le cnam
Grand Est

Développement web (2)

Sites dynamiques
et développement côté serveur

NFA017 (4 ECTS)

Séance 05
Production de documents avec PHP
2022

Le Cnam en Grand Est - Centre régional
4 avenue du Docteur Heydenreich CS 65228 F 54052 Nancy Cedex
tél +33 (0)3 83 85 49 00 www.cnam-grandest.fr

Ce support pédagogique a été rédigé par Nils Schaefer et Simon MAHIEUX, enseignants au Cnam Grand-Est.

Table des matières

1. La bibliothèque FPDF

La bibliothèque FPDF permet de générer des documents PDF dynamiques à partir d'un code en PHP. Il est donc possible de produire des documents à la volée à partir de données saisies par les visiteurs ou à partir de données en provenance d'une base de données. De plus, cette bibliothèque a pour avantage d'être gratuite, légère et simple à installer. Consultez le site officiel (www.fpdf.org) qui est complet et efficace tout en restant sobre.

1.1 L'Installation

L'installation de FPDF (*Free PDF*) est très simple, il suffit de récupérer l'archive qui contient les fichiers de la bibliothèque.

Dans sa petite taille de moins de 200 Ko, l'archive contient le code de la bibliothèque, la documentation de chacune des méthodes ainsi qu'un tutoriel !

Seuls le fichier *fpdf.php* et le répertoire *font* sont nécessaires sur le serveur Web. Ils peuvent être placés n'importe où, il n'y a pas de contrainte.

Pour pouvoir utiliser FPDF, il suffit de placer ces deux lignes de code PHP avant de commencer à créer un document PDF dans un script PHP.

```
define('FPDF_FONTPATH','font/');
require('fpdf.php');
```

Notez bien qu'il peut être nécessaire de modifier les chemins si la page PHP n'est pas située dans le même répertoire que le fichier fpdf.php et le répertoire font.

La première ligne n'est pas nécessaire si le répertoire font est présent au même endroit que le fichier fpdf.php. Dans la pratique, c'est souvent le cas, il suffit alors simplement de placer cette unique ligne au début de votre scripts PHP.

```
require('fpdf.php');
```

1.2 Un document basique

Dans un premier temps, nous allons voir les méthodes nécessaires pour créer un document sommaire avec une seule ligne de texte. Pour cela il faut appeler le constructeur FPDF qui permet de créer l'objet qui représente le document PDF.

1.2.1 Le constructeur

Le constructeur accepte des paramètres relatifs à la présentation et à la gestion du document. Voici la syntaxe du constructeur :

```
FPDF(string orientation,string unite,mixed format)
```

Le paramètre *orientation* permet de choisir si le document est présent en mode portrait ('P' pour *Portrait*) ou en mode paysage (« L » pour *Landscape*). Par défaut, le document est mode portrait.

Le paramètre *unite* permet de choisir le système de mesure qui sera utilisé par la suite dans le document. On peut choisir un des systèmes de mesure suivant :

- 'pt' pour le point (1/72 de pouce soit 0,35 millimètre)
- 'mm' pour le millimètre (système par défaut)
- « cm » pour le centimètre

- « in » pour le pouce (2,54 centimètres)

Le paramètre *format* permet de choisir la taille de la feuille sur laquelle on va pouvoir écrire. Les formats suivants sont disponibles : « A3 », « A4 », « A5 », « Letter », « Legal » ou un tableau de deux cases comportant la largeur et la hauteur exprimés dans le système de mesure choisi.

1.2.2 La méthode Write

Avec la méthode *Write*, nous allons écrire une ligne de texte dans notre futur document PDF. Voici sa syntaxe :

```
Write(float h,string texte, mixed lien)
```

Le paramètre *h* indique la hauteur du texte à écrire.

Le paramètre *texte* contient le texte (!) à écrire.

Le paramètre *lien* permet d'indiquer une URL pour que le texte affiché soit un lien.

1.2.3 La génération

Maintenant que nous venons de construire notre document PDF, il faut savoir comment le récupérer. Pour cela on utilise la méthode *Output* dont voici la syntaxe :

```
string Output(string nom,string destination)
```

Le paramètre *nom* permet de donner un nom au document PDF. Par défaut, le document porte le nom de *doc.pdf*… mais je suis sûr que vous trouverez un nom plus original.

