

Les superglobales Server, Env et Session

Table des matières

I. Contexte	3
II. Les informations serveur	3
III. Exercice : Appliquez la notion	7
IV. Les cookies	7
V. Exercice : Appliquez la notion	11
VI. Les sessions	11
VII. Exercice : Appliquez la notion	14
VIII. Auto-évaluation	14
A. Exercice final	14
B. Exercice : Défi	15
Solutions des exercices	17

I. Contexte

Durée : 1 h

Environnement de travail : Travailler sur son propre environnement PHP

Pré-requis : Bases de PHP, HTML, protocole HTTP

Contexte

Les variables superglobales sont des variables mises à disposition nativement par PHP.

Elles sont particulières du fait de leur syntaxe et de leur utilisation. Celles-ci sont accessibles partout dans le code, peu importe le contexte.

Elles permettent de disposer d'informations sur la configuration du serveur, des variables d'environnement qui y sont définies, ou encore de stocker des informations pendant un certain temps au niveau du serveur, mais aussi sur le navigateur de l'utilisateur.

Nous allons voir ce qu'elles comportent précisément et comment les utiliser à bon escient dans nos scripts.

II. Les informations serveur

Objectifs

- Utiliser la variable `$_SERVER` pour manipuler des informations propres au serveur
- Utiliser la variable `$_ENV` pour manipuler des informations propres à l'environnement

Mise en situation

Il existe de nombreuses situations où il s'avère nécessaire d'en connaître plus sur notre utilisateur ou sur l'environnement de travail sur lequel un script est exécuté.

Il peut être utile, par exemple, de connaître l'adresse IP du client afin de comptabiliser les visiteurs uniques d'un site web, ou bien le navigateur utilisé par un utilisateur afin d'adapter le comportement d'un programme.

Ceci est rendu possible grâce aux superglobales `$_SERVER` et `$_ENV`.

Rappel Requête HTTP <=> Serveur

Pour comprendre l'intérêt des variables superglobales, rappelons succinctement ce qu'il se passe lorsque nous appelons une page web :

Une requête est envoyée au serveur grâce au protocole HTTP, et celle-ci va lui formuler une demande en envoyant des paramètres comme des informations sur le client, le nom de la page ou du script que l'on cherche à afficher.

Une fois la requête réceptionnée par le serveur, le script PHP sera exécuté et le résultat renvoyé au navigateur.

Syntaxe

Les superglobales s'écriront toujours en **majuscules** et débuteront par un **underscore** (à l'exception de `$GLOBALS`): `$_SERVER`, `$_ENV`, `$_SESSION` par exemple.

Ce sont des tableaux associatifs dans lesquels les clés sont le nom des variables : `$_SERVER['REMOTE_ADDR']`.

\$_SERVER : information client / serveur HTTP

C'est à l'intérieur de la variable \$_SERVER¹ que seront stockées :

- Les informations sur la requête reçue : requête GET ou POST, IP du client, la date et l'heure de la requête, pour n'en citer que quelques-unes.
- Les informations sur le script PHP en charge de traiter la requête : nom du fichier PHP, chemin vers ce fichier.

Exemple

Ce script permet d'afficher l'ensemble des informations présentes dans \$_SERVER :

```
1 <?php
2
3 foreach ($_SERVER as $key => $serv) {
4     echo "[$key] = $serv<br>";
5 }
6
```

```
[DOCUMENT_ROOT] = /home/runner/superglobales
[REMOTE_ADDR] = 172.18.0.1
[REMOTE_PORT] = 60058
[SERVER_SOFTWARE] = PHP 7.2.24-0ubuntu18.04.3 Development Server
[SERVER_PROTOCOL] = HTTP/1.1
[SERVER_NAME] = 0.0.0.0
[SERVER_PORT] = 8000
[REQUEST_URI] = /
[REQUEST_METHOD] = GET
[SCRIPT_NAME] = index.php
[SCRIPT_FILENAME] = /home/runner/superglobales/index.php
[PHP_SELF] = index.php
[HTTP_HOST] = superglobales-devtest1.rnpl.co
[HTTP_USER_AGENT] = Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:74.0) Gecko/20101011 Firefox/74.0
[HTTP_ACCEPT] = text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
[HTTP_ACCEPT_ENCODING] = gzip, deflate, br
[HTTP_ACCEPT_LANGUAGE] = fr-fr;q=0.8,en-US;q=0.5,en;q=0.3
[HTTP_REFERER] = https://rpl.ni@devtest1.superglobales
[HTTP_IS] = trailers
[HTTP_UPGRADE_INSECURE_REQUESTS] = 1
[HTTP_X_FORWARDED_FOR] = 35.191.0.33
[HTTP_X_REPLIT_USER_ID] =
[HTTP_X_REPLIT_USER_NAME] =
[HTTP_X_REPLIT_USER_ROLES] =
[REQUEST_TIME_FLOAT] = 1583918063.5116
[REQUEST_TIME] = 1583918063
```

Parmi les données les plus utilisées :

- REMOTE_ADDR : adresse IP du client, utile pour connaître le nombre de visiteurs uniques sur un site web.
- HTTP_USER_AGENT : informations sur le navigateur client (nom, version), utile par exemple pour identifier un problème survenu sur un navigateur en particulier.
- REQUEST_METHOD : indique si la requête est GET ou POST.
- HTTP_REFERER : indique la page d'où la requête a été demandée, permet de contrôler l'origine de la requête et éventuellement de bloquer celles qui ne viendraient pas d'un endroit précis.

