TP 2 de Programmation Web - Master 1 Informatique : Protocole HTTP, CGI et JavaScript

Auteur : Olivier GLÜCK, Université Lyon 1

Objectifs

- Etude du protocole HTTP dans le cadre de la programmation CGI
- Manipulation de formulaires HTML
- Apprentissage de la programmation Web en JavaScript

Pré-requis

Protocole HTTP, formulaires HTML, principes de la programmation CGI, JavaScript

NB

Ce TP se déroule sur 6h (moitié séance 2 + séance 3 + moitié séance 4). *Il est impératif de travailler de façon importante entre chaque séance. Pour ce faire, il est nécessaire de conserver vos fichiers d'une séance sur l'autre.*

1. Installation d'Apache

Dans ce TP, on vous demande d'installer et de configurer un serveur Apache sur votre poste de travail. Pour contacter votre serveur Web, vous utiliserez le navigateur `Mozilla` ou `Netscape` avec l'adresse `http://localhost/` ou `http://192.168.9.xx/`.

0- Pensez à faire la remise à zéro de la configuration des machines avant de commencer le TP…

1- Configurez `eth0` sur le réseau du miroir (192.168.9.9) en vous branchant sur le commutateur central ; vous prendrez l'adresse IP **192.168.9.xx** où **xx** est le dernier octet de l'adresse IP inscrite sur l'étiquette de votre machine. Vérifiez que vous arrivez à « pinger » le miroir .

2- Installation des packages

Pour ce TP, les packages qui nous intéressent sont :

```
apache          1.3.28-2        Versatile, high-performance HTTP server
apache-common   1.3.28-2        Support files for all Apache webserver
apache-doc      1.3.31-1        Apache webserver docs
```

Tapez les commandes suivantes pour installer apache ; vérifiez pour chaque commande qu'il n'y a pas de message d'erreur.

```
ping 192.168.9.9
export ftp_proxy=http://192.168.9.9:3128/
export http_proxy=http://192.168.9.9:3128/
rm -Rf /etc/apache /var/www;
apt-get update; apt-get remove --purge apache apache-doc apache-common;
apt-get install nom_pkg
```

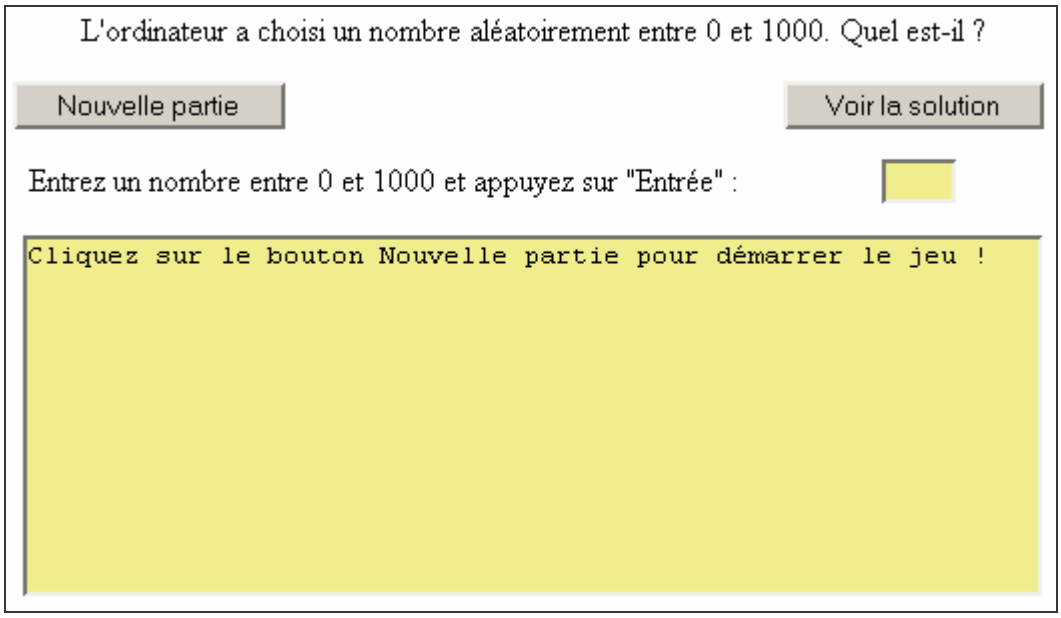
3- Pour vérifier que le package est correctement installé, faire un `dpkg -l` et vérifiez que la ligne correspondant au package commence par `i`

Des fichiers (comme par exemple les logos permettant la construction de votre site) sont disponibles sur le serveur de la salle TP. Vous pouvez les récupérer en montant **`192.168.9.9:/partage`** par NFS (en lecture seule).