Le paramètre *destination* est très intéressant car il permet de gérer l'endroit où le document PDF est généré. Il existe quatre possibilités :

- 'I' qui permet d'envoyer le document PDF directement au navigateur pour qu'il l'affiche. Dans ce cas, il faut que le navigateur dispose du plug-in nécessaire à l'affichage des documents PDF.
- 'D' qui permet d'envoyer le document PDF au navigateur en forçant le téléchargement.
- 'F' qui sauvegarde le document PDF directement sur le serveur.
- 'S' qui renvoie directement le document sous la forme d'une chaîne.

1.2.4 Un exemple simple

Nous allons maintenant créer une page blanche au format A5 avec le texte « Bienvenue sur sN War » pointant vers le site www.snwar.com. Ce document PDF sera envoyé au navigateur en forçant le téléchargement.

Voici le code PHP qui génère ce document :

```
$doc=new FPDF('P','mm','A5');
$doc->AddPage();
$doc->SetFont('Arial','',14);
$doc->Write(10,'Bienvenue sur sN War','http://www.snwar.com');
$doc->Output('snwar.pdf','D');
```

1.3 La mise en page

Il est possible de modifier les marges du document PDF. Plusieurs fonctions sont disponibles.

La méthode SetMargins permet de définir les marges de gauche, de droite et du haut du document. Son utilisation est très simple.

```
SetMargins(float gauche,float haut,float droite)
```

Chacune de ces 3 marges peut également être définie individuellement avec une des méthodes suivantes.

```
SetTopMargin(float marge)
SetLeftMargin(float marge)
SetRightMargin(float marge)
```

La marge basse d'un document est configurée à partir de la méthode suivante :

```
SetAutoPageBreak(boolean auto,float marge)
```

La marge basse d'un document permet d'éviter d'écrire trop bas dans le document mais elle permet également de détecter le moment où il faut changer de page.

En mettant le booléen *auto* à *true*, dès que le bas de page est atteint, une nouvelle page est créée et l'écriture continue sur cette dernière.

Pour passer à une nouvelle page dans le code, il faut utiliser la méthode *AddPage*.

```
AddPage(string orientation)
```

Le paramètre *orientation* peut prendre les valeurs 'P' (*Portrait*) ou 'L' (*Landscape*).

Pour obtenir le numéro de la page en cours de création, il faut utiliser la méthode suivante :

```
int PageNo()
```

1.4 Les informations relatives au document

Un document PDF contient des données qui permettent d'obtenir certaines informations sur l'auteur du document, le site qui a créé le document …

Pour définir l'auteur du document PDF, il faut utiliser la méthode suivante :

```
SetAuthor(string auteur)
```

Le « créateur du document » est, en général, le nom du site qui génère le document PDF. Pour le définir il faut utiliser la méthode suivante :

```
SetCreator(string createur)
```

Un document PDF peut également se voir attribuer des mots-clés permettant ainsi sont indexation. Pour définir des mots-clés, il faut utiliser la méthode :

```
SetKeywords(string mots-cles)
```

Les mots-clés doivent être séparés par des espaces.

Il est également possible de définir le sujet du document avec la méthode suivante :

```
SetSubject(string sujet)
```

Un titre peut aussi être donné au document avec la méthode :

```
SetTitle(string titre)
```

1.5 La construction d'un document

Pour construire le document PDF, il n'existe pas que la méthode *Write*. Heureusement d'autres méthodes peuvent être utilisées…

1.5.1 Les états

Le moteur FPDF peut être placé dans un état particulier pour de nombreux paramètres. C'est ainsi que l'on peut choisir la police, la taille des caractères, la couleur du texte …

Lorsque le moteur FPDF est placé dans un état pour un paramètre, il le conserve jusqu'à ce que l'état soit à nouveau modifié.

Pour choisir la police de caractère à utiliser dans le document, il faut utiliser la méthode suivante :

```
SetFont(string famille,string style,float taille)
```

Le paramètre *famille* permet de choisir la famille de la police. Voici les familles disponibles :

- Courier (police à chasse fixe)
- Helvetica ou Arial (sans serif)
- Times (avec serif)
- Symbol (symboles)
- ZapfDingbats (symboles)

Le paramètre *style* permet de mettre en gras (B), de souligner (U) et de mettre en italique (I). Ces trois choix peuvent être cumulés.

Le paramètre *taille* permet de préciser la taille du texte.

La méthode *SetFontSize* permet de modifier uniquement la taille du texte.

```
SetFontSize(float taille)
```

Pour définir la couleur du texte, il faut utiliser la méthode suivante :

```
SetTextColor(int r,int v,int b)
```

Chacune des trois composantes RVB doit avoir une valeur entre 0 et 255.

