

ARCHITECTURE ET CONCEPTION WEB & IHM

A faint, stylized cartoon elephant in the background. The elephant is light purple with large ears, a trunk, and small tusks. It has a simple, friendly expression with large eyes and a small smile. The elephant is positioned behind the main title text.

```
<?php echo "1. Introduction à PHP"; ?>
```

Déroulement du module

Détail des séances

Contenu

1. Le langage PHP
2. Communication avec un serveur
3. Programmation objet
4. BDD - persistance des données
5. Framework PHP : Laravel



La doc

Une seule référence à retenir :

<http://php.net>

Et Stackoverflow (et autres forums) ?

- ✓ C'est pratique... si vous êtes critique.
- ✓ Une réponse peut dater et ne plus correspondre à l'API actuelle.
- ✓ Toutes les bonnes réponses sont dans la doc officielle, et celle de PHP est très complète.
- ✓ Une bonne partie des "solutions" cite la doc : gagnez du temps !



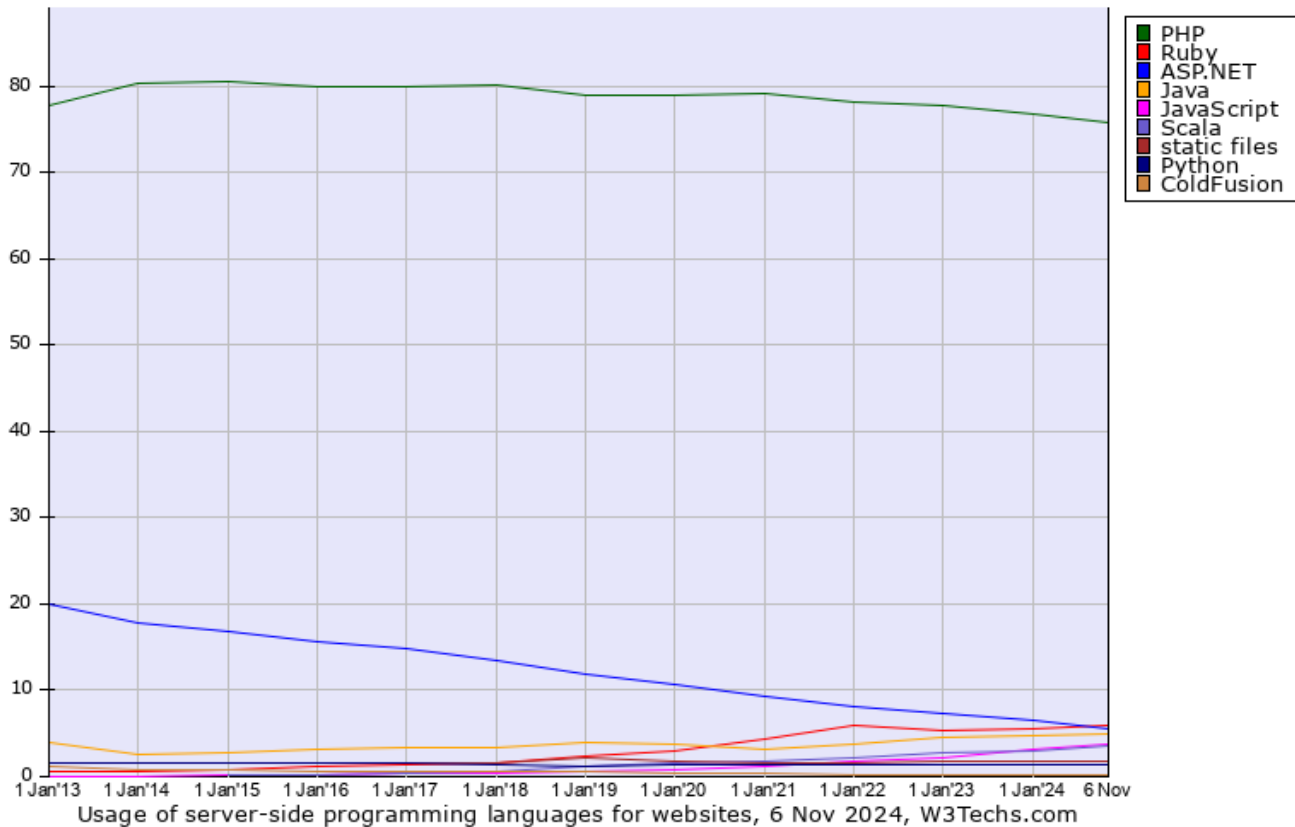
>>



L'environnement PHP

PHP, pourquoi ?