A la fin de la séance, n'oubliez pas de sauvegarder vos fichiers pour la séance suivante, soit en utilisant une disquette ou clé USB, soit en copiant vos fichiers dans `/root` et en utilisant `restore`.

2. Formulaires et JavaScript

Vous allez réaliser un petit jeu en associant du code JavaScript à un formulaire.
Le principe est le suivant : l'ordinateur choisit un nombre caché entre 0 et 1000 ; le joueur doit trouver le nombre en un minimum de coups. Voici une copie d'écran de l'interface utilisateur :



Manipulation

Réalisez l'interface ci-dessus à l'aide d'un formulaire contenant deux boutons de type "`button`", une zone de saisie de type "`text`" et une zone d'affichage de type "`textarea`" en lecture seule (attribut `readonly`).

Manipulation

Réalisez maintenant le jeu en lui-même à l'aide d'un script JavaScript. Vous mettrez en œuvre les spécifications suivantes (les copies d'écran correspondantes se trouvent ci-après) :

- *Quand la page du jeu se charge, une boîte de dialogue demande au joueur de confirmer le fait qu'il souhaite commencer une partie (`figure 1`) ; s'il clique sur « Annuler », le joueur doit cliquer sur « Nouvelle partie » pour commencer un nouveau jeu (`figure 1a`) sinon la `figure 1b` s'affiche et le curseur est placé dans la zone de saisie du nombre.*
- *Quand l'utilisateur saisit un nombre, le script doit vérifier qu'il s'agit bien d'un nombre entre 0 et 1000 ; si tel n'est pas le cas un message d'alerte doit s'afficher (`figure 2`).*
- *Si le nombre saisi correspond au nombre recherché, un message d'alerte félicite le joueur et affiche son nombre de coups ; sinon le script affiche dans la zone "`textarea`" si le nombre saisi est trop petit ou trop grand ; le texte doit s'insérer dans la zone au-dessus des messages précédents ; la `figure 3` montre un exemple complet dans lequel le joueur a gagné en 7 coups.*
- *Si le joueur a gagné, le script doit lui proposer une nouvelle partie après qu'il ait cliqué sur « OK » pour fermer la boîte de dialogue le félicitant.*
- *La `figure 4` montre ce qui s'affiche quand le joueur demande la solution.*

Manipulation

S'il reste du temps à la fin de la séance, vous pourrez remplacer la zone d'affichage "`textarea`" par un bloc présent dans le document et dans lequel le script inscrit ses messages.

