventuellement
  }
- la liste de paramètres formels
  - peut être vide ()
  - ou être de la forme ($p1, ..., $pn)
- *on ne s'intéresse ici qu'au passage par valeurs des arguments*

tableaux

- un tableau est un tableau associatif
  - il stocke des paires (clé,valeur)
  - la clé peut-être un entier ou une chaîne de caractères
  - les clés d'un même tableau peuvent être de type différents, mais elles sont toutes distinctes
  - les valeurs stockées sont quelconques (tableau éventuellement)
- déclaration éventuelle par $T=array();
- affectation de valeurs
  - $Mois[1] = "janvier";
  - $Nbjours["janvier"]=31;
  - $Semaine=array("lundi", "mardi", "mercredi", "jeudi", "vendredi", "samedi", "dimanche");
  - $Nbjours=array("janvier"=>31, "février"=>29);

exemple d'utilisation de tableaux

calcul de nombres premiers à partir de
http://www.info.univ-angers.fr/pub/pn/PHP/index.html

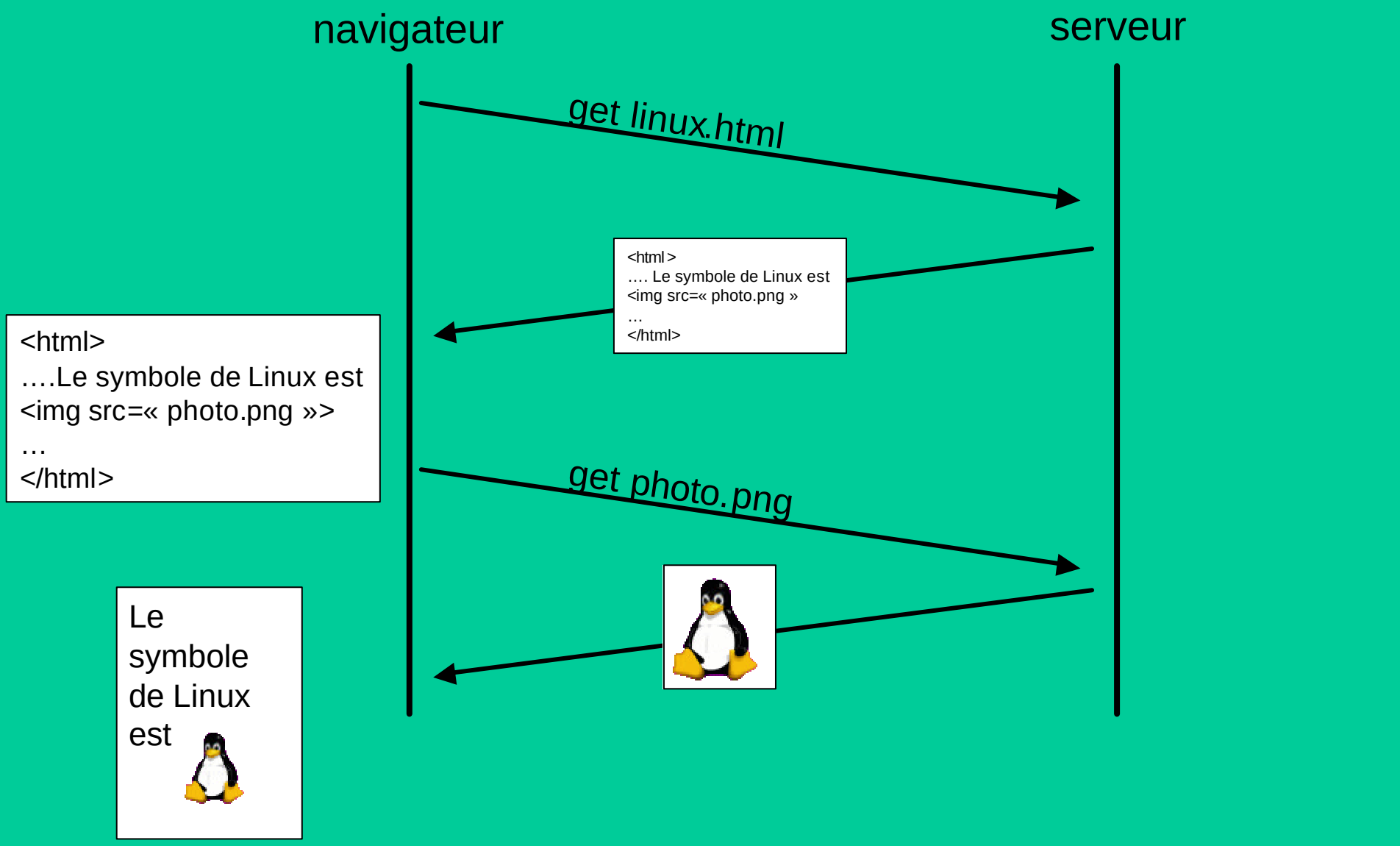
```
<HTML>
<HEAD>
<TITLE>nombres premiers</TITLE>
</HEAD>

<BODY BGCOLOR="white">
<H1>les nombres premiers inférieurs à <? print($lim); ?> sont </H1>
<?
// lim est le champ qui provient du formulaire
// méthode du crible d'Eratosthène
// on fabrique un tableau de booléens initialisé à vrai partout
 for ($i=1; $i<=$lim; $i++) {
     $prem[$i]=1; // tout ce qui n'est pas nul est vrai
 }
```