Attention

Selon le type de serveur (Apache, NGINX pour ne citer que les plus connus), certaines valeurs pourront différer, exister ou non.

Pour les vérifier rapidement, il est possible d'afficher des informations globales sur le serveur grâce à la fonction phpinfo().

```
1 <?php
2
3 phpinfo();
4
```

Variable	Value
\$_SERVER[DOCUMENT_ROOT]	/home/runner/superglobales
\$_SERVER[REMOTE_ADDR]	172.18.0.1
\$_SERVER[REMOTE_PORT]	48544
\$_SERVER[SERVER_SOFTWARE]	PHP 7.2.24-0ubuntu18.04.3 Development Server
\$_SERVER[SERVER_PROTOCOL]	HTTP/1.1
\$_SERVER[SERVER_NAME]	0.0.0.0
\$_SERVER[SERVER_PORT]	8000
\$_SERVER[REQUEST_URI]	/

¹ <https://www.php.net/manual/fr/reserved.variables.server.php>

\$_ENV : les variables d'environnement du serveur

Si \$_SERVER permet de stocker des informations précises concernant le traitement d'une requête HTTP par le serveur, la superglobale \$_ENV¹ permet quant à elle de stocker des informations sur l'environnement d'exécution global.

Attention

Si vous effectuez ce test depuis Repl.it, \$_ENV peut être vide comme c'est le cas ci-dessous.

Le serveur est tout simplement configuré pour ne pas *hydrater* la superglobale \$_ENV. C'est pourquoi il est important pour toute la suite de ce cours de réaliser les exemples et les exercices sur votre propre environnement (voir pré requis).

```
1 <?php
2
3 var_dump($_ENV);
4
```

```
array(0) {}
```

Complément `getenv()`

Si la superglobale n'a pas été déclarée, il est cependant possible de lire les variables d'environnement grâce à la fonction PHP `getenv()`.

Exemple

```
1 <?php
2
3 foreach (getenv() as $key => $value) {
4     echo "[$key] => $value<br>";
5 }
6
```

```
[PATH] => /home/runner/.apt/bin:/usr/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
[HOSTNAME] => fb8b465c163
[NDG_CONFIG_HOME] => /config
[LC_ALL] => en_US.UTF-8
[LANG] => en_US.UTF-8
[APT_OPTIONS] => -o debug::nolocking=true -o dir::cache=/tmp/apt/cache -o dir::state=/tmp/apt/state -o dir::etc::sourcelist=/tmp/apt/sources
/sources.list
[LD_LIBRARY_PATH] => /home/runner/.apt/usr/lib/x86_64-linux-gnu:/home/runner/.apt/usr/lib/lsb/x86_64-linux-gnu:/home/runner/.apt/usr/lib:
[LIBRARY_PATH] => /home/runner/.apt/usr/lib/x86_64-linux-gnu:/home/runner/.apt/usr/lib/lsb/x86_64-linux-gnu:/home/runner/.apt/usr/lib:
[INCLUDE_PATH] => /home/runner/.apt/usr/include:/home/runner/.apt/usr/include/lsb/x86_64-linux-gnu:
[CPATH] =>
[CPPPATH] =>
[PKG_CONFIG_PATH] => /home/runner/.apt/usr/lib/x86_64-linux-gnu/pkgconfig:/home/runner/.apt/usr/lib/lsb/x86_64-linux-gnu/pkgconfig:
[LD_PRELOAD] => /usr/local/lib/repl.so
[DISPLAY] => MAGIC
[HOME] => /home/runner
[TERM] => xterm-256color
```

Il est possible de cibler la récupération d'une variable spécifique en passant son nom en paramètre : `getenv('PATH')` ;.

Comme nous le voyons, les serveurs disposent par défaut de variables d'environnement. Toutefois, elles n'auront que peu d'intérêt pour nous.

Méthode Définir ses propres variables d'environnement

Vous avez également la possibilité d'ajouter une variable d'environnement depuis votre script courant grâce à la fonction `putenv`².

La durée de vie de cette valeur n'excédera cependant pas la durée de vie du script.

1 <https://www.php.net/manual/fr/reserved.variables.environment.php>

2 <https://www.php.net/manual/fr/function.putenv.php>

[illegible]

serveur de recette client

Syntaxe **À retenir**

- `$_SERVER` est un tableau associatif contenant des informations client/serveur liées à une requête HTTP.
- `$_ENV` contiendra quant à elle des informations plus générales sur l'environnement d'exécution des scripts.
- L'accès aux variables d'environnement peut se faire de deux manières : `$_ENV['DATABASE_NAME']` ou `getenv('DATABASE_NAME')`.