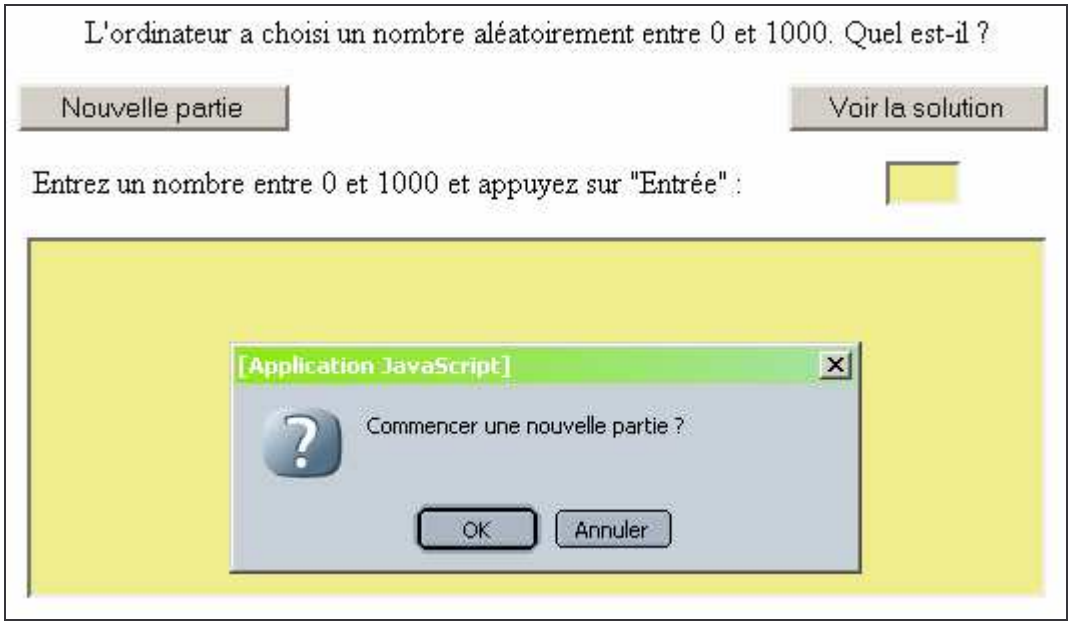


Figure 1 : au chargement

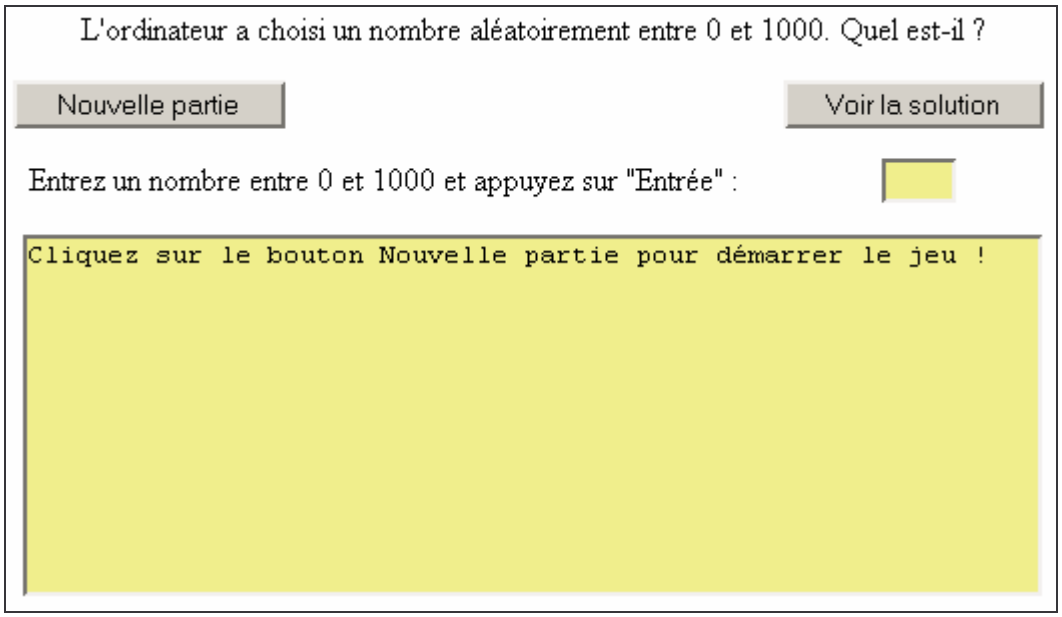


Figure 1a : le joueur a cliqué sur « Annuler »