nbpremiers.php

```
$lim2=floor(sqrt($lim));
for ($k=2; $k<=$lim2; $k++) {
    // on supprime les multiples de k
    $j=2*$k;
    while ($j<=$lim) {
            $prem[$j]=0; // le 0 représente le faux
            $j=$j+$k;
    }
}
// on écrit les nombres premiers dans la page web
for ($i=1; $i<=$lim; $i++) {
    if ($prem[$i]) {
            print("$i  ");
    }
}
?>
<BR><A HREF="index.html">retour</A>
</BODY>
</HTML>
```

nbpremiers.php

rappel sur le chargement d'une page web contenant une image



fabrication d’images en PHP

- un script php peut fabriquer une image qui sera envoyée au navigateur
- les images peuvent être de type GIF, JPEG ou PNG
- le script doit envoyer au navigateur l'en-tête correspondant au type de l'image grâce à la fonction header
- un même script ne peut pas envoyer le texte d'une page html et une image simultanément, l'image construite doit être "envoyée à travers" un marqueur IMG

```
<html>
<head>
<title>drapeau construit par php</title>
</head>
<body>
<h1>drapeau construit par php</h1>
<img src="fabriquedrapeau.php" alt="image construite par php"> <BR>
<A HREF="index.html">retour</A>
</body>
</html>
```

drapeau.html

```php
<?  // avant d'envoyer tout autre chose, on envoie l'entête adéquat
 header("Content-type: image/png");
 // on crée une image
 $im=imagecreate(300,200);
 // on définit les couleurs utilisées
 $bleu = ImageColorAllocate($im, 0, 0, 255);
 $blanc = ImageColorAllocate($im, 255, 255, 255);
 $rouge = ImageColorAllocate($im, 255, 0, 0);
 // on dessine le drapeau avec 3 rectangles
 ImageFilledRectangle ($im, 0, 0, 100, 200, $bleu);
 ImageFilledRectangle ($im, 100, 0, 200, 200, $blanc);
 ImageFilledRectangle ($im, 200, 0, 300, 200, $rouge);
 // on envoie l'image vers le navigateur
 imagePng($im);
 // on efface de la mémoire les données de l'image
 ImageDestroy ($im);
?>
```