PHP, c'est le langage de programmation utilisé par plus de 75% des sites web du monde.



Javascript, c'est seulement $\approx 3,5\%$

PHP, le langage

```
<!DOCTYPE html>
<html>
  <head>...</head>
  <body>
    <p><?php echo "Hello !"; ?></p>
  </body>
</html>
```

Acronyme récursif pour “PHP: Hypertext Preprocessor”

PHP est un langage de programmation :

- ✓ principalement utilisé côté serveur
- ✓ qui peut être inséré dans du HTML
- ✓ open source
- ✓ interprété (*En fait pas seulement, mais on va faire comme si ;))*)



PHP, le programme

```
<!DOCTYPE html>
<html>
  <head>...</head>
  <body>
    <p><?php echo "Hello !"; ?></p>
  </body>
</html>
```

Le fichier `page.php`

```
$ php -f page.php
<!DOCTYPE html>
<html>
  <head>...</head>
  <body>
    <p>Hello !</p>
  </body>
</html>
```

Interprétation de `page.php` *avec* `php`

`php` c'est aussi un programme :

- ✓ nommé PHP CLI (*Command Line Interpreter*)
- ✓ utilisable en ligne de commande
- ✓ qui interprète les blocs PHP présents dans un fichier
- ✓ qui affiche sur la sortie standard le résultat du fichier interprété

Syntaxe & environnement

Les variables

```
// Commentaire sur une ligne
```

```
/* Commentaire sur  
 * plusieurs lignes */
```

```
// Déclaration et affectation de variables  
$a;  
$a = 8;  
$b = 10;
```

```
// Définition d'une constante  
const NOM_MODULE = 'W31';
```

```
// Chaînes de caractères  
$message = 'calcul sur ' . $a . ' et ' . $b;
```

```
// Opérateurs arithmétiques  
$res = 8 + 10;  
$res *= 10 - 8;
```

```
// Test d'existence d'une variable  
isset($res);
```

```
// Test sur le type d'une variable  
is_int(8); // true  
is_string(8); // false
```

```
/* Obtenir le type d'une variable  
 sous forme de chaîne de caractères */  
gettype(8); // 'integer'  
gettype("plop"); // 'string'
```

Attention au typage dynamique !

```
$a = 10; // a est un entier  
echo gettype($a); // 'integer'  
$a = "dix"; // a est maintenant une chaîne de caractères  
echo gettype($a); // 'string'
```

Les tableaux

En PHP, tous les tableaux sont des tableaux associatifs, i.e. des dictionnaires.

```
// 1. Déclaration avec des clés implicites
$tab = [1, 'deux', 3]; // Avant : array(1, 'deux', 3);
echo $tab[1]; // 'deux'
```

```
// 2. Déclaration avec clés explicites
$tab = [ 0 => 1, 1 => 'deux', 2 => 3 ];
echo $tab[2]; // 3
```

```
// Taille d'un tableau
echo count($tab); // 3
```

```
// Afficher un tableau
print_r($tab);
/* Array
 * (
 *     [0] => 1
 *     [1] => 2
 *     [2] => 3
 * ) */
```

```
/* - Une clé peut être un entier ou une
 *   chaîne de caractères
 * - Une valeur peut être de n'importe
 *   quel type
 */
$notes = [
    'Jean'   => 3,
    'Simon'  => 12.3,
    'Alex'   => 'Absent',
    12       => ['Paul', 'Robert']
];
echo $notes['Jean']; // 3
```

```
// Ajout d'un élément à un tableau
$notes['Étienne'] = 8;
$notes[] = 9; // Ajout (13 => 9) à $notes
```

```
// Suppression d'un élément
unset($notes[12]);
```

Les structures de contrôle

```
// Conditionnelle
if ($load < 100) {
    echo 'Chargement en cours...';
}
// La directive 'else { ... }' est facultative
else {
    echo 'Chargement terminé !';
}
```

```
// Boucle for
for ($load = 1; $load < 100; $load++) {
    echo 'Chargement : '.$load.'/100';
}
```

```
// Boucle while
$load = 1;
while ($load < 100) {
    echo 'Chargement : '.$load.'/100';
    $load++;
}
```

```
// Itération sur un tableau
$tab = ["W31", "A31", "PPP"];

// Si on ne veut traiter que les valeurs
foreach( $tab as $module ) {
    echo "J'aime ".$module."\n";
}

/* Résultat :
*   J'aime W31
*   J'aime A31
*   J'aime PPP */

// Si on veut traiter les couples
// (clé => valeur)
foreach( $tab as $i => $module ) {
    echo $i . ") J'aime ".$module."\n";
}

/* Résultat :
*   0) J'aime W31
*   1) J'aime A31
*   2) J'aime PPP */
```