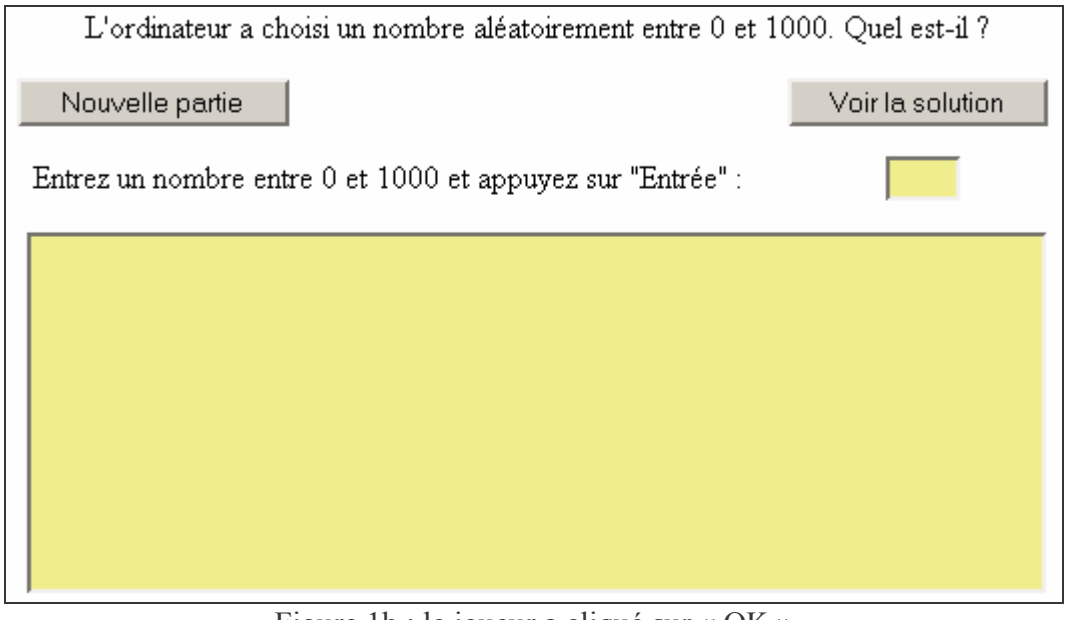


Figure 1b : le joueur a cliqué sur « OK »