Les méthodes *SetDrawColor* et *SetFillColor* permettent également de gérer la couleur des traits et du remplissage.

1.5.2 Le positionnement

Pour réaliser un document complexe, il faut souvent se positionner et écrire sur des endroits bien précis du document PDF.

Pour écrire du texte à une position précise, il faut utiliser la méthode suivante :

```
Text(float x,float y,string texte)
```

Les paramètres *x* et *y* précisent la position de début du texte dans le document.

Le paramètre *texte* est une chaîne de caractère à afficher.

Certaines fonctions écrivent toujours à la position courante sur le document, il faut donc pouvoir modifier la position courante afin de conduire la construction du document.

La méthode *SetXY* permet de faire ce changement de position.

```
SetXY(float x,float y)
```

Les paramètres *x* et *y* précisent la nouvelle position courante dans le document.

De la même manière, il existe les méthodes *SetX* et *SetY* qui permettent de modifier seulement une des deux coordonnées.

```
SetX(float x)
SetY(float y)
```

Bien sûr, on peut également, obtenir les coordonnées de la position courante à l'aide des deux méthodes suivantes :

```
float GetX()
float GetY()
```

1.5.3 Les dessins

Afin de réaliser des présentations plus efficaces, il est possible de réaliser des « dessins ». On dispose donc de méthodes pour tracer des traits et des rectangles.

La méthode *Line* permet dessiner une ligne entre deux points d'un document.

```
Line(float x1,float y1,float x2,float y2)
```

Les paramètres *x1*, *y1*, *x2* et *y2* indiquent les coordonnées des deux points permettant de tracer la droite.

La taille du trait ainsi que sa couleurs sont fixés par les méthodes *SetLineWidth* et *SetDrawColor*.

```
Rect(float x,float y,float w,float h,string style)
```

Les paramètres *x* et *y* indiquent la position du coin supérieur gauche du rectangle.

Les paramètres *w* et *h* indiquent la largeur et la hauteur du rectangle.

Le paramètre *style* permet de préciser la manière dont le rectangle va être construit :

- D : cette lettre permet d'indiquer qu'il faut dessiner le contour du rectangle
- F : cette lettre permet d'indiquer qu'il faut faire le remplissage du rectangle

Il est également possible de tracer le contour et de réaliser le remplissage en même temps.

1.5.4 Les images

Pour produire un document plus agréable, il est souvent nécessaire de placer des images. La bibliothèque FPDF permet de le faire.

Pour cela, il faut utiliser la fonction suivante :

```
Image(string fichier,float x,float y,float w,float h,string type,mixed lien)
```

Le paramètre *fichier* permet d'indiquer le fichier image qui doit être intégré au document. Il doit s'agir d'un fichier présent sur le serveur Web.

Les paramètres *x* et *y* indiquent la position du coin supérieur gauche de l'image dans le document.

Les paramètres *w* et *h* indiquent la largeur et la hauteur de l'image dans le document. Il est donc possible de déformer l'image. Si l'un de ces deux paramètres n'est pas renseigné, FPDF s'arrange pour le calculer en supposant qu'il faut conserver le rapport hauteur/largeur.

Le paramètre *type* permet d'indiquer le type de fichier dont il s'agit. Seuls les paramètres JPEG et PNG sont acceptés. Si ce paramètre n'est pas précisé, FPDF tente de deviner le format du fichier à l'aide de son extension.

Le paramètre *lien* permet de créer une image qui pointe vers une URL.

1.5.5 Les tableaux

Pour réaliser des tableaux ou des structures complexes, on dispose des méthodes *Cell* et *MultiCell*.

Voici la syntaxe de la méthode *Cell* :

```
Cell(float w,float h,string t,mixed bord,int ln,string align,int fill,mixed lien)
```

Les paramètres *w* et *h* permettent de préciser la largeur et la hauteur de la cellule.

Le paramètre *t* est la chaîne de caractères à afficher.

Le paramètre *bord* permet de gérer la bordure de la cellule. Plusieurs choix sont possibles. On peut tout simplement choisir d'avoir un bord (1) ou non (0) mais on peut également choisir d'activer seulement certains bords. Pour cela, on dispose des lettres L (gauche), T (haut), R (droite) et B (bas). Ces dernières peuvent être utilisées simultanément.

Le paramètre *ln* permet d'indiquer où se retrouve la position courante après avoir réalisé la cellule. Voici les trois valeurs possibles :

- 0 : à droite
- 1 : au début de la ligne suivante
- 2 : en dessous

Le paramètre *align* permet de spécifier l'alignement du texte dans la cellule. Voici les valeurs possibles :

- L : alignement à gauche
- C : centrage
- R : alignement à droite

Le paramètre *fill* permet d'indiquer si le fond de la cellule doit être coloré (1) ou transparent (0).