fabriquedrapeau.php

fonctions relatives aux images

voir la doc en ligne pour les paramètres des fonctions et les détails

- ImageCreate crée une image
- ImageColorAllocate définit les couleurs
- ImageCreateFromgif (jpeg ou png) pour créer une image à partir d'un fichier existant
- ImageGif (jpeg ou png) envoie l'image au navigateur
- il existe des fonctions pour dessiner un pixel, une ligne, un rectangle, un arc d'ellipse (donc aussi un arc de cercle, ou un cercle, ou une ellipse), un polygone, une chaîne de caractères (horizontalement ou verticalement)

cookies

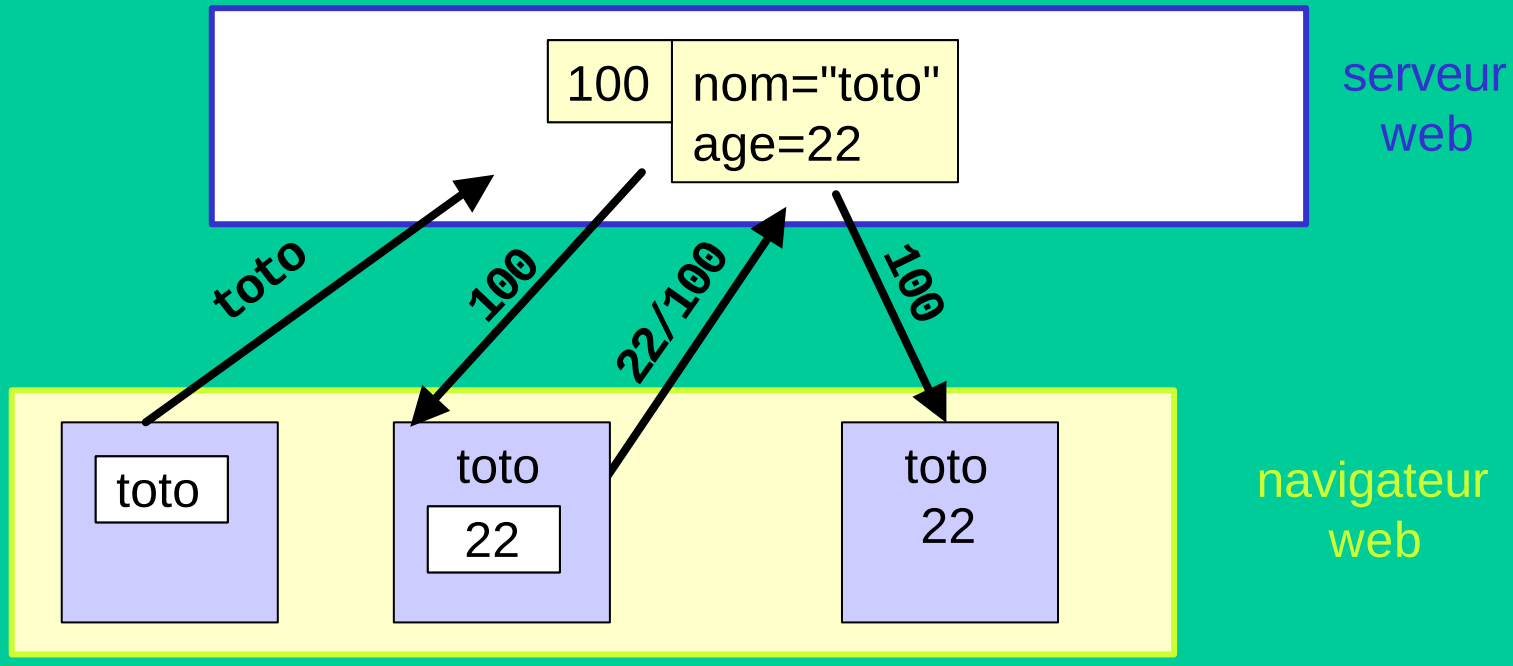
- Les cookies sont des couples (nom,valeur) échangés entre le serveur web et le navigateur. Ils sont stockés dans un fichier texte sur le poste client et servent à mémoriser des informations entre les différents passages d'une personne sur un même site (voir http://www.commentcamarche.net/securite/cookies.php3)
- Si le navigateur accède à un document du répertoire R sur un site W et qu'il possède des cookies valides pour R sur W, alors il envoie tous ces cookies au serveur.
- La fonction setcookie permet d'envoyer un cookie depuis le serveur vers le navigateur.
  - setcookie("nom", "toto") envoie le cookie "nom" avec "toto" comme valeur depuis le serveur vers le navigateur, par la suite $nom contiendra la valeur du cookie nom
  - d'autres paramètres servent à fixer la portée du cookie, sa durée de vie, …
  - à faire avant tout envoi d'autres caractères vers le navigateur (avant <html> notamment)

```
<? // cookie pour rappeler la date de la visite précédente du client
// on récupère la valeur du cookie
$valcook=$datedepassage;
// on met à jour le cookie chez le client
setcookie("datedepassage", date('d/m/Y \à H\h i\m\i\n s\s\e\c'),  time()+20*24*3600);
?>
<html><head><title>démo de cookie</title></head>
<body bgcolor="white">
<H1> Aujourd'hui nous sommes le
<?  print(date('d/m/Y \à H\h i\m\i\n s\s\e\c')."<BR>et ");
    if (!$valcook) {// le cookie n'était pas encore défini
            print ("c'est votre première visite chez nous"); }
    else {
            print ("vous êtes déja venu chez nous le : $valcook"); }
?>
</H1><BR><A HREF="index.html">retour</A>
</body>
</html>
```

