Cours INE11 @ ENSTA, Projet PHP & JavaScript

Benjamin Canou (benjamin@ocamlpro.com)

April 5, 2018

Part I
Partie serveur en PHP (séance 1 & 2)

Dans cette première partie, vous allez implanter la partie serveur d'un petit jeu d'aventure. Vous la compléterez par une partie client en JavaScript par la suite. L'application Web client-serveur résultante sera à rendre à l'issue de ces trois séances et constituera l'unique évaluation de cette partie du cours.

1 Dire bonjour

1.1 À tout le monde

Pour commencer votre séance de travail :

1. créez un répertoire `public_html` à la racine de votre dossier personnel s'il n'existe pas déjà.
2. Éditez-y un fichier `hello.php` qui contiendra un programme PHP minimal affichant simplement `bonjour`.
3. Lancez votre navigateur à l'URL `http://votrelogin.ensta.fr/hello.php` (les URLs données par la suite seront des suffixes de cette racine).
4. Corrigez le programme pour qu'il produise une sortie HTML minimale valide (et non juste le texte `bonjour`), si ce n'est pas déjà le cas.

Afin de terminer ce mini projet à la maison, sous GNU/Linux vous pouvez installer le serveur Apache, son module pour PHP et activer son option `userdir`. Sous windows, vous pouvez installer EasyPHP.

1.2 À une personne donnée dans l'URL

1. Faites en sorte que le programme dise bonjour à une personne dont le nom est fourni dans le paramètre `$_GET["name"]`.
2. Testez votre code en faisant pointer votre navigateur sur `/hello.php?name=madame`.

1.3 À un utilisateur

Ouvrez un nouveau fichier `game.php`.

Nous allons mettre en place une micro gestion d'utilisateurs et en profiter pour implanter la structure principale de notre application.

Le mécanisme est le suivant : lors de la connexion, le script enregistre le nom du joueur (passé via un formulaire) dans une variable de session `$_SESSION["username"]`. Ensuite pour vérifier qu'un utilisateur est bien connecté il suffira de tester si cette variable existe. Finalement, pour la déconnexion, il suffira de détruire cette variable de session.

Une fois le login effectué, nous allons avoir besoin de notre fichier PHP réagisse de façon appropriée aux différentes actions possible au cours du jeu (que nous appelerons commandes). Pour ceci, nous utiliserons un paramètre GET `command` qui pourra prendre ses valeurs dans un ensemble fini (`"login"`, `"logout"`, `"go"`, `"take"`, `"put"`, etc.), chacune associée à une fonction de traitement associée. Une des toutes premières actions réalisées par le script PHP sera de tester ce paramètre `command` afin de brancher sur la bonne fonction. Chaque commande pourra utiliser des paramètres GET supplémentaires qui lui sont propres, qui seront alors passés à la fonction de traitement. Nous allons implanter les commandes `login` et `logout`.

1. Écrivez le point d'entrée principal du programme. If faut en premier tester si `$_SESSION["username"]` est définie. Dans ce cas, cela signifie que le login a déjà été entré.

   - Si `$_SESSION["username"]` est défini, brancher sur la bonne fonction en fonction du contenu de `$_GET["command"]`.

     Pour l'instant, faites en sorte d'appeler `logout_command()` si `$_GET["command"]` vaut `"logout"`,

     `default_logged_command()` si `$_GET["command"]` n'est pas définie et `unknown_command()` sinon.
   - Sinon, regarder la variable `$_GET["command"]`. Si celle-ci vaut `"login"`, appeler la fonction `login_command()`. Sinon, appeler la fonction `default_unlogged_command()`.

2. Écrivez la fonction `default_unlogged_command()` qui affiche un formulaire avec un champ `"name"` pour le nom d'utilisateur et un champ caché `"command"` valant `"login"`.

3. Écrivez la fonction `login_command()` qui utilise le paramètre `$_GET["name"]` pour affecter la variable de session `$_SESSION["username"]`, puis appelle la fonction `default_logged_command()`.

4. Écrivez la fonction `default_logged_command()` qui dit bonjour à l'utilisateur, et affiche un lien vers

   `/game.php?command=logout`.