PHP et HTML

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Ma première page PHP</title>
  </head>
  <body>
    <p>Hello !</p>
  </body>
</html>
```

fullHTML.php

```
<?php echo '<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Ma première page PHP</title>
  </head>
  <body>
    <p>Hello !</p>
  </body>
</html>';
```

fullPHP.php

Un fichier d'extension `.php` peut contenir :

- ✓ uniquement du HTML
- ✓ uniquement du PHP
- ✓ un mélange de PHP et de HTML

Coexistence PHP/HTML

```
<!DOCTYPE html>
<html>
  <head>
    <title>HTML produit par PHP</title>
  </head>
  <body>
    <?php
      $num = rand(1,10);
      if ( $num < 5 ) {
        echo '<p>Inférieur à cinq</p>';
      } else {
        echo '<p>Supérieur à cinq</p>';
      }
    ?>
  </body>
</html>
```

page.php

```
<!DOCTYPE html>
<html>
  <head>
    <title>HTML contrôlé par PHP</title>
  </head>
  <body>
    <?php
      $num = rand(1,10);
      if ( $num < 5 ) {
        ?>
        <p>Inférieur à cinq</p>
        <?php } else { ?>
        <p>Supérieur à cinq</p>
        <?php } ?>
      }
    </body>
  </html>
```

memePage.php

Le PHP peut être alterné avec du HTML :

- ✓ cela permet de dissocier le contrôle du contenu
- ✓ mais attention à la cohérence du code disséminé !

Inclusion de fichiers

```
<!DOCTYPE html>
<html>
  <?php require('head.php'); ?>
  <body>
    <?php require('header.php'); ?>
    <main>...</main>
    <?php require('footer.php'); ?>
  </body>
</html>
```

Les instructions `include` et `require` permettent d'insérer un fichier PHP au sein d'un autre fichier. Cela évite la redondance de code.

Si le fichier inclus/requis n'existe pas :

- ✓ `require` stoppe l'exécution du script avec une erreur
- ✓ `include` continue l'exécution et affiche un avertissement

Variantes : `include_once` et `require_once` n'insèrent un fichier que si celui-ci n'a pas déjà été inclus.

PHP sur un serveur

Quelques pseudo-définitions

Client

Envoie une requête et attend une réponse.

Serveur

Attend une requête, la traite et envoie une réponse.

Protocole HTTP

Ensemble de règles à respecter pour transmettre des informations entre un client et un serveur. Ce protocole dispose de plusieurs méthodes dont GET et POST.

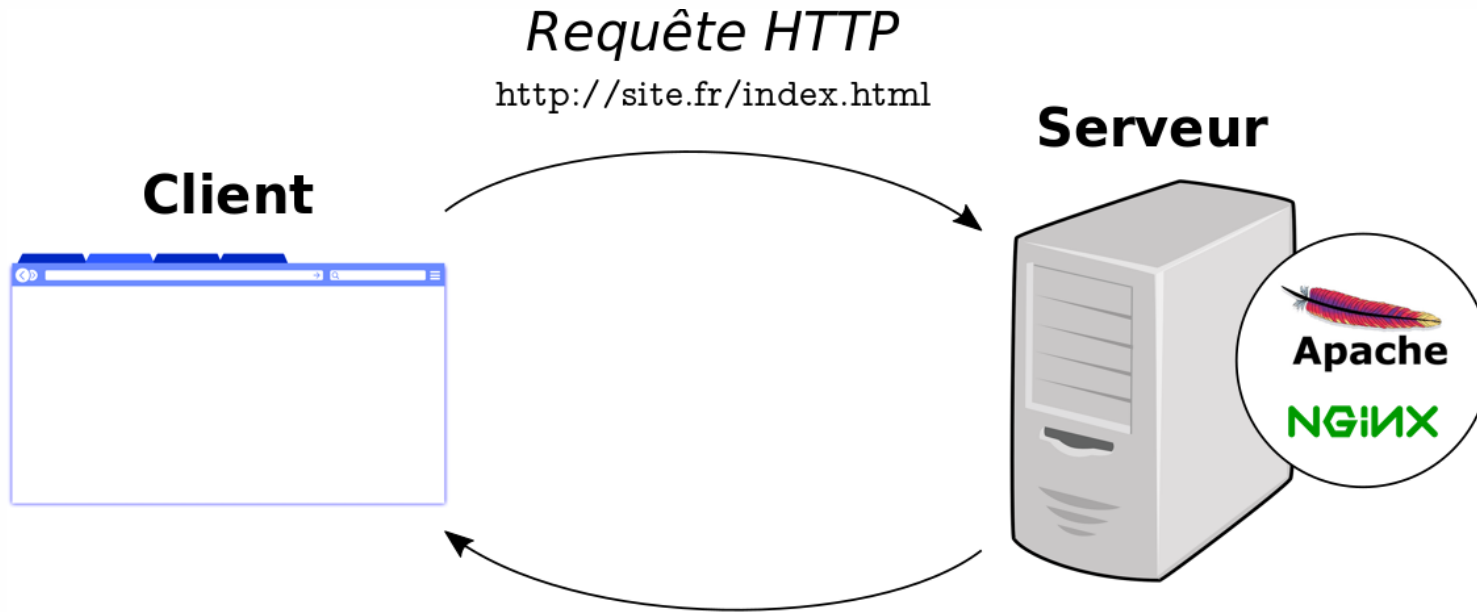
Requête/réponse HTTP

Messages entre client et serveur via le protocole HTTP.

Serveur HTTP

Logiciel qui traite les requêtes HTTP reçues par le serveur web.

Interactions client-serveur



Configuration du serveur HTTP

NGINX



Apache

C'est la cc
nécessai

