Technologies du web

3 - PHP

Client - Serveur

Coté serveur : le HTML est généré et envoyé

Coté client : le HTML est interprété par le navigateur

Serveur : **un ordinateur** (généralement très puissant) **relié au réseau** (Internet)

Le serveur **répond aux requêtes provenant du réseau**

Adresse du serveur

Sur un réseau IP (Internet Protocol) un serveur est identifié par son adresse IP

Ex : 147.99.34.8

Pour que ce soit pratique on peut aussi utiliser des noms de domaines

Ex : tela-botanica.org

HTTP

Protocole d’échange de données pour le web

Ensemble d’étapes codifiées et normalisées permettant l’échange entre un client et un serveur

Serveur web

Par abus de langage, serveur web désigne autant la machine que le logiciel répondant aux requêtes.

Une requête HTTP sera reçue par le serveur, dirigée vers le logiciel qui l’analysera et renverra un résultat en guise de réponse.

Soit un fichier statique, soit une réponse dynamique calculée par le serveur.

Client

Serveur

Requête HTTP

Internet

Réponse HTTP

Requête HTTP

Host : URL appelée
Method : GET / POST / PUT / DELETE

User-Agent : Identifiant de version du navigateur
Content-Type : type de contenu demandé
Authorization : information d’identification
Cookie : informations utilisateur transmises
Accept : type de réponse acceptée

Actions HTTP

GET : demande d’une donnée, d’une page
POST : envoi des données

DELETE
PUT
OPTIONS
HEAD

Code de réponse

200 : OK
404 : Not Found
403 : Forbidden
500 : Internal Error
301 : Moved Permanently
...

Visualiser les Headers

On trouve le détails des requêtes dans l'**onglet réseau des outils de développement** du navigateur.
Pour les ouvrir faire F12 ou clic droit -> Examiner l'élément

Sinon on peut lancer la requête avec la commande cURL :

```
curl -X GET http://perdu.com -vI
```

Composition de l’URL d’une requête

- Protocole
  - http://
- Domain cible
  - perdu.com, localhost
- Chemin
  - /le_chemin/vers/la_ressource
- Paramètres
  - **?**param1=valeur1**&**password=pouet**&**login=toto

Composition de l’URL d’une requête

Domaine :

https://**en.wikipedia.org**

Ressource :

https://en.wikipedia.org**/wiki/List_of_HTTP_status_codes**

Paramètres :

https://en.wikipedia.org/w/index.php**?search=coucou&title=Special**

Localhost

C’est le nom de domaine de la machine locale.

Son IP est 127.0.0.1

Ex : http://localhost/~moi/public_html/index.html

Logiciel serveur

Apache

NGINX

Varnish

Tomcat

Apache

Écoute les requêtes sur les ports 80 et 443

Sert des fichiers

Redirige, filtre, réécrit des requêtes

Gère les droits d’accès

… moult !

Servir des fichiers

On peut configurer le répertoire correspondant à la racine (root) du site.

Habituellement c'est **/var/www** (ou **/home/~*user*/public_html**)

Toutes les URL seront relatives à la racine :

http://localhost/index.html → /var/www/index.html

http://localhost/images/coucou.jpg → /var/www/images/coucou.jpg

Servir du contenu dynamique

Grâce à un langage de programmation, à un framework

PHP	→	Symfony, Laravel
Javascript	→	NodeJS
Python	→	Django
Ruby	→	Ruby On Rails
Java		

PHP

1994 : Créé par un gus pour site perso, dérivé de Perl

1995 : 1.0 Personal Home Page Tools

1998 : 3.0 Géré par une équipe, renommé PHP Hypertext Preprocessor

2000 : 4.0

2004 : 5.0 Modèle objet, professionnalisation du langage, gloire

2016 : 7.0 (PHP 6 abandonné)

Fichier PHP

Les fichiers en .php demandés à Apache seront d'abord interprétés par le moteur PHP

