Web, HTTP, PHP côté serveur

Licence Informatique 3ème année

Alexandre Niveau — Jean-Marc Lecarpentier

Web, HTTP, PHP côté serveur

Notes de cours

- **Web, HTTP, PHP côté serveur**

Travail personnel

Objectifs

Prendre en main le serveur web fourni par l'université, et expérimenter avec l'exécution de PHP côté serveur.

Exercice 1 — Mise en place de l'environnement de travail : *Hello world*

> Cet exercice a déjà été fait dans le cadre du cours « Technologies web 1 » en L1, puis « Technologies web 3 » en L2. Il est principalement destiné à celles et ceux qui ne l'ont pas suivi. Pour les autres, il ne s'agit que de se rafraîchir la mémoire : n'y passez pas 10 minutes !

❦ Questions

1. Le département met à votre disposition un espace personnel sur le web. Son fonctionnement est expliqué sur cette page de la FAQ du département. Si vous n'avez pas encore lu cette page, faites-le (pas la peine de lire les « Informations complémentaires pour les plus avancé·es »).

2. Ouvrez votre éditeur de texte favori (si vous ne savez pas lequel choisir, prenez Geany, simple et *relativement* complet et léger).
3. Avec l'éditeur de texte, créez un fichier `hello.html` dans le répertoire `www-dev` de votre espace personnel sur le serveur (s'il y en a déjà un, ouvrez-le).
4. Écrivez n'importe quoi dedans, par exemple *Hello world*. Enregistrez.
5. Ouvrez Firefox (on n'utilisera que Firefox dans ce module). Si vous ne trouvez pas l'icône, cliquez sur « Activités » en haut à gauche et tapez « firefox » dans le champ texte. Avec Firefox, rendez vous à l'URL `https://dev-`*`LOGIN`*`.users.info.unicaen.fr/`, où *`LOGIN`* est votre login de connexion (par exemple `dupont227`). Vous devriez voir ce que vous avez écrit dans le fichier. Si ce n'est pas le cas, demandez de l'aide.
6. Créez un répertoire `test` dans `www-dev`. Vous devriez le voir apparaître dans le listing si vous actualisez la page `https://dev-`*`LOGIN`*`.users.info.unicaen.fr/` dans le navigateur. Observez bien comment la barre d'adresse du navigateur est modifiée si vous cliquez sur `test`, ou sur le fichier `hello.html`. Assurez-vous d'avoir compris que le serveur **sert le contenu de `www-dev` à la racine de votre site `https://dev-`*`LOGIN`*`.users.info.unicaen.fr`** : n'hésitez pas à demander des explications à votre chargé·e de TP si ce n'est pas clair pour vous !

Attention à bien organiser votre serveur web, et attention aussi à ne pas y mettre tous vos fichiers, mais seulement ceux qui ont des raisons d'y être !

Exercice 2 — Premiers pas avec PHP côté serveur

Cet exercice vise à vous familiariser avec les pages dynamiques et le fonctionnement de PHP. Pour rappel, une page PHP est un script, qu'il faut exécuter pour en voir le résultat. C'est **le serveur** qui exécute le script quand un client y accède *via* HTTP : si vous essayez d'ouvrir une page PHP locale avec votre navigateur, ça ne fonctionnera pas (le navigateur vous proposera probablement de la télécharger).

❦ Exécution d'un script par le serveur

1. Créez un script PHP `hello.php` contenant le code suivant :

   ```
   <?php
   echo "Hello\n";
   echo "<strong>\n";
   echo "World!\n";
   ```

   Exécutez-le dans un terminal : trois lignes doivent s'afficher.
2. Copiez le script à la racine de votre `www-dev`, et allez à l'URL

`https://dev-LOGIN.users.info.unicaen.fr/hello.php`. Qu'est-ce qui s'affiche ? Quelles sont les différences par rapport au terminal, et pourquoi ?

3. Affichez la source de la page (dans Firefox, clic-droit puis « Afficher la source », ou Ctrl-U). Que voyez-vous :
   - le contenu du fichier PHP, ou bien
   - les trois lignes qu'affichait le terminal ?

   ⇒ Que signifie le terme « source » ici ?
4. Le navigateur peut-il savoir que la page a été générée par un script PHP ?
5. Que se passe-t-il si vous faites une erreur dans le script ?
6. Que se passe-t-il si vous écrivez quelque chose (par exemple `TEST`) *avant* le `<?php` ? (Enlevez-le après avoir testé, sinon la suite ne marchera pas !)