```
server {  
    listen 80;  
    server_name site.fr;  
  
    # Path to the root of your installation  
    root /var/www/site.fr/;  
    index index.php;  
  
    location ~ /\.php$ {  
        ...  
        fastcgi_pass module-php;  
        ...  
    }  
}
```

conditions
ule PHP.

ulier

Exemple de fichier de configuration de Nginx.

Transmettre des données

Mécanismes de transmission

PHP exploite deux mécanismes pour échanger des données avec le client :

L'URL

- ✓ le client concatène les données à la fin de l'URL
- ✓ le module PHP analyse l'URL et stocke les données dans le tableau global `$_GET`

Le formulaire HTML

- ✓ le client insère les données dans le corps de la requête HTTP
- ✓ le module PHP analyse les données de la requête et stocke les données dans le tableau global `$_POST`

Transmission via l'URL (GET)

Si le client envoie l'URL suivante comme requête au serveur :

```
http://site.fr/index.php?module=CAWEBI&nbEns=2
```

et que le script `index.php` est le suivant :

```
Vous suivez le module <? echo $_GET['module']; ?>  
<br>  
Il est composé de <? echo $_GET['nbEns']; ?>.
```

alors le client recevra le contenu suivant :

```
Vous suivez le module CAWEBI.<br>Il est composé de de 2 intervenants.
```

qui sera interprété comme du HTML si le client est un navigateur.

Transmission via un formulaire (POST)

Si le client envoie le formulaire suivant au serveur :

```
<form method="post" action="index.php">
  <input type="text" name="nom">
  <input type="number" name="age">
  <input type="submit">
</form>
```

et que le script `index.php` est le suivant :

```
Bonjour <?php echo $_POST['nom']; ?>
<br>
Avez-vous réellement <?php echo $_POST['age']; ?> ans ?
```

alors le client recevra un contenu semblable à :

```
Bonjour Bilbo.<br>Avez-vous réellement 142 ans ?
```

où le nom et l'âge dépendent des valeurs spécifiées dans le formulaire.

CONFIANCE ZERO, VÉRIFIEZ !

Règle d'or : Ne jamais faire confiance aux données transmises par le client, toujours les vérifier.

Via une URL

Un client peut écrire les paramètres qu'il veut dans l'URL, quelle que soit la valeur que vous attendez. Exemple :

```
http://site.fr/index.php?module=CAWEBI&nbEns=90  
http://site.fr/index.php?module=10935&nbEns=<script>alert('coucou');</script>
```

Via un formulaire

Un client peut :

- ✓ écrire ce qu'il veut dans une balise `input`.
- ✓ écrire son propre formulaire avec pour `action` votre script.

Comment vérifier et protéger ?

✓ Vérifier l'existence

```
if ( isset($_GET['param1']) && isset($_GET['param2']) ) { ... }
```

✓ Forcer le type (trans)typage

```
$val = (int) $_GET['param1'];
```

✓ Vérifier la valeur

```
if ($val < 10) { ... }
```

✓ Convertir les caractères spéciaux

Permet d'éviter les injections de code (faille XSS).