5. Écrivez la fonction `unknown_command()` qui affiche une erreur.

6. Et enfin, écrivez la commande de `logout()` qui détruit les données de session et affiche un message d'adieu et le formulaire de login.

Puis vérifiez que tout fonctionne comme prévu.

Les pages de votre application auront probablement une partie commune (titre, lien de logout, etc.). Vous pouvez factoriser cette partie dans une ou plusieurs fonction(s) appelée(s) par toutes les pages.

2 Un petit jeu d'aventure : le monde

Nous allons écrire un mini jeu d'aventure où un héros se déplace de pièce en pièce dans un monde constitué de salles.

2.1 Définition de la carte

La structure du monde sera stocké dans un tableau global `$world`. Les indices du tableau sont des numéros identifiant de façon unique chaque pièce. Les valeurs sont des tableaux associatifs comportant un champ `"name"` et, pour l'instant, un champ `"outs"`. Ce dernier est un tableau d'associations dont les clefs sont les noms des sorties de la pièce courante et les valeurs les indices des pièces de destination. Un exemple de tel tableau est donné ci-après.

```
$world = array(
  array("name" => "chambre Jaune",
        "outs" => array ("porte" => 1,
                         "fenêtre" => 2)),
  array("name" => "chambre Verte",
        "outs" => array ("porte" => 0,
                         "fenêtre" => 2)),
  array("name" => "jardin",
        "outs" => array ("fenêtre Jaune" => 0,
                         "fenêtre verte" => 1))
)
```

2.2 Victoire et défaite

Certaines pièces peuvent avoir un champ optionnel `win` et un autre `lose`. S'il est présent, il s'agit d'une chaîne contenant un message à afficher lors de l'entrée dans la pièce, justifiant la victoire ou la défaite.

2.3 Commandes de navigation

1. Ajoutez une variable de session `$_SESSION["room"]` contenant l'indice de la pièce dans laquelle se trouve le héros. Par convention, un nouveau joueur démarre dans la pièce 0.

2. Reprogrammez la commande par défaut pour afficher la pièce courante. En cas de victoire ou d'échec, il conviendra d'afficher le message associé.

3. Implantez une commande `"go"` prenant un paramètre donnant l'indice de la pièce dans laquelle se déplacer. On ne se préoccupera pas des tricheurs et il est donc inutile de vérifier que la pièce est bien atteignable depuis la pièce courante. Après avoir effectué le déplacement, il va de soi que la commande par défaut doit aussi être effectuée.

4. Étendez la commande par défaut pour afficher une liste HTML de liens vers les salles atteignables depuis la pièce courante sous le format : «aller dans 'la cour' par 'la porte cassée'». Pensez à la boucle `foreach`.

3 Un petit jeu d'aventure : l'inventaire

Dans un tel jeu, il est d'usage de collecter et utiliser des objets. Nous allons commencer par implanter la première partie.

3.1 Inventaire

1. Ajoutez un tableau global `$objects` associant un numéro unique à chaque type d'objet présent dans le jeu à son nom.
2. Ajoutez une variable de session `$_SESSION["inventory"]` qui est un tableau associant les identifiants des objets présents dans le sac de notre aventurier au nombre qu'il en possède.
3. Affichez cet inventaire dans la commande par défaut.

3.2 Récupération d'objets

Les salles peuvent contenir des objets que le personnage peut prendre.

1. Ces objets sont stockés dans le champ `"stuff"` de chaque pièce, sous le même format que l'inventaire du héros. Enrichissez un peu votre monde en conséquence.
2. Ajoutez une commande `"take"` ayant pour paramètre le numéro identifiant de l'objet à prendre. Cette commande ajoute une unité de cet objet au sac du héros.
3. Affichez la liste des liens vers les commandes de récupération d'objet possibles.

3.3 Suppression et dépôt d'objets

Pour que la commande `"take"` soit correcte, il faut qu'elle supprime l'objet de la carte au moment de l'ajouter à l'inventaire. Cette commande n'est pas possible directement si vous avez suivi les consignes à la lettre, pourquoi ? Trouvez une solution que vous validerez avec votre enseignant avant de l'implanter.