essaicookie.php

sessions

- Une session permet de mémoriser dans un fichier sur le serveur web des informations relatives à un utilisateur parcourant les pages d'un site. Seul, un identifiant de session correspondant au nom du fichier transite (via un cookie ou une réécriture d'URL) entre le serveur et le navigateur. Cet identifiant permet au serveur de retrouver les données associées à chaque utilisateur du site.



- session_start() permet de créer une nouvelle session ou de récupérer les données associées à cette session si elle existe déjà
- session_register("nom") enregistre la variable nom dans la session
- session_unregister("nom") supprime la variable nom de la session
- session_is_registered("nom") retourne vrai si la variable nom est enregistrée dans la session, et faux sinon
- session_destroy() détruit toutes les données de la session
- $nom retourne la valeur de la variable nom lorsqu'elle a été enregistrée dans la session

```
<HTML>
<HEAD>
 <TITLE>Page de départ pour session PHP</TITLE>
</HEAD>

<BODY BGCOLOR="white">
<H3>Page de départ pour session PHP</H3><BR>
<H3>Donnez votre nom</H3>
<FORM ACTION="PHPsession1.php" METHOD=POST>
<INPUT TYPE=text NAME="nom" SIZE="20">
<INPUT TYPE=submit VALUE="OK">
</FORM>
<A HREF="index.html">retour</A>
</BODY>
</HTML>
```