Complément`$_SERVER1``$_ENV2``getenv()3``putenv()5`

III. Exercice : Appliquez la notion

Pour réaliser cet exercice vous aurez besoin de travailler sur votre environnement de développement PHP (voir pré requis).

**Question 1**

[solution n°1 p.19]

Affichez les variables serveur suivantes :

- La méthode de requête utilisée pour accéder à la page courante
- L'adresse IP de l'utilisateur qui demande la page courante
- Le chemin absolu vers le fichier correspondant à la page courante

Indice :

Vous pouvez vous aider de la documentation⁷.

Question 2

[solution n°2 p.19]

Définissez une variable d'environnement que vous appellerez `contexte` et dont la valeur sera `developpement`. Affichez cette variable, une fois définie.

IV. Les cookies

1 <https://www.php.net/manual/fr/reserved.variables.server.php>
2 <https://www.php.net/manual/fr/reserved.variables.environment.php>
3 <https://www.php.net/manual/fr/function.getenv.php>
4 <https://www.php.net/manual/fr/function.getenv.php>
5 <https://www.php.net/manual/fr/function.putenv.php>
6 <https://repl.it/>
7 <https://www.php.net/manual/fr/reserved.variables.server.php>

Objectifs

- Savoir créer un cookie
- Savoir utiliser un cookie
- Connaître les bases légales concernant l'utilisation des cookies

Mise en situation

Pour améliorer l'expérience utilisateur, PHP et le protocole HTTP nous permettent de stocker des données directement sur la machine de l'utilisateur.

C'est par exemple grâce à eux qu'il est possible pour un utilisateur de ne pas avoir à saisir des identifiants à chaque connexion, ou de mémoriser certaines préférences d'utilisation du site.

Définition Cookie

Un cookie est une information envoyée par le serveur HTTP au client, généralement sous format texte.

Ce dernier conserve ces informations sur sa machine pendant un temps défini, qui seront renvoyées au serveur à chaque requête HTTP.

Ces éléments sont définis et stockés côté client pour un domaine en particulier.

La communication des cookies se fera par le protocole HTTP via les en-têtes **cookie** pour l'envoi de la requête et **set-cookie** pour l'envoi de la réponse.

Le cookie sera ensuite accessible sur toutes les pages car échangé lors de chaque requête émises par le client.

Attention

Les utilisateurs peuvent décider de supprimer les cookies, ce qui peut entraîner des comportements tels que la déconnexion d'un site.

Pour créer un cookie, le nom ne doit pas avoir d'espace ou alors être séparé par des underscore.

Syntaxe

Techniquement, un cookie est défini en PHP grâce à la méthode `setcookie()`.

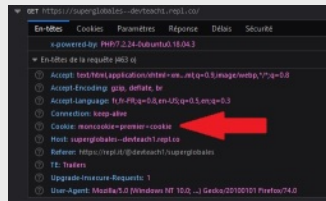
`setcookie()` accepte plusieurs paramètres :

- `name` : le nom du cookie
- `value` : la valeur du cookie
- `expires` : la date d'expiration du cookie (facultatif, en timestamp, par défaut expire à la fermeture du navigateur)
- `path` : les répertoires pour lesquels le cookie est disponible (facultatif, par défaut le répertoire où il a été défini)
- `domain` : le domaine pour lequel le cookie est disponible (facultatif, par défaut le domaine où il a été défini)

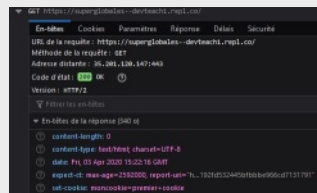
Exemple

```
1 <?php
2
3 setcookie('moncookie', 'premier cookie');
4
```

Lors d'échanges entre le client et le serveur, le cookie est transmis par le client dans l'en-tête de la requête HTTP.



Celui-ci est ensuite renvoyé par le serveur dans l'en-tête de la réponse HTTP (voir dernière ligne de l'image ci-dessous).



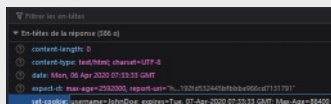
Complément

D'un point de vue sécurité, la méthode accepte deux paramètres :

- `secure` : transmission du cookie par le protocole HTTPS
- `httponly` : cookie accessible uniquement par le protocole HTTP/HTTPS

Exemple Définition d'un cookie expirant dans 1 jour

```
1 <?php
2
3 setcookie('username', 'JohnDoe', time() + 86400);
4
```



Explications : **username** est le nom du cookie, **JohnDoe** sa valeur et **time() + 86400** indique que le cookie va expirer un jour après sa création.

La fonction `time()`¹ retourne un timestamp, c'est-à-dire le nombre de secondes depuis le 1^{er} janvier 1970, auquel nous ajoutons la durée de conservation du cookie, en secondes, souhaitée : 3600 secondes pour 1 h, 86400 pour 24 h, et ainsi de suite.