Le paramètre *lien* permet d'indiquer une URL pour que le texte de la cellule soit un lien.

Dans le même genre, on retrouve la méthode *MultiCell* qui permet de réaliser une cellule avec du texte sur plusieurs lignes. La syntaxe est proche de la méthode *Cell* :

```
MultiCell(float w,float h,string txt,mixed border,string align,int fill)
```

1.6 Dérivation de la classe FPDF

Pour avoir encore une meilleure mise en page, il est possible de dériver la classe FPDF c'est-à-dire de créer une nouvelle classe qui hérite de la classe FPDF.

Cela permet de surcharger des méthodes particulières pour leur donner une utilité particulière.
La méthode *Header* permet ainsi de définir un en-tête pour le document. Par défaut, cette méthode est vide.

Voici un exemple de nouvelle classe dérivée de FPDF.

```
class PDF extends FPDF
{
        function Header()
        {
        $this->SetFont('Arial','B',15);
        $this->Cell(80);
        $this->Cell(30,10,'Titre',1,0,'C');
        $this->Ln(20);
        }
}
```

De la même manière, on dispose de la méthode *Footer* qui peut également être redéfinie pour créer un pied de page.

1.7 Plus loin

Sur le site www.fpdf.org de nombreux scripts sont disponibles pour repousser les limites de la bibliothèque FPDF…

On trouve des scripts permettant de :

- Réaliser des codes-barres
- D'afficher du texte de manière circulaire
- D'ajouter des images avec gestion de la transparence
- De créer des cellules avec un contenu ajusté
- …

2. La bibliothèque GD

La bibliothèque GD permet de générer et de manipuler des images à partir d'un script PHP. Plusieurs utilisations peuvent donc être mises à profit. La bibliothèque GD est généralement utilisée pour générer une image dynamique qui est renvoyée par le script PHP au lieu d'une page HTML ou manipuler une ou plusieurs images avant de les enregistrer sur le serveur Web. Ces deux utilisations sont intéressantes.

2.1 L'Installation de GD

La bibliothèque GD est disponible de base dans les dernières distributions de PHP. Il faut néanmoins réaliser quelques opérations pour qu'elle puisse être utilisée.

Dans un premier temps, il faut éditer le fichier *php.ini* qui est présent dans le répertoire de Windows. Il faut retirer le point-virgule de la ligne suivante :

```
extension=php_gd2.dll
```

Par la suite, il faut s'assurer que le fichier *php_gd2.dll* est bien présent dans le répertoire de PHP. Ce fichier est disponible sous Plei@d dans l'archive *php_gd2.zip*.

Voilà, la bibliothèque GD est opérationnelle. Toutes ses fonctions peuvent donc être utilisées.

2.2 La gestion de l'image

2.2.1 Créer une image

Lorsqu'on veut créer une image, il y a plusieurs possibilités. Il est possible de créer une image à palette ou une image en couleurs vraies. Cela dépend de l'utilisation qui sera faite de l'image et du format final. Pour produire une image au format BMP, il faut tabler sur une image à palette alors que pour une image au format JPEG, il faut tabler sur une image en couleurs vraies.

Pour créer une image à palette, il faut utiliser la fonction suivante :

```
ressource imagecreate(int x_size,int y_size)
```

Les paramètres *x* et *y* correspondent à la largeur et à la hauteur de l'image en pixels.

Cette fonction retourne une ressource qui permet par la suite de manipuler l'image.

La fonction permettant de créer une image en couleurs vraies est similaire :

```
ressource imagecreatetruecolor(int x_size,int y_size)
```

2.3 Ouvrir une image

Pour ouvrir une image, on utilise également une fonction de « création ». Pourquoi cela ? En fait, c'est très simple… Lorsqu'on ouvre une image, la bibliothèque GD va créer, en mémoire, une ressource qui contient toutes les données de l'image.

La fonction suivante permet de faire une ouverture de fichier avec détection automatique du format de l'image.

```
ressource imagecreatefromstring(string fichier)
```

Le paramètre *fichier* permet d'indique le fichier à ouvrir.

Pour ouvrir une image d'un type bien déterminé, plusieurs autres fonctions peuvent être utilisées.

```
ressource imagecreatefromjpeg(string fichier)
ressource imagecreatefromgif(string fichier)
ressource imagecreatefrompng(string fichier)
```

Par la suite, une image, qu'elle soit créée ou ouverte, va être manipulée de la même manière.