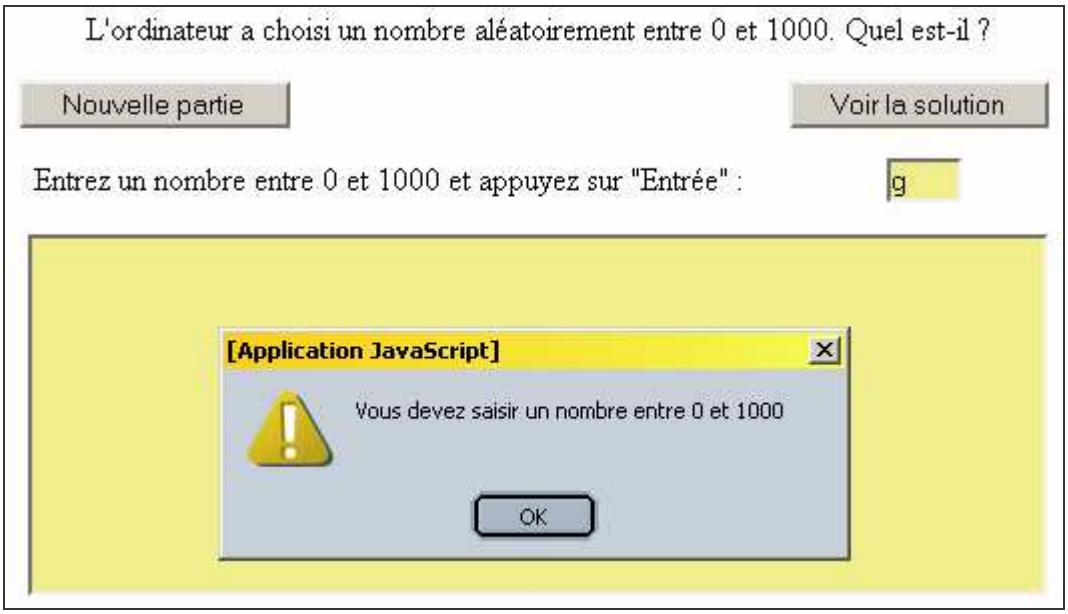


Figure 2 : erreur de saisie du nombre

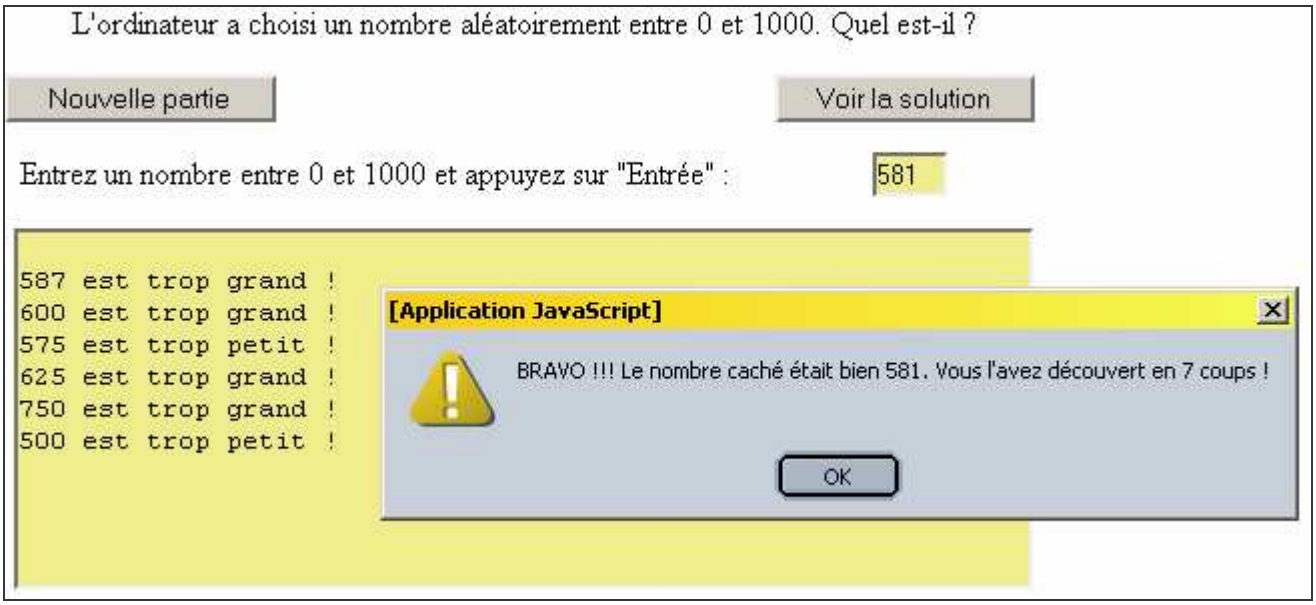


Figure 3 : le joueur a gagné en 7 coups

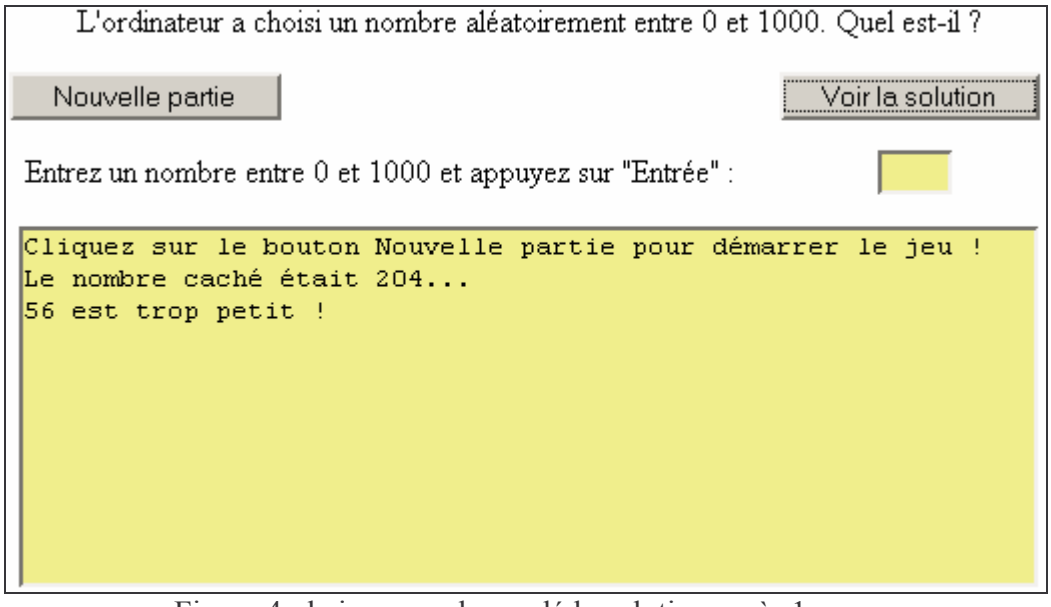


Figure 4 : le joueur a demandé la solution après 1 coup

3. HTTP, CGI et formulaires

Manipulation

Ré-installez le serveur Apache (cf. 1) proprement comme cela a été fait lors du TP1 et démarrez le.

Manipulation

Créez, en shell, un script CGI minimal appelé `Decode.cgi` que vous placerez dans le répertoire `/root/cgi-bin` et qui se contente de renvoyer au client la date et l'heure du serveur. Pour l'instant, il ne vous est pas demandé d'exécuter ce script via votre navigateur.