Dans ce cas, le cookie sera disponible sur la machine cliente et sera utilisable pendant une journée.

Méthode Effacer un cookie

```
1 <?php
2
3 setcookie('username', '', time() - 60);
4
```

L'astuce pour effacer un cookie est de redéfinir le cookie sans sa valeur et avec une date d'expiration passée. Ici, 60 secondes avant sa création.

¹ <https://www.php.net/manual/fr/function.time>

Une fois créé, le cookie pourra être utilisé côté serveur grâce à la superglobale `$_COOKIE`. Il s'agit d'un tableau associatif dont les clés sont les noms des cookies.

Attention

Une fois que les cookies ont été créés, ils ne seront accessibles que lors du prochain chargement de page, dans le tableau `$_COOKIE`.

Exemple

```
1 <?php
2
3 setcookie('username', 'JohnDoe', time() + 3600);
4
5 print_r($_COOKIE);

1 Array([username] => JohnDoe)
```

Le cookie est bien disponible, nous aurions aussi pu écrire : `$_COOKIE['username']`.

Pour l'effacer :

```
1 <?php
2
3 setcookie('username', '', time() - 3600);
4
5 print_r($_COOKIE);

1 Array()
```

Texte légal Que dit la Loi ?

Le RGPD (Règlement Général sur la Protection des Données) définit les règles concernant le stockage et la manipulation de données personnelles. Les cookies sont ainsi soumis à certaines obligations :

- Recueil d'un consentement pour permettre le stockage chez l'utilisateur : il s'agit souvent d'un bandeau rappelant la loi avec un bouton d'acceptation ou de refus,
- Le consentement n'est valable que pendant 13 mois maximum, il doit être à nouveau soumis passé ce délai,
- Les cookies ne sont valables que pendant 13 mois maximum.

Tous les cookies ne sont pas concernés par cette réglementation. Les cookies techniques (identifiants de session, gestion d'un panier d'achat...) nécessaires au bon fonctionnement d'un site ne requièrent pas le consentement de l'utilisateur.

Syntaxe À retenir

- La méthode `setcookie(name, value, expires)` permettra de créer et effacer un cookie : `setcookie('stayConnect', true, time() + 864000)`.
- La superglobale `$_COOKIE` permettra d'utiliser les cookies : `if ($_COOKIE['stayConnect'])`.

Complément`$_COOKIE1``setcookie()2``time()3`Les cookies et la loi⁴

V. Exercice : Appliquez la notion

Pour réaliser cet exercice vous aurez besoin de travailler sur votre environnement de développement PHP (voir pré requis).

**Question**

[solution n°3 p.19]

Déclarez un cookie permettant de stocker la date de dernière mise à jour du panier pour un site marchand. Ce cookie aura une validité de 3 h.

Dans un second temps, une fois ce cookie déclaré, affichez son contenu et supprimez-le.

VI. Les sessions

Objectifs

- Créer une session
- Utiliser une session

Mise en situation

Tout site web nécessitant une connexion utilisateur se doit de disposer d'un système de sessions.

Nous allons voir ce qu'est une session et quels en sont les tenants et aboutissants.

Définition Session

Une session est un espace de temps défini pendant lequel des données relatives à un utilisateur sont stockées sur le serveur.

En PHP, une session est représentée par la superglobale `$_SESSION`, elle aussi accessible depuis toutes les pages de notre site ou application.

1 <https://www.php.net/manual/fr/reserved.variables.cookies.php>

2 <https://www.php.net/manual/fr/function.setcookie.php>

3 <https://www.php.net/manual/fr/function.time.php>

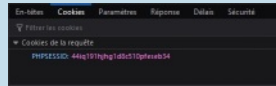
4 <https://www.cnil.fr/fr/cookies-traceurs-que-dit-la-loi>

5 <https://repl.it/>

Remarque

Les données d'une session sont stockées sur le serveur, mais lorsque celle-ci est initialisée, un cookie nommé **PHPSESSID** est créé chez le client.

Il permet de savoir à qui appartient la session et n'est valable que pour la session en cours.



Le processus de la session



La première étape est de démarrer la session avec **session_start()**: cela permettra de créer le cookie **PHPSESSID** et ainsi d'identifier la session de l'utilisateur courant.

Ensuite, la page HTML peut être construite, et **\$_SESSION** utilisée au besoin.

La dernière étape est la destruction de la session. Pour cela, il faut invoquer **session_destroy()** afin d'effacer les données de session et supprimer le cookie PHPSESSID grâce à la méthode `setcookie()` vue précédemment.

Attention

`session_start()` devra être appelée au début de chaque page où les sessions seront utilisées.

Remarque

`session_destroy()` permet de définir le moment ou l'endroit où la session ne doit plus être utilisée.

Pour utiliser à nouveau la session par la suite, il suffira de **rappeler** `session_start()`.

La session sera récupérée grâce au cookie PHPSESSID.

Complément

En pratique, `session_destroy()` sera très rarement utilisée.

Pour effacer les variables de session, nous écrirons `$_SESSION = []`.

