Requêtes et Réponse HTTP en PHP

Récap PHP lire une requête et créer une réponse

PHP	Description	Retour
Requête HTTP		
`$_SERVER["REQUEST_METHOD"]`	La METHODE HTTP : GET, POST, ...	String
`$_SERVER["REQUEST_URI"]`	L'URI	String
`getallheaders()`	Les headers HTTP	Map
`$body = file_get_contents("php://input")`	Le body	String
`$_GET`	Query param : ?param=value	Map
`$_POST`	`<form>` inputs values	Map
`json_decode($body)`	Décode une string JSON	Array ou Map
Réponse HTTP		
`http_response_code(200)`	Status code	NULL
`header("Content-Type:application/json")`	Header	NULL
`json_encode($body)`	Encode un objet ou array en String JSON	String
`echo`	écrire dans le body	NULL

> Map est un Array qui à un texte pour clé.

Lire les infos de la Requête HTTP

Le PHP est un langage back-end. Le code php est contenu dans un script executé par un serveur web lorsque celui-ci reçoit une requête HTTP. En PHP il est possible de recupérer les données de la requête.

La requête HTTP

Une requête HTTP est faite toujours en trois parties :

- la request line, défini la METHOD HTTP et l'url
- les headers, défini des infos complementaire comme le type de ressource demandée par exemple.
- Le body, il peut être vide et il contient des données envoyés par le client comme les informations d'une formulaire, un fichier ou du JSON.

Les requêtes `GET`

Les requêtes `GET` permettent de récupérer des données d'un serveur, se sont elles qui sont utilisées par les navigateurs web.
Elle ont la particularitée de pouvoir contenir des variables.

```
GET localhost:8000/product.php?id=2 HTTP/1.1
```

On s'en sert tout particulièrement dans une balise `<a>` pour renvoyer vers une page qui afficher un produit spécifique d'une boutique en ligne par exemple.

```
<a href="localhost:8000/product.php?id=2">Air Max 2022</a>
```

Les requêtes POST

Les requêtes `POST` permettent d'envoyer des données au serveur, elles sont utilisées par les forumlaires HTML ou par l'API JavaScript `fetch()` pour envoyer des données au serveur.

Une requête POST envoyée par un formulaire HTML

```
<form action="/add-user.php" method="post">
  <input type="text" name="name" >
  <input type="email" name="email">
  <input type="password" name="password">

  <input type="submit" value="Submit">
</form>
```

Lors de l'envoi de données via un formulaire HTML, le contenu des balises `input` en placée dans le body de la requête POST avec comme identifiant leurs attributs `name` respectifs. Lors du clique sur le bouton `submit` la page navigue vers la page `add-user.php` .

```
POST localhost:8000/add-user.php HTTP/1.1
Content-Type : multipart/form-data

name:Thibaut
email:thibaut@gmail;com
password:Thibaut123
```

Une requête POST envoyée en JavaScript.

```
let user = {
    name : "Thibaut",
    email : "thibaut@gmail;com",
    password : "Thibaut123"
};
fetch("localhost:8000/add-user.php",{
    method:"POST"
    body : JSON.stringify(user);
});
```

```
POST localhost:8000/add-user.php HTTP/1.1
Content-Type : application/json

{
    "name" : "Thibaut",
    "email" : "thibaut@gmail;com",
    "password" : "Thibaut123"
}
```

L'objectif dans ce chapitre va être d'apprendre à récupérer chaqu'un de ces éléments.

Lire les données de la requête HTTP

Dans la requête :

```
GET localhost:8000/product.php?id=2 HTTP/1.1
```

La méthode est `GET`.

La méthode

```
$method = $_SERVER["REQUEST_METHOD"];
```

L'URI

L'uri c'est la ressource demandée donc tout ce qui a après le nom de domaine et le port.
Dans la requête :

```
GET localhost:8000/product.php?id=2 HTTP/1.1
```

L'URI est `/product.php?id=2`.

```
$uri = $_SERVER["REQUEST_URI"];
```

Récupérer les paramètres l'URL - `query string`

Il est possible de placer des paramètres dans l'url, ses paramètres sont appelés : `query string` .

Syntaxe query string :

host/ressource?param1=value&param2=value&paramN=value

```
$param1 = $_GET["param1"];
$param2 = $_GET["param1"];
// ...
$paramN = $_GET["paramN"];
```

Exemple :

```
<a href="localhost:8000/product.php?id=2">Air Max 2022</a>
```

```
$user_id = $_GET["id"];
$results = $bdd->query("SELECT * FROM Product WHERE id=$id");
```

> **Note**
> Dans une string il est possible de placer directement une variable sans concaténation. N'oublie pas le dollar ! `$` .
>
> ```
> $age = 24;
> echo "J'ai $age ans !";
> ```

Les Headers (en-tête)

Soit une requête HTTP `GET` :

```
GET localhost:8000/product.php?id=2 HTTP/1.1
Host : localhost:8000
Prenom : Massinissa
```

```php
$headers = getallheaders();

$headers["Host"];       // => localhost:8000
$headers["Prenom"];     // => Massinissa
```

Le Body

Le body de la requête se trouve dans un fichier caché du serveur web, on peut y accéder via la fonction `file_get_contents()` qui permet simplement de lire tout le contenu d'un fichier et renvoyer le tout dans une `string` .

```php
$body = file_get_contents("php://input");
```

Body JSON

Si le body est une string JSON je peut ensuite la convertir en Array Map PHP avec la fonction `json_decode()` .