❦ Accès au contexte HTTP depuis le script

Le serveur ne se contente pas d'exécuter le script PHP : il lui fournit diverses informations sur la requête HTTP, effectuée par le client, qui a conduit à cette exécution. D'autre part, le script ne se contente pas d'écrire le *corps* de la réponse HTTP : il peut demander au serveur de renvoyer tel champ d'en-tête, ou même tel code de statut.

0. Avant de faire la suite, **attention :** il ne doit rien y avoir dans le fichier avant la balise ouvrante de PHP (`<?php`), **même pas une espace ou un saut de ligne !** Vérifiez bien !
1. Ajoutez la ligne suivante juste avant le premier `echo` :

   ```
   http_response_code(404);
   ```

   Elle permet de changer le code de statut de la réponse HTTP.
2. Actualisez la page : vous ne devriez voir aucune différence ! En revanche, si vous actualisez après avoir ouvert l'onglet « Réseau » des outils développeur (raccourci Ctrl-Shift-E), vous devriez constater que la réponse envoyée est bien une 404.
3. Ajoutez la ligne suivante juste en-dessous de l'autre :

   ```
   header("Content-Type: text/plain");
   ```

   et actualisez la page. Que se passe-t-il, et pourquoi ? Cliquez sur la réponse HTTP dans l'onglet « Réseau », pour voir les en-têtes de la réponse : voyez-vous la différence avec ou sans cette ligne ?
4. PHP refuse de modifier l'en-tête de la réponse HTTP après avoir commencé à en écrire le corps. Vérifiez cela en lui faisant afficher quelque chose avant l'appel à `http_response_code` (et enlevez-le après avoir testé).
5. Avec une fonction de debug type `var_export`, affichez le résultat de la fonction `getallheaders()`, et comparez à ce que vous voyez dans l'onglet « Réseau ».

6. Avec une fonction de debug type `var_export`, affichez le contenu de la variable `$_GET` (attention à l'underscore) : il devrait être vide. Que se passe-t-il si vous modifiez la barre d'adresse afin d'aller à l'URL `https://dev-LOGIN.users.info.unicaen.fr/hello.php?toto=12&blabla=truc` ?
7. **Optionnel :** faire en sorte que la page s'affiche « en mode texte » si on passe dans l'URL la valeur `texte` au paramètre `mode`, et en « mode visuel » (HTML interprété) sinon.

Exercice 3 — Manipulation de fichiers, inclusion et redirection

Téléchargez l'archive de l'exercice, qui contient un fragment de HTML et un fragment de PHP, ainsi qu'un script `inclusion.php` qui utilise ces fragments.

❦ Inclusion avec `include`/`require`

1. Placez le contenu de l'archive sur votre serveur, regardez bien le contenu des scripts, observez le résultat. Assurez-vous de bien comprendre ce qui se passe.
2. Remplacez les trois `include` par des `require`, et regardez à nouveau le résultat. Quelle est la différence ?
3. Enlevez l'inclusion du fragment inexistant et vérifiez que cette fois la conclusion s'affiche bien.
4. Dupliquez le contenu du `body` (copiez le contenu, du début jusqu'à la conclusion, et collez-le après la conclusion). Chaque inclusion aura donc lieu deux fois. Que se passe-t-il, et pourquoi ?
5. Remplacez les quatre `require` par des `require_once`, et expliquez le résultat.

Il existe aussi `include_once`, mais il est moins utile. À moins d'avoir une bonne raison, il vous est recommandé d'utiliser `include` pour inclure un fragment de contenu (HTML ou PHP), et `require_once` pour inclure un fichier PHP contenant des déclarations (constantes, fonctions, classes…).

❦ Récupération du contenu d'un fichier dans une variable

Les instructions `include` et `require` exécutent immédiatement le code : elles sont en quelque sorte remplacées par le résultat de l'exécution. Il est souvent utile de récupérer ce résultat avant affichage, pour le modifier ou pour le passer à une fonction.

1. Dans un nouveau script PHP, récupérez avec la fonction `file_get_contents` le contenu du fragment HTML dans une variable `$frg` et l'afficher.
2. Faites de même avec le fragment PHP. Que se passe-t-il, et pourquoi ?
3. Pour récupérer le résultat de l'exécution du fragment PHP, il faut utiliser un `include`/`require` en activant l'*output buffering* (voir cours). Faites-le.

❦ Redirection et `header`

La fonction PHP `header` permet de manipuler les en-têtes HTTP de la réponse envoyée par le serveur.