Pour n'effacer qu'une seule variable, nous utiliserons la méthode `unset()` : `unset($_SESSION['variable'])`.

Exemple

```

1 <?php
2 session_start();
3
4 $_SESSION['username'] = 'JohnDoe';
5 echo "Bonjour " . $_SESSION['username'];
6 print_r($_SESSION);
7
  
```

```
8 $_SESSION = [];  
9 print_r($_SESSION);  
10  
11 session_destroy();  
12
```

```
Array ( [username] => JohnDoe )  
Array ( )
```

Attention

Si l'utilisateur ne permet pas l'utilisation des cookies, PHP passera les données de session (PHPSESSID) directement via l'URL.

Cas d'utilisation des sessions

- Sur un site e-commerce, pour garder le contenu d'un panier pendant le processus de commande : choix des produits -> choix de l'adresse de facturation/livraison -> choix du mode de livraison -> paiement.
- Pour gérer des droits utilisateurs : un utilisateur qui se connecte à son compte peut avoir accès ou non à certaines parties du site : s'il est administrateur, contributeur ou simple utilisateur, par exemple.
- Lors d'un quiz en ligne, dans ce cas, il n'y a pas forcément de compte utilisateur, mais les réponses peuvent transiter d'étape en étape du quiz grâce aux sessions.
- Stocker les préférences de navigation d'un utilisateur.
- Exemple du dashboard : l'utilisateur va définir comment placer ses blocs d'informations, comment les mettre en page : thème, taille, etc.

Syntaxe **À retenir**

- `session_start()` permet de démarrer une session.
- `$_SESSION` est la superglobale qui stocke les variables de session et que nous pourrons utiliser dans notre code.
- `session_destroy()` permet de détruire les variables de session, toutefois nous préférons écrire `$_SESSION = []` ou `unset($_SESSION['variable'])`.
- Pour détruire proprement une session, il faut aussi effacer le cookie PHPSESSID : `setcookie('PHPSESSID', '', time() - 3600)`.

Complément

`$_SESSION`¹

¹ <https://www.php.net/manual/fr/reserved.variables.session.php>

VII. Exercice : Appliquez la notion

Pour réaliser cet exercice vous aurez besoin de travailler sur votre environnement de développement PHP (voir pré requis)



Question

[solution n°4 p.19]

Créez un script **index.php** dans lequel vous initialiserez trois variables de session auxquelles vous attribuerez une valeur : `$name`, `$age`, `$favoriteLangages`.

Dans un second script **display-session.php**, vous afficherez le message *"Bonjour, je m'appelle ... et j'ai ... Mes langages de programmation préférés sont ..."*, que vous aurez récupéré grâce aux variables de session.

Vous prendrez soin de supprimer la session à la fin de ce second script.

Attention, vérifiez que ces variables ont bien été définies avant de les afficher.

VIII. Auto-évaluation

A. Exercice final

Exercice 1

[solution n°5 p.20]

Exercice

Grâce à quelle superglobale est-il possible d'afficher des informations sur la requête reçue ou sur le script en charge de traiter cette requête ?

- ☐ \$GLOBALS
- ☐ \$_SESSION
- ☐ \$_SERVER

Exercice

Quelle clé de la superglobale `$_SERVER` permet d'afficher l'adresse IP de l'utilisateur ?

- ☐ REMOTE_ADDR
- ☐ HTTP_USER_AGENT
- ☐ REQUEST_METHOD

Exercice

Quelle fonction PHP pouvez-vous utiliser pour afficher les informations de configuration de votre serveur et les informations propres à vos superglobales ?

- ☐ `phpinfo()` ;
- ☐ `serverinfo()` ;
- ☐ `server_info()` ;
- ☐ `php_info()` ;

1 <https://repl.it/>

Exercice

Au sein de quelle superglobale retrouve-t-on les variables d'environnement ?

- ☐ \$_ENVIRONMENT
- ☐ \$_ENV
- ☐ \$_REQUEST

Exercice

Grâce à quelle fonction est-il possible de déclarer une variable d'environnement à la volée ?

Exercice

Quel est l'ordre correct des arguments de la fonction `setcookie` ?

- ☐ Nom, valeur, délai d'expiration
- ☐ Valeur, nom, délai d'expiration
- ☐ Délai d'expiration, nom, valeur

Exercice

Suite à la mise en place de la réglementation RGPD, il est désormais nécessaire de requérir le consentement de l'utilisateur qu'importe le type de cookies utilisés sur un site Internet.

- ☐ Vrai
- ☐ Faux

Exercice

Lorsqu'une session est initialisée, dans quel cookie les informations de cette session sont-elles stockées côté client ?

Exercice

Quelle fonction est-il nécessaire d'initialiser sur toutes les pages devant faire appel à une session ?

- ☐ `session_init()`
- ☐ `session_debut()`
- ☐ `session_start()`

Exercice

À quelle méthode faut-il faire appel pour mettre fin à une session ?