Les portions de code PHP sont placées entre des balises pour pouvoir être interprétées :

```
<?php

    // code PHP

?>
```

Fichier .php

```php
<?php

    echo 'Coucou !';

    echo "Comment ça va aujourd’hui ?";

?>
```

PHP et HTML

```
<p>Paragraphe de texte hors PHP</p>

<?php

    echo '<h1>Coucou !</h1>';

    echo "<h2>Comment ça va aujourd'hui ?</h2>";

?>

<p>Paragraphe de texte hors PHP</p>
```

Interprétation du PHP

Le texte entre les balises **<?php … ?>** sera interprété par le moteur PHP, le résultat sera renvoyé par le serveur

En dehors de ces balises le texte sera renvoyé tel quel.

Commentaires en PHP

```php
<?php

    // Commentaire sur une ligne

    /* bloc de commentaires
       sur plusieurs
       lignes
    */

    // echo 'code commenté qui n'affichera rien';

    echo 'pas un commentaire, s'affichera normalement';
```

Variables en PHP

```php
<?php

    $a = 'Résultat : ';     // affectation d'une chaîne

    $b = 5;                 // affectation d'un nombre

    $resultat = $b + 42;    // affectation d'un résultat

    echo "$b + 42 =<br>";
    echo $resultat;

?>
```

Chaines de caractères

```php
<?php

    $message = 'Coucou !';
    $message2 = $message . ' Comment ça va ? (2)'; // concaténation
    $message3 = "$message2 Bien bien ? (3)";       // substitution
    $message4 = '$message $message2 (4)';

    echo "message : $message<br>";
    echo "message2 : $message2<br>";
    echo "message3 : $message3<br>";
    echo "message4 : $message4<br>";
```

Opérations

Opérations élémentaires

Exemple	Nom	Résultat
+$a	Identité	Conversion de *$a* vers int ou float, selon le plus approprié.
-$a	Négation	Opposé de *$a*.
$a + $b	Addition	Somme de *$a* et *$b*.
$a - $b	Soustraction	Différence de *$a* et *$b*.
$a * $b	Multiplication	Produit de *$a* et *$b*.
$a / $b	Division	Quotient de *$a* et *$b*.
$a % $b	Modulus	Reste de *$a* divisé par *$b*.
$a ** $b	Exponentielle	Résultat de l'élévation de *$a* à la puissance *$b*. Introduit en PHP 5.6.

Condition

if, else, elseif

```
<?php

    if ($a == $b) {
        echo "égalité";
    } elseif ($a > $b) {
        echo "supériorité";
    } else {
        echo "différence";
    }
```

Comparaisons

> et **<**

>= et **<=**

== et **!=** égalité et différence

=== et **!==** comparaison stricte, teste valeur ET type

Logique

&& et **||** (équivalent de AND et OR)

```
<?php

    $a = 10

    if ($a > 5 && $a < 15) { … }

    if ($a > $b || $a > 42) { … }
```

Boucle

```php
<?php

    for ($i = 0; $i < 42; $i++) {

        echo $i;

    }
```

Tableau (Array)

Un Array contient une liste de valeurs indexées.
Ici l'index est défini automatiquement, en commençant par 0

```
<?php

    $semaine =
        [ 'lundi', 'mardi', 'mercredi', 'jeudi', 'vendredi' ];

    echo $semaine[0];       // affiche lundi
    echo $semaine[3];       // affiche jeudi

    print_r($semaine);      // affiche le tableau
```

Tableau avec index

Il est possible et pratique de définir l'index d'un tableau associatif

Couples clé-valeur : `['index' => 'valeur']`

```php
<?php

    $ages = [
        'Thierry' => 42,
        'Bobby'   => 23,
    ];

    echo $ages['Bobby'];        // affiche 23
```