debutsessionPHP.html

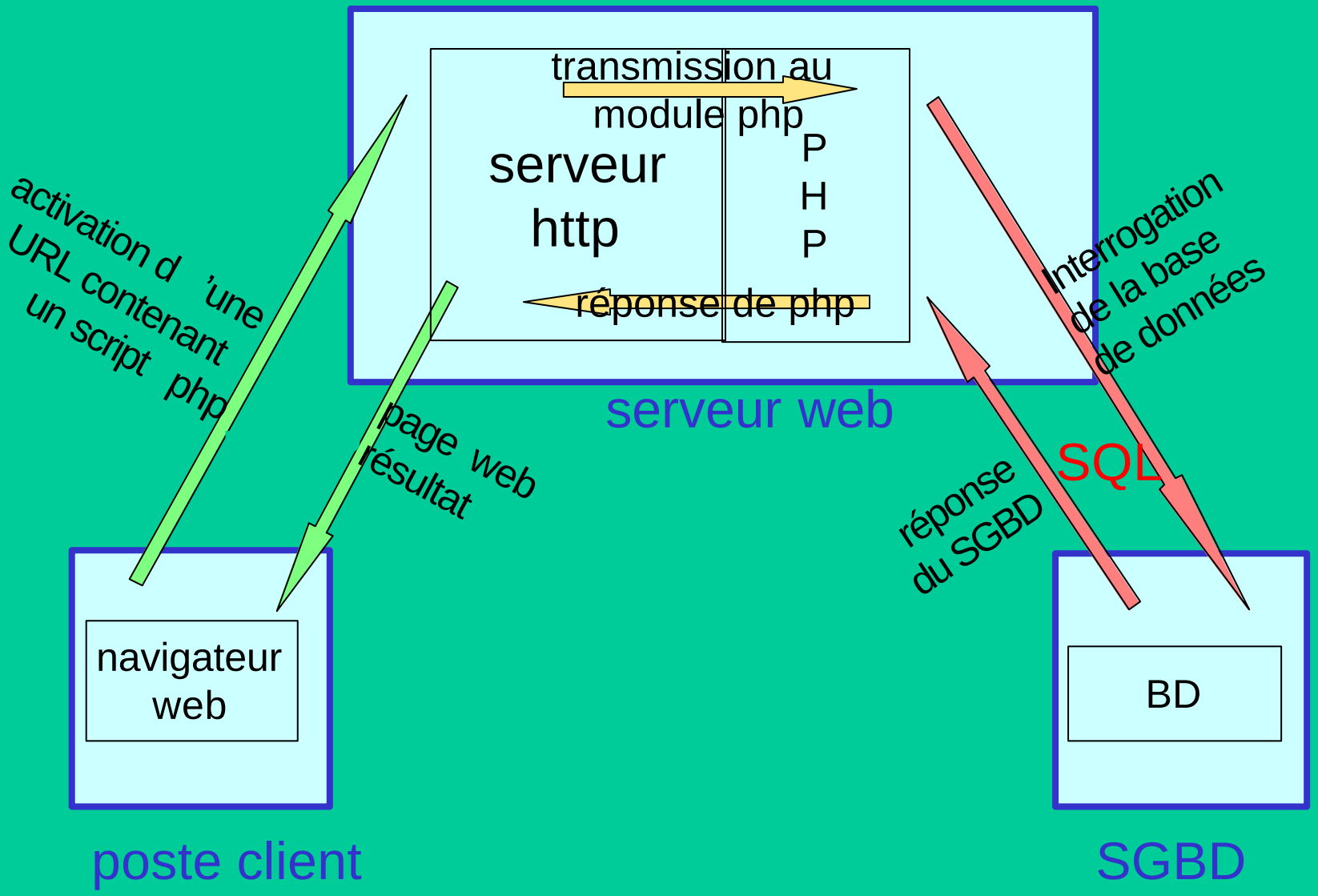
```
<?
// en tout premier on crée une session
$debsession = session_start();
?>
<HTML>
<HEAD><TITLE>Deuxième page pour session PHP</TITLE></HEAD>
<BODY BGCOLOR="white">
<H3>Deuxième page pour session PHP</H3><BR>
<?
// on stocke le nom à mémoriser dans une variable
$nompersonne=$nom;
// on enregistre la variable dans la session
session_register("nompersonne");
?>
<H3><? echo $nom ?> Donnez votre age</H3>
<FORM ACTION="PHPsession2.php" METHOD=POST>
<INPUT TYPE=text NAME="age" SIZE="3">
<INPUT TYPE=submit VALUE="OK">
</FORM>
</BODY>
</HTML>
```

PHPsession1.php

```
<?
//on va rechercher la session
$debsession=session_start();
?>
<HTML>
<HEAD>
<TITLE>Troisième page pour session PHP</TITLE>
</HEAD><BODY BGCOLOR="white">
<H3>Troisième page pour session PHP<H3></BR>
<H3>votre nom est :
<?
// on va rechercher le nom dans la session
echo $nompersonne
?>
<H3><BR>
<H3>votre age est : <? echo $age ?><H3>
<BR><BR>
<A HREF="index.html">retour</A>
</BODY>
</HTML>
```