B. Exercice : Défi

Vous êtes chargé de compléter l'espace de connexion à un site Internet, vous disposez du code suivant.

En l'état, celui-ci permet d'afficher un formulaire lorsque l'on accède au script directement (méthode HTTP GET).

Lorsque ce formulaire est soumis, l'attribut action étant vide, celui-ci sera traité par cette même page. Cela permettra d'accéder au script via la méthode HTTP POST.

Vous constaterez que, lorsque l'on accède à la page via cette méthode, on ne visualise plus le formulaire. On affiche en revanche le contenu des éléments précédemment saisis.

```

1 <!DOCTYPE html>
2 <html lang="fr">
3 <head>
4     <meta charset="utf-8"/>
5 </head>
6 <body>
7 <?php
8
9 if ('GET' === $_SERVER['REQUEST_METHOD']) {
10     ?>
11     <form method="post" action="">
12         <label for="name">Votre nom</label>
13         <input type="text" name="name" id="name" placeholder="Saisissez votre nom">
14
15         <label for="email">Votre adresse e-mail</label>
16         <input type="email" name="email" id="email" placeholder="Saisissez votre e-mail">
17
18         <button type="submit">Soumettre le formulaire</button>
19     </form>
20     <?php
21 } elseif ('POST' === $_SERVER['REQUEST_METHOD']) {
22     var_dump($_POST['name']);
23     var_dump($_POST['email']);
24 }
25 ?>
26
27 </body>
28 <style>
29 /**
30     Attention, ce CSS est là uniquement pour rendre le formulaire "agréable" à la lecture sans
31     que vous n'ayez
32     à récupérer deux fichiers distincts.
33     Dans un cas d'usage "réel", ces éléments doivent être externalisés
34     */
35     body {
36         font-family: Calibri, serif;
37     }
38
39     form {
40         max-width: 50%;
41     }
42
43     form label {
44         display: block;
45         font-weight: bold;
46         margin-bottom: 10px;
47     }
48
49     form input, form select, form textarea {
50         display: inline-block;
51         margin-bottom: 10px;
52         padding: 10px;
53         width: 80%;
54     }
55
56     form input[type="radio"],

```



```

56     form input[type="checkbox"],
57     form input[type="submit"] {
58         width: auto;
59     }
60
61     button[type=submit], button[type=reset] {
62         padding: 10px;
63         margin-top: 15px;
64     }
65 </style>
66 </html>
67

```

Pour réaliser cet exercice vous aurez besoin de travailler sur votre environnement de développement PHP (voir pré requis).



Question 1

[solution n°6 p.21]

À partir de votre code ou en partant de la solution de l'exercice précédent, voici ce que vous allez devoir réaliser.

Lorsque le formulaire est soumis :

- Plutôt que d'afficher les éléments saisis, assignez-les à des variables de session du même nom.

Question 2

[solution n°7 p.23]

À partir de votre code ou en partant de la solution à la question précédente, voici ce que vous allez devoir réaliser :

Lorsque vous affichez la page directement :

- Si les variables de session name/email sont initialisées, affichez à l'utilisateur : *Bienvenue %s, votre adresse e-mail est la suivante %s,*
- Affichez également le message : *Votre adresse IP est la suivante : %s.*

Question 3

[solution n°8 p.24]

Enfin, à partir de votre code ou en partant de la solution à la question précédente, voici ce que vous allez devoir réaliser.

Vous allez devoir mettre en place un compteur de visites, pour cela :

- Initialisez un cookie dont la valeur sera 1, avec une durée de validité d'une journée si le cookie n'existe pas,
- S'il existe, redéfinissez ce cookie afin d'incrémenter sa valeur,
- Affichez un message à l'utilisateur lui indiquant le nombre de fois où il a visité la page.

Solutions des exercices

1 <https://repl.it/>

p. 7 Solution n°1

```

1 <?php
2 echo $_SERVER['REQUEST_METHOD'] . PHP_EOL;
3 echo $_SERVER['REMOTE_ADDR'] . PHP_EOL;
4 echo $_SERVER['SCRIPT_FILENAME'] . PHP_EOL;
5 ?>

```

p. 7 Solution n°2

```

1 <?php
2 putenv('contexte=developpement');
3 echo getenv('contexte');
4 ?>

```

p. 11 Solution n°3

```

1 <?php
2 // 1ère étape : déclaration
3 setcookie('cartUpdate', date('d/m/y h:i:s'), time() + 10800);

1 <?php
2 // Seconde étape, car les cookies ne sont accessibles qu'après un rechargement de la page
3
4 // Affichage
5 print_r($_COOKIE['cartUpdate']);
6
7 // Suppression
8 setcookie('cartUpdate', '', time() - 3600);
9 ?>