4 Un petit jeu d'aventure : conditions

Pour l'instant, ce jeu n'est pas très difficile. Nous allons corser un peu les choses en ajoutant des pré-requis aux commandes. Pour l'instant, un pré-requis sera une liste d'objets à posséder, sous la forme d'un tableau d'entiers correspondants aux identifiants des objets nécessaires.

Les pré-requis seront attachés aux sorties des pièces par un nouveau champ `"condition"`, et de même pour les objets.

1. Éditez votre monde pour le rendre un peu plus difficile. Vous pourrez par exemple ajouter une clef nécessaire à l'ouverture d'une porte. Pour tester les conditions sur les objets vous pourrez établir que la clef est bouillante, et que votre héros doit donc se munir de gants pour la saisir.
2. Éditez les commandes `"go"` et `"take"` pour prendre en compte les conditions.
3. Lorsqu'une tâche est impossible, affichez la liste les objets manquants pour la mener à bien.

5 Pour aller plus loin

Voici quelques idées pour approfondir :

- Ajouter du formatage et des images associées aux pièces ou aux objets
- Ajouter des messages associés aux actions, des personnages, dialogues etc.

- Implanter des adversaires et des combats
- Améliorer la gestion des utilisateurs, par exemple en stockant l'état sur disque plutôt que dans la session, afin de ne pas tout perdre lors de la déconnexion.
- Définir une syntaxe concrète pour définir des mondes et charger le monde depuis un fichier au démarrage du script

Part II
Partie client en JavaScript (séance 3)

Nous allons maintenant améliorer le jeu d'aventure des séances précédentes en lui ajoutant un affichage de la carte en JavaScript. Pour ceci, nous utiliserons l'objet `XMLHttpRequest` pour effectuer les requêtes et l'élément HTML canvas pour dessiner à l'écran.

`http://developer.mozilla.org/en/docs/HTML/Canvas`

6 Export des données géographiques en JSON

1. Étendez la variable `$world` pour ajouter pour chaque pièce deux données visuelles : une couleur (au format CSS) et une liste de points aux angles des murs de la pièce dans le carré de 300x300.

2. Écrivez une fonction `world_to_json` affichant la carte dans un format lisible par JavaScript, par exemple :

```
[ { "color": "red",
    "points": [[10, 10], [10, 20], [20, 20], [20, 10]] },
  { "color": "green",
    "points": [[10, 20], [10, 30], [20, 30], [20, 20]] }, ... ]
```

3. Ajoutez une commande `map` renvoyant au client le JSON fabriqué par la fonction précédente. Faites attention à bien renvoyer un en-tête HTTP `Content-type` approprié pour JSON ave la fonction PHP `header`, et à ne pas émettre de HTML parasite.

7 Initialisation et remplissage du canvas

Suivez le lien donné en en-tête pour répondre à cette section.

1. Ajoutez un élément `<canvas>` aux pages générés par votre code PHP de taille 300x300 pour afficher la carte. Cette carte doit bien sûr être affichée une fois le login effectué uniquement.

2. En utilisant l'API canvas, remplissez le fond de carte de gris clair après le chargement de la page.

3. Complétez le code de dessin pour obtenir la carte via une requête HTTP et l'afficher dans le canvas.

8 Position du héros

1. Ajoutez au monde la position du centre de chaque pièce. Vous pouvez aussi la calculer automatiquement à partir des coordonnées des murs.
2. Exportez-la au client via une commande `where` répondant la position courante du héros au format JSON `{"x":number,"y":number}`.
3. Affichez un petit bonhomme à l'emplacement du héros sur la carte.

9 Pour aller plus loin

Voici quelques idées pour approfondir :

1. Faites en sorte de ne plus avoir à changer de page pour attraper un objet en remplaçant les liens par des requêtes HTTP effectuées en JavaScript et des mises-à-jour locales de la page.
2. Faites de-même pour le changement de salle.
3. Ajoutez une visualisation graphique des autres éléments du jeu.
4. etc.