PHPsession2.php

Web+BD par script PHP



fonctions relatives à l'accès à une base de données

- de nombreuses bases de données sont accessibles via PHP, ici c'est postgresql qui est utilisé
- $c = pg_Connect("host=sirius dbname=etudinfo
  user=etudinfo password=reso99")
  pour se connecter sur sirius à la base etudinfo en tant qu'utilisateur etudinfo dont le mot de passe est reso99, $c contient l'identifiant de la connexion et contient false si la connexion a échoué
- pg_Close($c) ferme la connexion $c
- $result = pg_Exec($c, $r)
  exécute la requête SQL $r sur la connexion $c et renvoie un résultat
- $result est false s'il y a un problème d'exécution de la requête

- si $r est une requête de mise à jour de la base (insert, update, delete) alors pg_cmdtuples ($result) retourne le nombre de lignes modifiées
- si $r est une requête d'interrogation (select)
  - pg_NumRows($result) retourne le nombre de lignes du résultat
  - pg_numfields ($result) retourne le nombre de champs du résultat
  - pg_result ($result, $i, $j) retourne la valeur du j^{ème} champ de la i^{ème} ligne du résultat ($j peut aussi être indiqué avec le nom de la colonne de la table)

```
<HTML>
<HEAD>
 <TITLE>vue sur une table de postgresql</TITLE>
</HEAD>
<BODY>
<H2> affichage des personnages triés par ordre d'age </H2>
<center>
<table width=70% border=5>
<tr><td> NOM </td><td> AGE </td></tr>
 <!-- début du script -->
<?
/* connexion à la base */
    $c = pg_Connect("host=sirius user=etudinfo password=reso99
     dbname=etudinfo");
```

perso_age_postgres.php

```
/* définition et exécution de la requête */
    $req = "SELECT * FROM   personnage order by age";
    $result = pg_exec($c, $req);
/* parcours du résultat */
    $number = pg_numRows($result); /* le nombre de lignes du résultat */
    $i = 0;
    while ($i < $number) {
      $nom=pg_result($result, $i, "nom");
      $age=pg_result($result, $i, "age");
      print "<tr><td>$nom</td><td>$age</td></tr>\n";
      $i++;
    }
  pg_close($c);
?>
<!-- fin du script -->
<!-- la fin de la table et de la page -->
</table></center>
</BODY>
</HTML>
```

perso_age_postgres.php

Formulaire de saisie des données déclenchant l'insertion

```
<FORM ACTION="insertperso_postgres.php" METHOD=POST>
     nom <INPUT TYPE="text" NAME="nomperso" SIZE="20">
     age  <INPUT TYPE="text" NAME="ageperso" SIZE="4">
     <INPUT TYPE=submit VALUE="ajout">
</FORM>
```

```
<HTML>
<HEAD>
 <TITLE>insertion dans une table de postgresql</TITLE>
</HEAD>

<BODY BGCOLOR="white">
<?
// le symbole @ désactive les messages d'erreurs éventuels
// connexion à la base
   $c = @pg_Connect("host=sirius user=etudinfo password=reso99
      dbname=etudinfo");
//
```

Insertperso_postgres.php

```
//définition et exécution de la requête
   $req = "INSERT INTO personnage VALUES ('$nomperso', $ageperso)";
   $result = @pg_Exec($c, $req);
// analyse du résultat
   if (!$result) {          // il y a eu un problème
      print ("l'insertion de ($nomperso, $ageperso) ne s'est pas faite à cause d'une
      erreur");
   }
   else { // tout va bien
      print ("l'insertion de ($nomperso, $ageperso) s'est bien passée");
      print("<FORM ACTION=\"perso_age_postgres.php\" METHOD=POST>\n");
      print("<INPUT TYPE=submit VALUE=\"contenu de la table\">\n");
      print("</FORM>\n");
   }
  pg_close($c);
?>
<BR><A HREF="index.html">retour</A>
</BODY></HTML>
```

Insertperso_postgres.php