```

p. 14 Solution n°4

```

1 <?php
2 // index.php
3 require_once("display-session.php");
4 session_start();
5
6 $_SESSION['name'] = 'Laurent';
7 $_SESSION['age'] = 27;
8 $_SESSION['favoriteLangages'] = ['PHP', 'HTML', 'CSS'];
9
10 echo 'Session initialisée';

1 <?php
2 //display-session.php
3
4 if (isset($_SESSION['name']) && isset($_SESSION['age']) &&
    isset($_SESSION['favoriteLangages']))
5 {
6     echo sprintf("Je m'appelle %s et j'ai %s ans", $_SESSION['name'], $_SESSION['age']) .
        PHP_EOL;
7     echo sprintf("Mes langages de programmation favoris sont les suivants") . PHP_EOL;
8     foreach ($_SESSION['favoriteLangages'] as $langage)

```

```

9  {
10     echo $langage . PHP_EOL;
11 }
12 } else {
13     echo "Toutes les variables n'ont pas été définies" . PHP_EOL;
14 }
15
16 session_destroy();
17 setcookie('PHPSESSID', '', time() - 3600)

```

Exercice p. 14 Solution n°5

Exercice

Grâce à quelle superglobale est-il possible d'afficher des informations sur la requête reçue ou sur le script en charge de traiter cette requête ?

- ☐ \$GLOBALS
- ☐ \$_SESSION
- ☒ \$_SERVER

Exercice

Quelle clé de la superglobale \$_SERVER permet d'afficher l'adresse IP de l'utilisateur ?

- ☒ REMOTE_ADDR
- ☐ HTTP_USER_AGENT
- ☐ REQUEST_METHOD

Exercice

Quelle fonction PHP pouvez-vous utiliser pour afficher les informations de configuration de votre serveur et les informations propres à vos superglobales ?

- ☒ phpinfo();
- ☐ serverinfo();
- ☐ server_info();
- ☐ php_info();

Exercice

Au sein de quelle superglobale retrouve-t-on les variables d'environnement ?

- ☐ \$_ENVIRONMENT
- ☒ \$_ENV
- ☐ \$_REQUEST

Exercice

Grâce à quelle fonction est-il possible de déclarer une variable d'environnement à la volée ?

putenv

Exercice


Quel est l'ordre correct des arguments de la fonction `setcookie` ?

- ☒ Nom, valeur, délai d'expiration
- ☐ Valeur, nom, délai d'expiration
- ☐ Délai d'expiration, nom, valeur

Exercice

Suite à la mise en place de la réglementation RGPD, il est désormais nécessaire de requérir le consentement de l'utilisateur qu'il importe le type de cookies utilisés sur un site Internet.

- ☐ Vrai
- ☒ Faux

 Tous les cookies ne sont pas concernés par cette réglementation. Les cookies techniques (identifiants de session, gestion d'un panier d'achat...) nécessaires au bon fonctionnement d'un site ne requièrent pas le consentement de l'utilisateur.

Exercice

Lorsqu'une session est initialisée, dans quel cookie les informations de cette session sont-elles stockées côté client ?

PHPSESSID

Exercice

Quelle fonction est-il nécessaire d'initialiser sur toutes les pages devant faire appel à une session ?

- ☐ `session_init()`
- ☐ `session_debut()`
- ☒ `session_start()`

Exercice

À quelle méthode faut-il faire appel pour mettre fin à une session ?

`session_destroy`

p. 17 Solution n°6

```
1 <!DOCTYPE html>
2 <html lang="fr">
3 <head>
4     <meta charset="utf-8"/>
5 </head>
6 <body>
7 <?php
8 session_start();
9 if ('GET' === $_SERVER['REQUEST_METHOD']) {
10     ?>
11     <form method="post" action="">
12         <label for="name">Votre nom</label>
13         <input type="text" name="name" id="name" required placeholder="Saisissez votre nom">
14     </form>
```

```

15     <label for="email">Votre adresse e-mail</label>
16     <input type="email" name="email" id="email" required placeholder="Saisissez votre e-
mail">
17
18     <button type="submit">Soumettre le formulaire</button>
19 </form>
20 <?php
21 } elseif ('POST' === $_SERVER['REQUEST_METHOD']) {
22     $_SESSION['name'] = $_POST['name'];
23     $_SESSION['email'] = $_POST['email'];
24 }
25
26 ?>
27
28 </body>
29 <style>
30 /**
31     Attention, ce CSS est là uniquement pour rendre le formulaire "agréable" à la lecture
    sans que vous n'ayez
32     à récupérer deux fichiers distincts.
33     Dans un cas d'usage "réel", ces éléments doivent être externalisés
34     */
35     body {
36         font-family: Calibri, serif;
37     }
38
39     form {
40         max-width: 50%;
41     }
42
43     form label {
44         display: block;
45         font-weight: bold;
46         margin-bottom: 10px;
47     }
48
49     form input, form select, form textarea {
50         display: inline-block;
51         margin-bottom: 10px;
52         padding: 10px;
53         width: 80%;
54     }
55
56     form input[type="radio"],
57     form input[type="checkbox"],
58     form input[type="submit"] {
59         width: auto;
60     }
61
62     button[type=submit], button[type=reset] {
63         padding: 10px;
64         margin-top: 15px;
65     }
66 </style>
67 </html>
68