```
$val = htmlspecialchars($_GET['param1']);
```

Les cookies

Présentation

Définition

Un cookie est une information textuelle indexée par une clé.



Propriétés

Un cookie :

- ✓ est créé par le serveur (PHP)
- ✓ est stocké chez le client
- ✓ transite via le protocole HTTP
- ✓ a une durée de vie spécifiée à la création
- ✓ est automatiquement détruit par le client à expiration

Créer un cookie

1. Un client fait une requête HTTP
2. Le fichier ciblé contient un appel à la fonction PHP `setcookie()` :

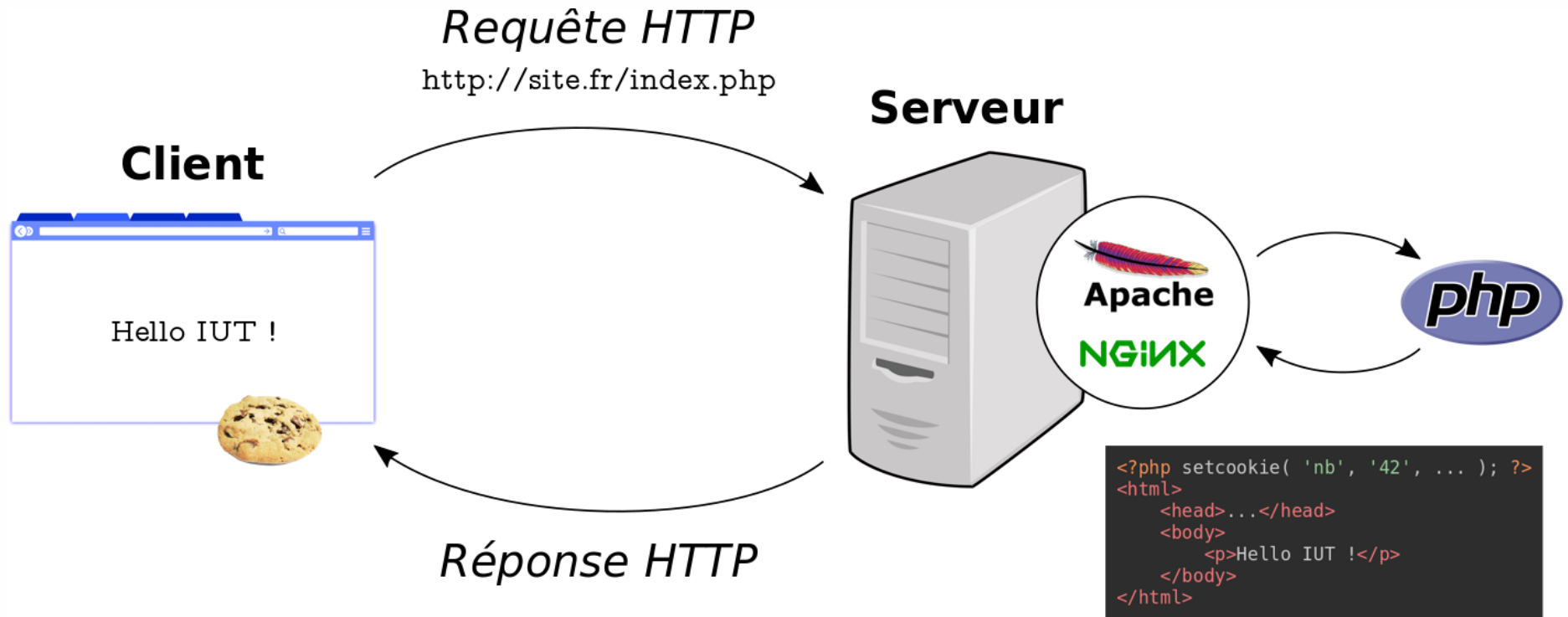
```
setcookie( 'nom', 'valeur', time() + 60*60*24*10);
```

3. Le cookie `'nom' => 'valeur'` est créé lorsque le serveur PHP exécute cette fonction.
4. La demande de création est transmise au client dans l'entête de la réponse HTTP :

