# R3.01 - ProgWeb
# TD 1 - Initiation

## Introduction

Dans certains exercices vous verrez apparaître :

- 🚫→ Vous ne devez pas utiliser ces mots-clés ou ces fonctions !
- 🛠️→ Fonctions ou mots-clés suggérés. Consultez le site **php.net**. Pas d'obligation de les utiliser si vous (et pas Google) trouvez une autre façon de faire.

## Exercice 1 - Script autonome

Vous devez vous rappeler avoir appris en R1.04 à écrire quelques scripts d'administration système. Nous avions utilisé Bash mais aussi PHP qui, même s'il est connoté "Web", est parfaitement adapté pour réaliser de tels scripts d'automatisation et d'administration.

Nous n'allons pas vraiment utiliser PHP pour écrire de tels scripts autonomes en R3.01, mais il est bien de rappeler que PHP peut aussi être utilisé dans ce cadre.

Voici donc un exercice pour vous remettre les idées au clair sur ce mode.

### Question 1

Pour qu'un script quelconque puisse être autonome et exécutable en ligne de commande, comme on le fait avec d'autres commandes système, il lui faut deux choses :

- **Une ligne spéciale tout en haut du script**
  Comment la nomme-t-on ?
  Quel est son rôle ?
  Donner cette ligne dans le cas de PHP
- **Des droits particuliers sur le script**
  Quels sont-ils ?
  Comment les positionne-t-on ?

### Question 2

Écrivez un script autonome qui affiche ceci à l'écran (rappel : l'instruction PHP **echo** affiche ce qui est placé derrière elle). Testez-le dans votre Terminal, en prenant soin d'appliquer les exigences de la **Question 1**, bien entendu !

Note : proposez 2 façons de produire ce résultat.

```
Bonjour tout le monde.
Une valeur approchée de PI vaut 3.1428
```

La valeur **3.1428** doit être calculée dans le script par la division de **22** par **7**. Vous aurez sans doute plus de chiffres derrière la virgule mais ce n'est pas important, vous pouvez les laisser s'afficher.

# Pages Web

A partir de maintenant, nous n'allons plus écrire que des scripts pour le Web.

Il faudrait donc un serveur Web pour servir des pages Web, mais pour du test on peut utiliser le serveur Web intégré au programme **php**. C'est ainsi que vous procéderez pour les TD/TP :

- Dans un Terminal, placez-vous dans le dossier contenant vos scripts
- Lancez l'interpréteur php en mode "Serveur Web" (**-S**) de la manière suivante :

  **php -S localhost:8888**

- Ouvrez un navigateur Web et entrez l'URL (adapter **<nom de votre script>**) :

  **http://localhost:8888/<nom de votre script>**

# Exercice 2 - Structures de contrôle

Vous avez à disposition, sur Moodle, un fichier PHP **`depts-exo2.php`** (ne vous trompez pas de fichier) qui contient la déclaration et l'initialisation d'un tableau nommé **`$depts`** composé de la liste des départements de France métropolitaine.

## Question 1

Dupliquez ce fichier **`depts-exo2.php`** en un script **`exo2-ul.php`** et complétez-le pour afficher les départements sous forme d'une liste HTML (**`<ul>`**).

🚫 **`foreach()`**

🛠️ **`count(), sizeof(), htmlentities()`**

## Question 2

Dupliquez votre script **`exo2-ul.php`** en **`exo2-table.php`** et modifiez votre code pour afficher cette liste dans une table HTML (**`<table>`**) avec une colonne pour le numéro de département et une autre colonne pour le nom. Prévoyez aussi des entêtes de colonnes.

Êtes-vous satisfait·e du résultat ?

Pourquoi ?

## Question 3

Comme vous n'êtes pas entièrement satisfait·e du résultat de la **Question 2**, refaites le même exercice mais en utilisant le fichier **`depts-fix.php`** comme point de départ et en adaptant le code en fonction de la nouvelle définition du tableau **`$depts`**.

Profitez-en pour afficher aussi le nom de la préfecture dans une 3ème colonne.

🛠️ **`foreach()`**

## Question 4

On souhaite maintenant ajouter le nom de la région, en information complémentaire entre **`()`**, juste derrière le nom du département.

Pour ce faire, vous disposez d'un fichier **`regions.php`** contenant un tableau **`$regions`**, qui associe le nom d'une région avec la liste (tableau) des codes des départements qui la constitue.

Sur la base de la **Question 3**, dupliquez votre script en un **`exo2-regions.php`**.

Cette fois-ci vous devez :

- Inclure, à l'exécution, le contenu du fichier **`regions.php`**.

- Ecrire une fonction **`trouve_region($code)`** qui renvoie le nom de la région du département dont on passe le **code** en paramètre
- Adaptez ensuite votre code pour afficher le nom de la région entre **`()`** pour chaque département.