1. Dans une nouvelle page PHP, mettez le contenu suivant :

   ```
   <?php
   header('X-Mon-Entete-Perso: lorem ipsum dolor sit amet');
   ?>
   ```

   Ouvrez l'onglet « Réseau » des outils développeur de Firefox (raccourci : control-shift-E), accédez à la page, et vérifiez que l'en-tête custom a bien été transmis. La spec de HTTP autorise tous les en-têtes commençant par `X-` — elle garantit qu'ils ne seront jamais utilisés dans une future version. Cela permet d'étendre HTTP pour des usages spécifiques à une application sans s'exposer à de futures incompatibilités.

2. La fonction `header` ajoute (ou modifie) des en-têtes HTTP, mais dans certains cas très particuliers elle va un peu plus loin.

   Mettez le contenu suivant dans une nouvelle page PHP :

   ```
   <?php
   header('Location: https://fr.wikipedia.org/wiki/Chien');
   ?>
   <p>Voilà du <strong>HTML</strong> qu'on n'aura pas le temps de voir !<
   ```

   Comprenez-vous ce qui se passe ? Observez le déroulement avec l'outil « Réseau » de Firefox, et vérifiez que `header` n'a pas simplement ajouté un en-tête, mais a aussi modifié le code de statut de la réponse HTTP.

Exercice 4 — L'automate cellulaire élémentaire sur le web

❦ Mise en place

1. Retrouvez le script `rule-110.php` fait au TP précédent. Peu importe, pour

l'instant, si vous avez terminé l'exercice : l'important est que le script affiche quelques générations de l'automate. Lancez-le dans un terminal pour vous assurer que c'est le cas.

2. Copiez le script sur votre serveur web, et accédez-y via HTTP dans un navigateur. Est-il bien exécuté ? Le résultat visuel est-il correct ? Si non, savez-vous pourquoi ?
3. Affichez la source de la page (dans Firefox, clic-droit puis « Afficher la source », ou Ctrl-U) : vous devriez retrouver l'affichage du terminal (sauf si vous avez utilisé le `PausingTerminalDisplayer` de la 2e partie de l'exercice : si c'est votre cas, comprenez-vous pourquoi ça ne marche pas ?).
4. Pour que le résultat visuel soit correct, une solution est d'utiliser l'élément HTML `pre`. Le faire.

❦ Retour au TP précédent

Attention : avant de continuer cet exercice, il faut d'abord **terminer l'exercice du TP précédent**. **Si vous êtes en difficulté** et que vous n'avez pas le temps d'aller plus loin, ce n'est pas bien grave.

❦ Une vue HTML

1. Si ce n'est déjà fait, faire en sorte que l'affichage « navigateur-compatible » soit géré par un `HtmlDisplayer` (même s'il ne s'agit que de rajouter un élément `pre` autour de la sortie terminal).
2. Faire en sorte que le code HTML de la page généré par le `HtmlDisplayer` soit valide HTML5. NB: il est interdit d'afficher des choses en-dehors du `HtmlDisplayer`, lui seul doit avoir la main sur l'affichage !

❦ Paramétrage de l'automate via l'URL

1. Faire en sorte que la taille du monde et le nombre de générations soient configurables via des paramètres d'URL `nbCells` et `nbGenerations`.
2. Ajouter un paramètre d'URL pour qu'on puisse choisir la règle d'évolution à utiliser.

❦ Un même script pour le terminal et le web

On peut remarquer que le cœur du programme, c'est-à-dire la représentation du monde et la gestion de l'évolution, n'a pas changé depuis qu'on a mis notre script sur le web. C'est normal, car il s'agit du **modèle**, de la **« logique métier »** du programme. Seuls l'affichage et la façon de récupérer les paramètres ont changé. On va rendre ça encore plus explicite en faisant en sorte que notre script puisse être utilisé à la fois

dans un terminal et sur le web, sans qu'on ait à le modifier.

La fonction `php_sapi_name()` renvoie le nom du SAPI (*Server Application Programming Interface*) utilisé. Si le script est exécuté dans un terminal, la fonction renvoie la chaîne `'cli'` (pour *Command Line Interface*) ; sinon, il y a un paquet de valeurs possibles, mais on pourra supposer pour simplifier que dans tous les autres cas le script est exécuté par un serveur web.

1. Écrire un script qui affiche « Vous êtes dans un terminal » lorsqu'on l'exécute en ligne de commande, et « Vous êtes sur le web » lorsqu'on y accède via HTTP.
2. Modifier `rule-110.php` pour que le script fonctionne aussi bien dans le terminal que sur le web. Pour les paramètres, dans le cas de l'exécution dans un terminal, on passera les arguments à la commande : dans PHP, la variable `$argv` est un tableau contenant chacun des arguments passés au programme (ainsi que le nom du script, dans la case d'indice 0).
3. Vérifiez auprès de votre chargé·e de TP que votre code est propre (bien factorisé notamment).