```
Set-Cookie: nom=valeur; expires=...; path=/; domain=site.fr
```

5. Le client stocke le cookie dans un fichier.
... et le serveur n'en garde aucune trace.

Schéma de la création d'un cookie



Lire un cookie existant

1. Le client possède un cookie.
2. Il fait une requête HTTP vers le même serveur.
3. Le cookie de ce serveur est ajouté à l'entête de la requête sous la forme :

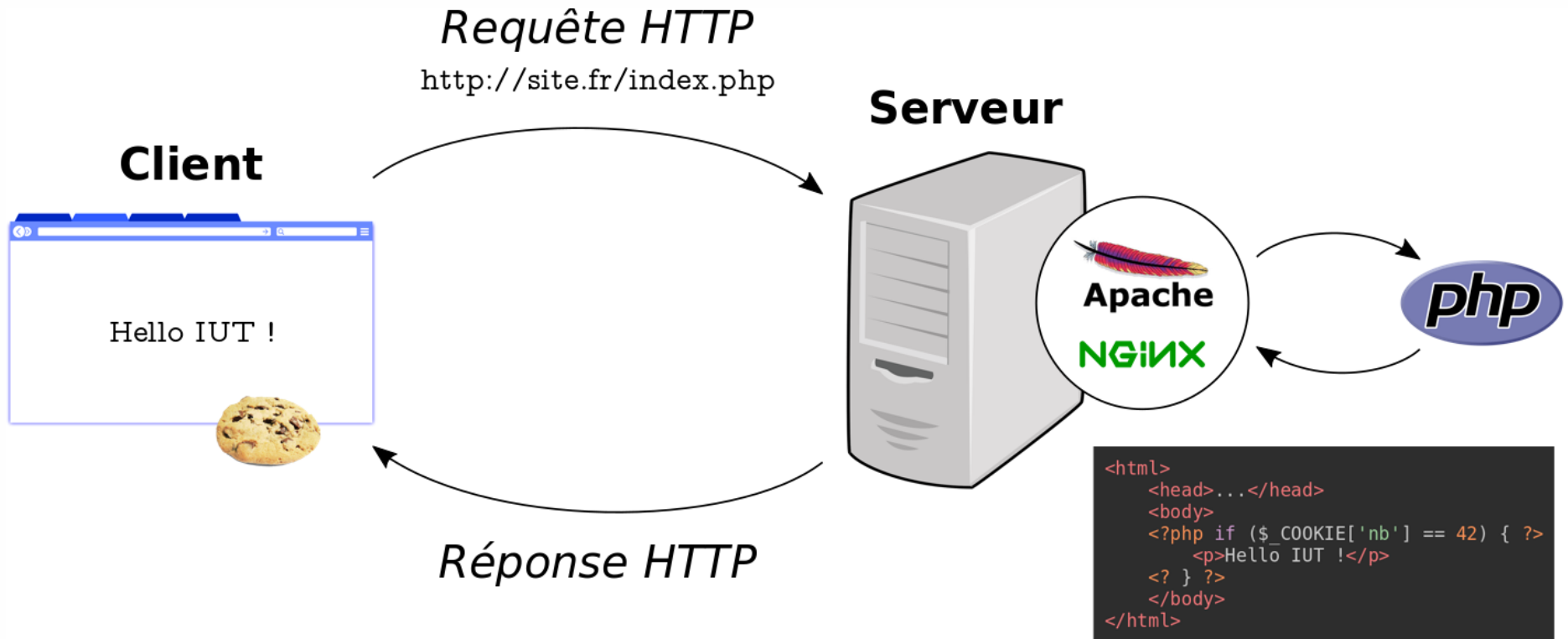
```
Cookie: nom=valeur
```

4. Le serveur PHP peut accéder au tableau global `$_COOKIE` pour exploiter l'information du cookie.

Un serveur peut stocker plusieurs cookies pour un même site. Lors d'une requête vers ce site, ils sont tous ajoutés à l'entête.

```
Cookie: nom=valeur; autrenom=autre valeur ; ...
```

Schéma de la lecture d'un cookie



Supprimer un cookie

C'est le même mécanisme que pour la création :

1. Un client fait une requête HTTP
2. Le fichier ciblé contient un appel à la fonction PHP `setcookie()` :

```
setcookie( 'nom', '', 0); // Expiration le 1er janvier 1970 à 0h00
```

3. Le cookie est créé lorsque le serveur PHP exécute cette fonction.
4. Il est transmis au client dans l'entête de la réponse HTTP :