🚫 de coller le contenu de **`regions.php`** dans votre script

🛠️ **`require_once(), in_array(), foreach(), global`**

# Exercice 3 - Paramètres d'URL

Vous savez certainement qu'une URL peut contenir des paramètres.

Si vous ne le saviez pas, voici un exemple :

**`http://localhost:8888/ajout_panier.php?produit=XYZ&qte=20`**

Explication des parties de cette URL :

- **`http`** : protocole utilisé. En local du **HTTP** c'est OK, sur un serveur public ce sera certainement du **HTTPS**.
- **`localhost:8888`** : adresse du serveur + éventuellement le port utilisé, ici **`8888`**. On verra ça en détail en R3.06
  Sur un serveur public le port sera sans doute absent. Ce sera alors le port par défaut en fonction du protocole : **`80`** pour **HTTP**, c'est de plus en plus rare, et **`443`** pour du **HTTPS**.
- **`ajout_panier.php`** : nom du script PHP qui construit la page Web
- **`?`** : séparateur indiquant que ce qui suit ce sont les paramètres du script.
- **`produit=XYZ`** : un paramètre (**`produit`**) et sa valeur (**`XYZ`**)
- **`&`** : séparateur entre chaque paire (nom, valeur) de paramètres

Côté serveur, le script (ici **`ajout_panier.php`**) reçoit ses paramètres dans la superglobale (voir cette notion à la fin du CM) qui s'appelle **`$_GET`**. C'est un tableau.

## Question 1

Créez un script **`exo3.php`** qui affiche le contenu du tableau **`$_GET`**, et testez avec l' URL :

**`http://localhost:8888/exo3.php?nom=ochon&prenom=paul`**

🛠️ **`_GET, print_r(), <pre>...</pre>`**

## Question 2

Créez un script **`exo3-regions.php`** qui affiche la liste des régions (juste les noms) issues du tableau **`$regions`** du fichier **`regions.php`**, mais en respectant les contraintes suivantes :

- Utilisez une liste **`<ul>`**
- Paginez votre affichage en n'affichant que 5 lignes à la fois
- Récupérez le numéro de page dans un paramètre d'URL nommé **`page`**
- Placez deux liens (**`<a href...>`**) sous la liste permettant de se déplacer vers la page précédente et la page suivante. Adaptez l'URL de chaque lien en fonction du numéro de la page courante.
- Si on est sur la page 1, ne pas afficher le lien "page précédente"
- Si on est sur la dernière page, ne pas afficher le lien "page suivante"

🛠️ **`require_once(), $_GET, array_slice(), count()`**

## Question 3

Dupliquez le script **`exo3-regions.php`** en **`exo3-regions-q3.php`** et modifier-le en rendant cliquables les noms de régions (**`<a href...>`**) et en envoyant ces liens vers un nouveau script nommé **`exo3-regions-detail.php`**.

Le nouveau script doit :

- Afficher le nom de la région dans un tag **`<h1>`**
- Afficher en dessous la liste (**`<ul>`**) des départements appartenant à cette région
- Avoir un lien permettant de retourner sur la page des régions

🛠️ **`urlencode()`**

# Exercice 4 - Sérialisation

## Introduction à la sérialisation

La sérialisation permet de convertir une variable PHP quelle qu'elle soit (de n'importe quel type scalaire, tableau ou objet) en une chaîne de caractères qui en décrit son contenu à l'aide d'une syntaxe spéciale. Cette variable peut donc être très complexe comme, par exemple, un tableau multi-dimensionnel. C'est d'ailleurs plutôt pour ce genre de "structures" complexes que la sérialisation est intéressante.

La sérialisation va de pair avec la désérialisation qui fait l'opération inverse, à savoir recréer en mémoire une variable à partir de sa représentation fournie sous forme de chaîne (issue d'une précédente sérialisation évidemment).

Un intérêt majeur de la sérialisation est qu'une chaîne de caractères est facilement stockable dans un fichier ou transportable par le réseau Internet.

Ainsi, on peut sérialiser des données (des variables) en vue de les restaurer (par désérialisation) plus tard et/ou ailleurs.

Attention, certains types de données ne sont pas sérialisables, comme les ressources, qui sont par exemple des handlers associés à l'ouverture d'un fichier. Ca ne sert à rien de pouvoir les sérialiser car ces variables n'ont d'existence que le temps que le fichier est

ouvert et uniquement sur la machine et associé au processus qui a fait cette action d'ouverture du fichier.

Pour info : ce mécanisme de sérialisation n'est pas propre à PHP. Il existe aussi dans d'autres langages, mais le format qu'utilise PHP lui est spécifique et non interopérable avec d'autres langages. Le format **JSON** (ou **XML**) peut être une option intéressante si on souhaite de l'interopérabilité.