Soit la requête HTTP suivante :

```
POST localhost:8000/add-user.php HTTP/1.1
Content-Type : application/json

{
    "name" : "Thibaut",
    "email" : "thibaut@gmail;com",
    "password" : "Thibaut123"
}
```

Je récupére mon utilisateur ainsi :

```php
<?php
$headers = getallheaders();
$body = file_get_contents("php://input");
if($headers["Content-Type"] == "application/json"){
    $user = json_decode($body);
}
?>
<p>Bonjour, <?= $user["name"] ?> !</p>
```

Lire les données d'un formulaire

Les données d'un formulaire PHP sont placées dans le body, à la différence d'un body `JSON` il est beaucoup plus simple de récupérer ces données.
Il suffit de lire le contenu du tableau `$_POST` .

Soit un formulaire HTML :

```html
<form action="/add-user.php" method="post">
  <input type="text" name="name">
  <input type="email" name="email">
  <input type="password" name="password">

  <input type="submit" value="Submit">
</form>
```

Les données du formulaire sont simplement dans le tableau `$_POST` .

```php
echo $_POST["name"];
echo $_POST["email"];
echo $_POST["password"];
```

Envoyer la réponse HTTP

Tout l'objectif du PHP c'est de construire une réponse HTTP à partir des données de la requête, que la réponse soit du HTML pour un site full-stack ou du JSON pour une API REST.

Structure d'une réponse HTTP

Une réponse HTTP est assez similaire à une requête à la différence de sa première ligne.

Syntaxe :

```
HTTP/1.1 StatusCode Reason
Header
Header
...

Body
```

Exemple réponse HTML:

```
HTTP/1.1 200 OK
Content-Type:text/html

<html>
<head>...</head>
<body>
    <h1>Basket Adidias taille 42</h1>
    <p>Livraison en 2 jours !</p>
</body>
</html>
```

Exemple réponse JSON:

```
HTTP/1.1 200 OK
Content-Type:application/json

[
    {
        "name" : "mathieu",
        "email" : "mathieu@mail.com",
        "tel" : "0601020304",
    },
        {
        "name" : "thibaut",
        "email" : "thibaut@mail.com",
        "tel" : "0601080304",
    }
]
```

Envoyer le status code

L'envoi du status code ce fait avec la fonction `http_response_code()` .
Le status code défini l'état de la réponse :

- 100 à 199 : Continue, l'utilisateur doit continuer avec une autre requête. Ces codes sont peu utilisés.
- 200 à 299 : Réussite
- 300 à 399 : Redirection de requête
- 400 à 499 : Erreur à cause du client, exemple 404 NOT FOUND ou mauvais mot de passe
- 500 à 599 : Erreur à cause du serveur, exemple echec de connexion à la BDD.
  Syntaxe :

```
http_response_code(200);    // Set status code to 200
```

Exemple :

```php
$results = $bdd->query("SELECT * FROM Product WHERE id=$_GET['id']");
if($results == false){
    http_response_code(500);    // Erreur serveur
}else{

    $products = $results->fetchAll();
    if(count($products) == 0){
        http_response_code(404);    // Ressource not found
    }else{
        http_response_code(200);    // Tout c'est bien passé.
    }
}
```

Voir la liste des status code HTTP : https://www.rfc-editor.org/rfc/rfc9110#name-overview-of-status-codes

Envoyer un header

Pour écrire une ligne de header il faut simplement utiliser la fonction `header()` .
Syntaxe :

```php
header("Header : value");
```

Exemple :

```php
header("Content-Type : application/json");
```

Le Body de la réponse HTTP

Le body est surement la partie la plus importante de la réponse.
Comme dit au début du cours, le body de la requête c'est simplement l'entiéreté du texte écrit dans le HTML et donc par extension tout ce que est affichés via `echo` .

Exemples HTML seulement:

J'écrit simplement du HTML comme d'habitude.

```
<h1>Hello World !</h1>
```

Ici le body est :

```
<h1>Hello World !</h1>
```

Exemples HTML + PHP :

Je modifie le HTML avec le PHP

```
<?php
$prenom = "Massinissa";
?>
<h1>Hello <?= $prenom ?> !</h1>
```

Ici le body est :

```
<h1>Hello Massinissa !</h1>
```

Exemples JSON + PHP :

Je place du JSON dans le body; pour que le JSON soit valide je n'écrit aucun HTML et je ne place qu'un seul et unique `echo` associé à la fonction `json_encode()` .

```
<?php
$user = [
    "name" => "Massinissa",
    "email" => "massi@mail.com",
    "tel" => "0697800000"
];
echo json_encode($user);
?>
```

Ici le body est :

```
{
    "name" : "Massinissa",
    "email" : "massi@mail.com",
    "tel" : "0697800000"
}
```

Exemple réaliste avec BDD SQL:

```php
$results = $bdd->query("SELECT * FROM Product LIMIT 100");
$products = $results->fetchAll();       // $products est un array

echo json_encode($products);       // json_encode transforme l'array en une string.
```

> Attention à n'ecrire qu'un seul et unique echo dans votre script. Sinon le format JSON ne sera pas respecté.