```
Set-Cookie: nom=valeur; expires=...; path=/; domain=site.fr
```

5. Le client stocke le cookie dans un fichier.
- ... et sa date d'expiration déclenchera le mécanisme de suppression par le navigateur.

Cookies, les 3 règles d'or

Règle n°1

L'instruction `setcookie()` doit TOUJOURS être appelée AVANT tout texte du corps de la réponse HTTP.

Règle n°2

Un cookie est stocké chez le client, il peut donc le lire et le modifier
→ TOUJOURS sécuriser le contenu d'un cookie.

Règle n°3

Un client peut refuser de stocker un cookie ou le supprimer
→ Ne JAMAIS supposer qu'un cookie existe.

Les sessions

Présentation

Définition

Une session PHP est un mécanisme permettant de conserver des données d'un client sur le serveur entre plusieurs requêtes.

Propriétés

Une session :

- ✓ est créée par le serveur (PHP)
- ✓ est stockée sur le serveur
- ✓ possède un identifiant unique attribué à sa création
- ✓ transmet uniquement l'identifiant au client dans un cookie

Créer une session

1. Un client fait une requête HTTP
2. Le fichier ciblé contient la fonction PHP `session_start()` :

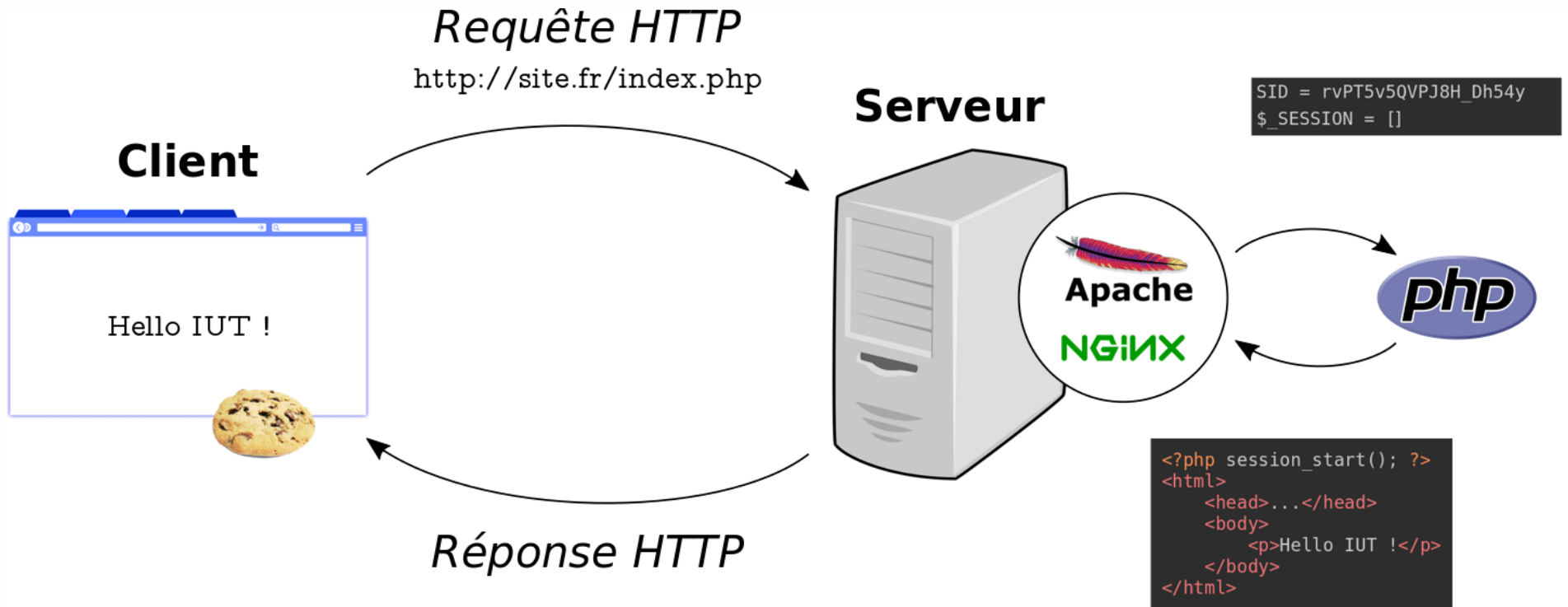
```
session_start();
```

3. Un identifiant unique est créé et associé à la session du client
4. Un tableau `$_SESSION` associé à l'identifiant est créé sur le serveur
5. L'identifiant est transmis au client par cookie :