La littérature informatique parle aussi de linéarisation et délinéarisation. Nous resterons avec les termes de sérialisation et désérialisation, anglicismes plus couramment employés.

## Fonctions de sérialisation / désérialisation PHP

Pour ce faire, PHP dispose de deux fonctions :

- **`serialize(<var>)`** retourne une représentation de **`<var>`** sous forme de chaîne
- **`unserialize(<chaine>)`** retourne une variable créée (type et valeur) à partir d'une représentation textuelle donnée par l'argument **`<chaine>`**

## Question 1

Créez un script **`exo4-q1.php`** qui inclut (**`require_once`**) le fichier **`serial.php`**.

Regardez le contenu de ce fichier, il déclare et initialise 5 variables (**`$ser_var1`** à **`$ser_var5`**)

À l'aide de la fonction **`serialize()`**, affichez à l'écran la chaîne résultant de la sérialisation de chacune de ces variables. Faites-les une par une (donc en 5 exécutions séparées) sinon vous ne verrez pas bien le résultat affiché.

Essayez d'identifier comment est constituée la sérialisation (comment est structurée la chaîne retournée).

## Question 2

Opération inverse, créez un script **`exo4-q2.php`** qui va passer en paramètre à la fonction **`unserialize()`** le contenu lu dans le fichier **`data`**. Affectez la valeur retournée par **`unserialize()`** à une variable.

Enfin, utilisez **`print_r()`** ou **`var_dump()`** pour afficher le contenu de cette variable.

Si tout va bien, vous venez de redonner vie à un tableau !

🚫 aucun **`require(), include()`** ou dérivé

🛠️ **`file_get_contents()`**

## Question 3

Sur la base de la **Question 2**, après avoir désérialisé le tableau, ajoutez-y l'artiste suivante pour l'année 2014 :

- Artiste : Christine and the Queens
- Album : Chaleur Humaine
- Ventes : 850000

Vérifiez cet ajout au tableau à l'aide d'un **`print_r()`**.

Si tout est satisfaisant, ajoutez la portion de code permettant de sérialiser votre tableau modifié et d'écrire le résultat de la sérialisation dans un fichier **`data.new`**.

Regardez le contenu de ce fichier pour vérifier que vous y trouvez bien vos petits...

🚫 aucun **`require(), include()`** ou dérivé

🛠️ **`file_get_contents(), file_put_contents()`**

# 🚨 Important pour la sécurité 🔥☣️‼️

Dans l'exercice précédent, les fichiers **`data`** et **`data.new`** sont stockés au même endroit que les scripts.

Pour les besoins de ce TD, c'est acceptable mais **ce n'est pas à reproduire dans un environnement de production**. En effet, ces fichiers sont directement accessibles depuis le Web en utilisant, dans le cas présent, l'URL suivante :

**`http://localhost:8888/data`**

Testez-le.

## Solution

En production, il conviendra de **TOUJOURS** placer les fichiers sensibles dans un dossier inaccessible par une URL mais qui sera toujours accessible depuis un script PHP puisqu'il s'exécute sur le serveur, sans restriction, ou presque.

Un dossier dans **`..`** (ou tout autre dossier placé à un niveau au-dessus de la racine du site) sera une excellente idée.

Les **`require_once()`** et autres **`file_get_contents()`** devront alors utiliser un chemin comme **`../db/data`** par exemple.

Note : il faut une configuration adaptée sur un vrai serveur Web pour que ça fonctionne. Avec notre mini serveur lancé avec **`php -S`**, ça ne marchera pas, donc pas la peine de tester. Mais gardez cela quand même en tête !!!

# Exercice Bonus

## Question 1

A partir des fichiers **`regions.php`** et **`depts-fix.php`**, créez un script qui présente les informations sous la forme suivante, en respectant les alignements, couleurs, police (Courier), taille de police. Exemple pour la Bretagne.

| | | | |
|---|---|---|---|
| **Bretagne** | Ille-et-Vilaine | 35 | Rennes |
| | Côtes d'Armor | 22 | St-Brieuc |
| | Finistère | 29 | Quimper |
| | Morbihan | 56 | Lorient |

Note : la table doit afficher les régions en les triant par taille décroissante de nombre de départements qu'elles contiennent et par ordre alphabétique de nom pour les régions ayant le même nombre de départements.

Au sein d'une région, les départements doivent être triés par la longueur de leur nom (ordre croissant) et par leur numéro de département pour les longueurs identiques.

Vous mettrez une alternance de couleurs sur les lignes des départements.

A vous de chercher sur **php.net** (jouez le jeu, n'allez pas sur Google) les fonctions utiles.

## Question 2

Rendez les en-têtes des 3 dernières colonnes cliquables (par des liens vers une URL avec un paramètre adapté) pour trier par ordre alphabétique ou par ordre numérique, croissant ou décroissant (une bascule), la colonne cliquée.