Boucle sur un tableau

```
foreach ($tableau as $index => $valeur) {
```

Exemple :

```
<?php

    $semaine =
    [ 'lundi', 'mardi', 'mercredi', 'jeudi', 'vendredi' ];

    foreach ($semaine as $index => $jour) {

        echo 'Jour n°' . $index + 1 . ' : ' . $jour;

    }
```

Fonctions natives

PHP est constitué de nombreuses fonctions prédéfinies (fonctions mathématiques, de manipulation de fichiers, de données, de dates, …)

```
nom_fonction(param1, param2, …);
```

```
date('Y-m-d H:i:s');    // affiche l'heure de la requête
                        // ex: 2018-09-26 14:02:46
```

Variable globales

PHP propose d’accéder via les nombreuses variables globales aux paramètres de la requête ayant appelé le script

```
$_SERVER['REMOTE_ADDR'];       // affiche l’IP du client
```

Paramètres d’URL

La variable globale $_GET permet d’obtenir les paramètres de l’URL ayant appelé le script

https://en.wikipedia.org/w/index.php**?search=coucou&title=Special**

```
echo $_GET['search'];   // affiche coucou

echo $_GET['title'];    // affiche Special
```

Exemple qui affiche l’heure du serveur

```
<html>
     <head>
         <meta charset="UTF-8">
     </head>
     <body>
         <h1>Il est actuellement:</h1>
         <?php
             echo date('Y-m-d H:i:s');
         ?>
     </body>
</html>
```

Résumé

1. L’utilisateur fait une requête depuis son navigateur (url, lien...)
2. La requête est envoyée vers le serveur désigné (DNS / IP / Protocole HTTP)
3. Le logiciel serveur HTTP reçoit et analyse la requête
4. Il charge la ressource désignée depuis le répertoire ‘root’ du site
5. Si la ressource est un fichier PHP, le code est interprété
6. Le résultat du traitement est renvoyé par le serveur vers le client (HTTP)
7. Le navigateur reçoit le résultat HTML/CSS/Javascript
8. Le navigateur analyse ce résultat et l’affiche

Interagir avec une page web

Avec les formulaires !

Des champs :

- text
- list
- radio
- checkbox

Et des boutons

Balise form

```
<p>Saisissez votre message :</p>

<form>

    <!-- contenu du formulaire →

</form>
```

Types de champs de formulaires

Texte (text)
Case à cocher (checkbox)
Liste (select)
Choix unique (radio)
Bouton (button, submit, reset)

Exemple de formulaire simple

```
<form>
    <label>Nom</label>
    <input type="text" name="formnom" id="nom">
</form>
```

Attributs optionnels: placeholder, maxlength, size, …

Attention : name != id

Le **name** c'est pour retrouver la valeur saisie coté traitement du formulaire
Le **id** c'est pour ajouter du style notamment

Formulaires : Textes spéciaux

```
<form>
    <label>Mot de passe</label>
    <input type="password" name="mdp" id="mdp">

    <label>Email</label>
    <input type="email" name="courriel" id="courriel">

    <label>URL</label>
    <input type="url" name="site" id="site">
</form>
```

Formulaires : Nombres

```
<form>
    <input type="number" name="age" id="age">
    <input type="range" name="un-nbr" id="un-nbr">
</form>
```

Attributs : min, max, step, value

Formulaires : Texte long

```
<form>
    <label>Nom : </label><br/>
    <textarea name="details" id="details" rows="10">
        Du texte…
    </textarea>
</form>
```

Formulaires : Cases à cocher

```
<form>
    <p> Cochez votre choix: <br />
        <input type="checkbox" name="frites" id="frites">
        <label for="frites">Frites</label><br />
        <input type="checkbox" name="steak" id="steak">
        <label for="steak">Steak haché</label><br />
    </p>
</form>
```