```

p. 17 Solution n°7

```

1 <!DOCTYPE html>
2 <html lang="fr">
3 <head>
4     <meta charset="utf-8"/>
5 </head>
6 <body>
7 <?php
8 session_start();
9
10 if ('GET' === $_SERVER['REQUEST_METHOD']) {
11     if (isset($_SESSION['name']) && isset($_SESSION['email'])) {
12         echo sprintf('Bienvenue %s, votre adresse e-mail est la suivante %s <br/>',
13 $_SESSION['name'], $_SESSION['email']);
14         echo sprintf('Votre adresse IP est la suivante : %s <br/>', $_SERVER['REMOTE_ADDR']);
15     } else {
16         ?>
17         <form method="post" action="">
18             <label for="name">Votre nom</label>
19             <input type="text" name="name" id="name" required placeholder="Saisissez votre
20 nom">
21             <label for="email">Votre adresse e-mail</label>
22             <input type="email" name="email" id="email" required placeholder="Saisissez votre
23 e-mail">
24             <button type="submit">Soumettre le formulaire</button>
25         </form>
26     <?php
27 } elseif ('POST' === $_SERVER['REQUEST_METHOD']) {
28     $_SESSION['name'] = $_POST['name'];
29     $_SESSION['email'] = $_POST['email'];
30 }
31
32 ?>
33
34 </body>
35 <style>
36 /**
37     Attention, ce CSS est là uniquement pour rendre le formulaire "agréable" à la lecture
38     sans que vous n'ayez
39     à récupérer deux fichiers distincts.
40     Dans un cas d'usage "réel", ces éléments doivent être externalisés
41     */
42     body {
43         font-family: Calibri, serif;
44     }
45
46     form {
47         max-width: 50%;
48     }
49
50     form label {
51         display: block;
52         font-weight: bold;
53         margin-bottom: 10px;
54     }
55 }

```

```

54
55     form input, form select, form textarea {
56         display: inline-block;
57         margin-bottom: 10px;
58         padding: 10px;
59         width: 80%;
60     }
61
62     form input[type="radio"],
63     form input[type="checkbox"],
64     form input[type="submit"] {
65         width: auto;
66     }
67
68     button[type=submit], button[type=reset] {
69         padding: 10px;
70         margin-top: 15px;
71     }
72 </style>
73 </html>
74

```

p. 17 Solution n°8

```

1 <!DOCTYPE html>
2 <html lang="fr">
3 <head>
4     <meta charset="utf-8"/>
5 </head>
6 <body>
7 <?php
8 session_start();
9
10 if (!isset($_COOKIE['countVisits'])) {
11     setcookie('countVisits', 1, time() + 86400);
12 } else {
13     $countVisits = (int)$_COOKIE['countVisits'] + 1;
14
15     setcookie('countVisits', $countVisits, time() + 86400);
16 }
17
18 echo sprintf("Vous avez visité cette page %s fois <br/>", $_COOKIE['countVisits']);
19
20 if ('GET' === $_SERVER['REQUEST_METHOD']) {
21     if (isset($_SESSION['name']) && isset($_SESSION['email'])) {
22         echo sprintf('Bienvenue %s, votre adresse e-mail est la suivante %s <br/>',
23 $_SESSION['name'], $_SESSION['email']);
24         echo sprintf('Votre adresse IP est la suivante : %s <br/>', $_SERVER['REMOTE_ADDR']);
25     } else {
26         ?>
27         <form method="post" action="">
28             <label for="name">Votre nom</label>
29             <input type="text" name="name" id="name" required placeholder="Saisissez votre
30 nom">
31
32             <label for="email">Votre adresse e-mail</label>
33             <input type="email" name="email" id="email" required placeholder="Saisissez votre
34 e-mail">
35

```



```

32
33         <button type="submit">Soumettre le formulaire</button>
34     </form>
35     <?php
36     }
37 } elseif ('POST' === $_SERVER['REQUEST_METHOD']) {
38     $_SESSION['name'] = $_POST['name'];
39     $_SESSION['email'] = $_POST['email'];
40 }
41
42 ?>
43
44 </body>
45 <style>
46 /**
47     Attention, ce CSS est là uniquement pour rendre le formulaire "agréable" à la lecture
48     sans que vous n'ayez
49     à récupérer deux fichiers distincts.
50     Dans un cas d'usage "réel", ces éléments doivent être externalisés
51     */
52 body {
53     font-family: Calibri, serif;
54 }
55
56 form {
57     max-width: 50%;
58 }
59
60 form label {
61     display: block;
62     font-weight: bold;
63     margin-bottom: 10px;
64 }
65
66 form input, form select, form textarea {
67     display: inline-block;
68     margin-bottom: 10px;
69     padding: 10px;
70     width: 80%;
71 }
72
73 form input[type="radio"],
74 form input[type="checkbox"],
75 form input[type="submit"] {
76     width: auto;
77 }
78
79 button[type=submit], button[type=reset] {
80     padding: 10px;
81     margin-top: 15px;
82 }
83 </style>
84 </html>

```