```
Set-Cookie: PHPSESSID=rvPT5v5QVPJ8H_Dh54yWuTfhBDbHZ_BkDC1TaWBn8RxBcDpy
```

... et le client ne connaît donc que son identifiant.

Schéma de la création d'une session



Lire les données d'une session

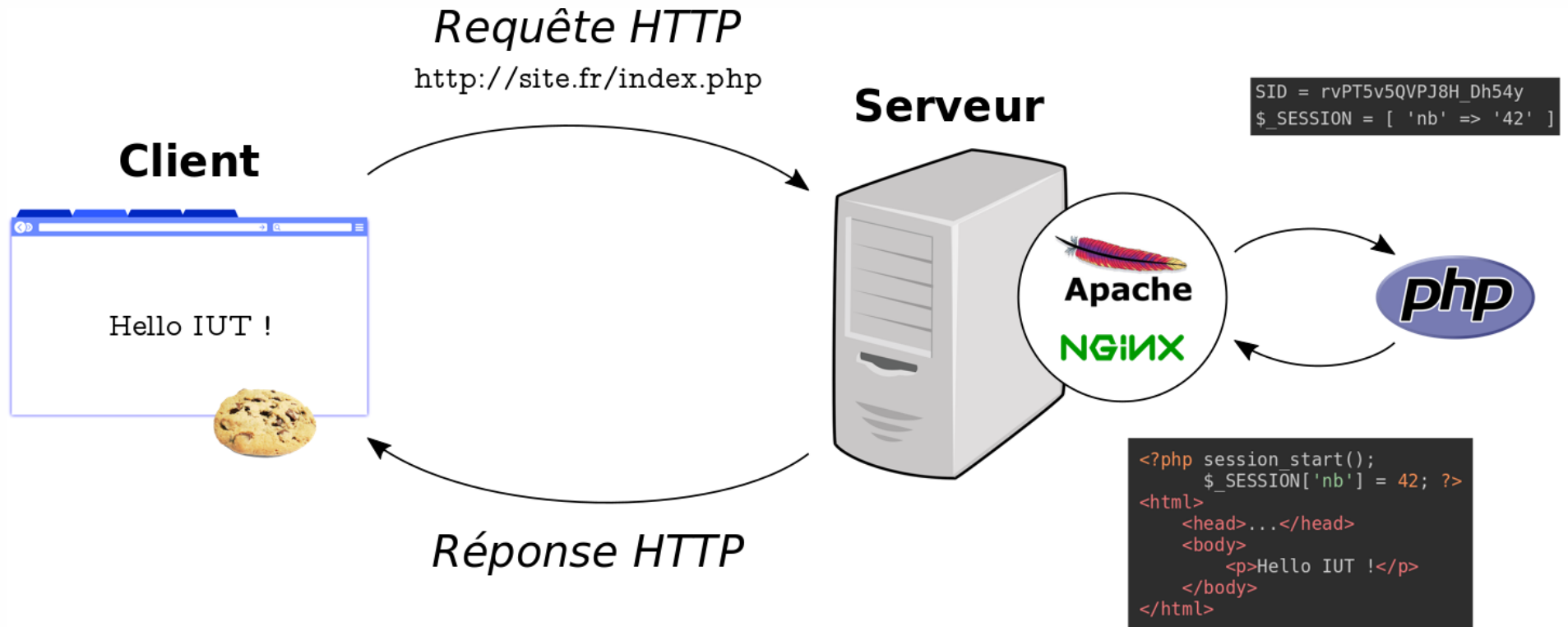
1. Le client possède un cookie avec son identifiant de session.
2. Le client fait une requête HTTP vers le même serveur.
3. Le cookie de session du domaine est ajouté à l'entête de la requête :

```
Cookie: PHPSESSID=rvPT5v5QVPJ8H_Dh54yWuTfhBDbHZ_BkDC1TaWBn8RxBcDpy
```

4. Le serveur PHP charge le tableau `$_SESSION` associé à l'identifiant.

... et peut alors accéder aux variables de session du client.

Schéma de la lecture d'une session



Détruire une session

1. Le client possède un cookie avec son identifiant de session.
2. Le client fait une requête HTTP vers le même serveur.
3. Le fichier ciblé contient la fonction PHP `session_destroy()` :

```
session_destroy();
```

4. Toutes les données du tableau de la session du client sont détruites.

Attention : la session existe toujours, seules les données sont détruites.

Sessions : les 2 règles d'or

Règle n°1

L'instruction `session_start()` doit TOUJOURS être appelée AVANT tout texte du corps de la réponse HTTP.

Règle n°2

Un client peut refuser de stocker un cookie
→ ne JAMAIS supposer qu'une session existe.

Note : à l'inverse des cookies, les données des sessions sont stockées sur le serveur. Un client ne peut donc ni les modifier ni les supprimer.