Formulaires : Choix unique

Le **name** doit être le même !

```
<form>
    <p> Cochez un seul choix: <br/>
        <input type="radio" name="sauce" id="ketchup">
        <label for="ketchup">ketchup</label><br/>
        <input type="radio" name="sauce" id="mayo">
        <label for="mayo">Mayonnaise</label><br/>
    </p>
</form>
```

Formulaires : Liste déroulante

```
<form>
    <select name="pays">
        <option value="france">France</option>
        <option value="espagne">Espagne</option>
        <option value="italie">Italie</option>
    </select>
</form>
```

Formulaires : Boutons

```
<input type="button" value="un bouton">

<button type="button">Mon bouton</button>
```

Formulaires : Particularités

La propriété required pour rendre un champ obligatoire :

```
<input type="text" name="prenom" id="prenom" required>
```

La propriété checked pour rendre un radio/checkbox sélectionnée par défaut :

```
<input type="checkbox" name="oui" id="non" checked>
```

Le type hidden pour cacher un champ et sa valeur

```
<input type="hidden" name="cachecache" value="42">
```

Formulaires : Envoi du formulaire

```
<input type="submit" value="Envoyer">
```

Formulaires : Action et Method

Par défaut, une requête HTTP est envoyée lorsque le bouton submit d'un formulaire est actionné.

La méthode HTTP et l'URL de destination de la requête sont précisées dans les attributs de la balise form :

```
<form action="URL" method="GET/POST">
```

Formulaires : Exemple

```
<form method="GET" action="login.php">
    <fieldset>
        <legend>Sign in</legend>
        <label for="login">Login</label>
        <input name="login" id="login" type="text"><br/>

        <label for="pwd">Password</label>
        <input name="pwd" id="pwd" type="password"><br/>

        <input type="submit" value="login">
    </fieldset>
</form>
```

Formulaires : Résumé

1. L’utilisateur remplit le formulaire
2. Il clique sur submit
3. Le navigateur envoie une nouvelle requête à l’adresse spécifiée dans l’attribut “action” avec les données du formulaire et suivant la méthode choisie (GET ou POST)
4. Le serveur traite la requête et envoie une réponse sous la forme d’une nouvelle page HTML
5. Le navigateur affiche cette nouvelle page

Traitement coté PHP

- Avec GET les données sont passées en paramètres dans l'URL d'appel au script PHP (action)
  - Pratique si peu de données ou si l'URL doit pouvoir être réutilisée

- Avec POST les données sont passées dans le corps de la requête, les paramètres ne seront pas visibles dans l'URL
  - Pratique pour envoyer des fichiers ou lorsque les paramètres ne doivent pas être visible ou bien sont trop nombreux

Traitement coté PHP

On va retrouver les données passées au formulaire dans les variables globales :

`$_GET`

`$_POST`

`$_REQUEST` (Array de $_GET et $_POST)

Exemple du login

On va écrire un formulaire de login avec un couple de champs login et password dans un fichier form.php

Le formulaire enverra les données au script login.php

1. Demande de form.php au serveur
2. Réception et affichage de form.php coté client
3. Envoi des données à login.php sur le serveur
4. Réception de la réponse provenant du traitement des données par login.php

Le formulaire : form.php

```
<html>
    <body>
        <form action="login.php" method="post">

            <input type="text" name="login" required>
            <input type="password" name="pass" required>

            <input type="submit" value="Me connecter">

        </form>
    </body>
</html>
```

Traitement : login.php

```
<?php
    $message = ''; // Message à afficher à l'utilisateur
    if (!empty($_POST)) { // le formulaire a été envoyé
        // on vérifie login et le mot de passe
        if ($_POST['login'] == 'sam' && $_POST['pwd'] =='toto') {
            $message = 'login ok';
        } else {
            $message = 'erreur de login';
        }
    } else { // $_POST EMPTY
        $message = 'aucune donnée envoyée';
    }
?>
<html><body><p>
    <?php echo $message; ?>
</p></body></html>
```