Question et manipulation

Comment, en utilisant la commande `telnet`, vérifier que le serveur HTTP accepte les exécutions de CGI ? Simulez une telle exécution, avec la méthode `GET` puis `POST`. Si l'exécution de CGI ne fonctionne pas sur votre serveur, quel est le type de l'erreur ? Modifiez alors la configuration du serveur pour que l'exécution des CGI présents dans `/root/cgi-bin` soit possible. Vous pourrez vous aider de la documentation Apache disponible sur votre machine. En cas d'échec, n'oubliez pas de consulter les logs du serveur pour en savoir un peu plus sur les raisons de l'échec. Une fois que cela semble fonctionner, essayez maintenant de faire la requête via votre navigateur web. Cela fonctionne t-il également ?

Question et manipulation

Quelle est l'identité de l'utilisateur exécutant le script CGI ? Utilisez les commandes `id` et `whoami` pour le vérifier. Où cela se configure t-il ?

Manipulation

Ecrivez un formulaire contenant un menu déroulant à sélection simple, un bouton radio proposant quatre choix (`add`, `mult`, `div` et `sub`) et un bouton de soumission provoquant l'exécution de `Decode.cgi`. Le menu déroulant doit permettre de sélectionner un chiffre entre 0 et 9.

Manipulation

Modifiez `Decode.cgi` pour qu'il affiche, soit le contenu de la variable d'environnement `QUERY_STRING` si l'utilisateur a choisi la méthode `GET`, soit son équivalent si l'utilisateur a choisi la méthode `POST`. La chaîne CGI affichée correspond t-elle aux données saisies dans le formulaire ?

Question et manipulation

Utilisez `ethereal` pour visualiser les échanges Client/Serveur ayant lieu entre le moment où l'utilisateur clique sur le bouton de soumission et l'affichage de la réponse. Combien de messages sont échangés ? Où se trouvent les données du formulaire ? Vous ferez une capture en utilisant la méthode `GET` et une autre en utilisant la méthode `POST`.

Question et manipulation

Utilisez l'en-tête `Location:` pour provoquer la redirection de la réponse CGI vers `http://localhost/` et examinez les échanges avec `ethereal`. A quoi ressemble la réponse du serveur ?

Manipulation

Supprimez la redirection. On souhaite maintenant que le script `Decode.cgi` crée, pour chaque couple (`NOM`, `VALEUR`) issu des champs du formulaire, une variable de nom `NOM` dont la valeur est `VALEUR`, à partir des données présentes dans la chaîne CGI. Le script devra désormais afficher chaque couple sous la forme `NOM=VALEUR`. Vous penserez à url-décoder (en partie seulement) les champs `VALEUR` en transformant chaque caractère + en caractère espace.

Manipulation

On voudrait que `Decode.cgi` soit appelé lors de n'importe quelle exécution de CGI écrit en shell mais qu'il permette l'exécution d'un autre CGI/shell. L'idée est de faire en sorte que n'importe quelle soumission de formulaire provoque l'exécution de `Decode.cgi` pour positionner correctement les variables shell (cf. question précédente) puis que `Decode.cgi` lance l'exécution d'un script fils réalisant le traitement effectivement souhaité. Le chemin absolu du deuxième script (réalisant le traitement) sera placé dans un bouton caché du formulaire nommé `SCRIPT_PATH`. Modifiez `Decode.cgi` pour permettre ce fonctionnement. Vous pourrez utiliser la commande `sed 's/%2F/\//g'` pour transformer les `%2F` en `/` dans la variable `SCRIPT_PATH`.

Manipulation

*Modifier votre formulaire pour qu'il exécute le script `/root/scripts/table.sh` après `Decode.cgi`. Ecrivez le script `table.sh` permettant d'afficher la table d'opérations associée au bouton radio sélectionné et correspondant au chiffre choisi. Par exemple, si l'utilisateur choisit le chiffre `3` et l'opération `mult`, un click sur le bouton de soumission doit afficher 3*0=0, 3*1=3, …*

Manipulation

S'il reste du temps à la fin du TP, vous pourrez :

- ✓ *reprendre la partie 2 du TP1 (configuration d'apache) si vous ne l'aviez pas terminée ;*
- ✓ *écrire un formulaire qui demande le chemin d'un répertoire ; quand le formulaire est soumis, un programme CGI écrit en shell doit afficher le contenu du répertoire ; le programme affichera de manière différente les répertoires (par ex. en rouge et en gras), les fichiers `.cgi` (en bleu), les fichiers `.sh` (en vert),… Pouvez-vous ainsi voir le contenu de `/etc/apache` ?*
- ✓ *rajouter un bouton permettant de proposer l'affichage de la table d'une autre opération.*