2.3.1 Renvoyer ou sauvegarder une image

Pour renvoyer une image à la place d'une page HTML, il faut, dans un premier temps, modifier l'en-tête qui va être envoyé au navigateur afin que celle-ci corresponde bien avec l'en-tête d'une image.

Pour cela, il faut utiliser la fonction *header* en précisant le type d'en-tête…

```
header("Content-type: image/png");
header("Content-type: image/jpeg");
header("Content-type: image/gif");
...
```

Ensuite, pour envoyer les données au navigateur, il faut appeler une des fonctions suivantes :

```
bool imagejpeg(ressource image,string fichier,int qualité)
bool imagegif(ressource image,string fichier)
bool imagepng(ressource image,string fichier)
```

Le paramètre *image* est une ressource de type image obtenue avec une fonction de « création ». Si le paramètre *image* est le seul indiqué alors le script PHP va renvoyer l'image vers le navigateur. Il faut donc utiliser le bon en-tête avec la bonne fonction.

Le paramètre *fichier* permet, plutôt que de renvoyer l'image vers le navigateur, de la stocker dans un fichier.

Le paramètre *qualité* permet de préciser une qualité pour l'image JPEG de 0 (basse) à 100 (haute).

2.3.2 Détruire une image

Il ne faut pas oublier qu'une ressource image prend de la place en mémoire sur le serveur Web. Il faut donc dès que possible rendre disponible cette mémoire sans attendre la fin du script PHP.

Pour détruire une ressource image, il faut utiliser la fonction suivante :

```
bool imagedestroy(ressource image)
```

2.4 La Synthèse d'images

Pour produire une image à partir d'une ressource image vide, de nombreuses fonctions sont disponibles.

On se retrouve avec une gestion des états pour chaque image. Ainsi, il est possible d'activer ou de désactiver certains paramètres.

Par exemple, pour activer l'anti-aliasing lors de la création d'une image, il faut utiliser la fonction suivante :

```
bool imageantialias(ressource image,bool actif)
```

Le paramètre *actif* doit être mis à *true* pour que l'anti-aliasing soit actif et à *false* dans le cas contraire. Par la suite, toutes les fonctions qui permettent de dessiner dans l'image utiliseront cet état.

Pour chaque couleur qui sera utilisée dans une image, il faut appeler la fonction suivante :

```
int imagecolorallocate(ressource image,int r,int v,int b)
```

Le paramètre *image* identifie l'image dans laquelle la couleur sera utilisée.

Les paramètres *r*, *v* et *b* sont les trois composantes RVB qui permettent d'aboutir à une couleur finale.

On peut utiliser la fonction *imagefill* pour effectuer un remplissage avec une couleur.

```
bool imagefill(ressource image,int x,int y,int couleur)
```

La fonction *imageline* permet de tracer une droite.

```
bool imageline(ressource image,int x1,int y1,int x2,int y2,int couleur)
```

Il est possible de dessiner diverses formes géométriques dans une image comme des rectangles, des cercles … On trouve une version avec ou sans remplissage de la forme…

```
bool imagerectangle(ressource image,int x1,int y1,int x2,int y2,int couleur)
bool imageellipse(ressource image,int cx,int cy,int w,int h,int couleur)
...
bool imagefilledrectangle(ressource image,int x1,int y1,int x2,int y2,int couleur)
...
```

Il est aussi possible de placer du texte dans une image avec la fonction suivante :

```
bool imagestring(ressource image,int police,int x,int y,string s,int couleur)
```

Pour découvrir les nombreuses fonctions disponibles, consultez le site www.php.net.

2.5 Le traitement d'images

Il est aussi intéressant de réaliser des traitements sur une image qui vient d'être envoyée au serveur Web.

Une fonction très intéressante permet de redimensionner une image, la voici…

```
bool imagecopyresampled(ressource dst_image,ressource src_image,int dst_x,int dst_y,int
src_x,int src_y,int dst_w,int dst_h,int src_w,int src_h)
```

Les paramètres permettent de préciser la dimension d'origine et la dimension d'arrivée ;

Ce genre de traitement est très intéressant lorsqu'une image est envoyée sur un serveur Web par un navigateur car cette dernière peut alors être convertie en utilisant une taille optimale pour ce à quoi elle va servir. Cela permet d'éviter les images ayant un taille d'origine de 1024x768 pixels qui se retrouvent affichées à l'écran au format 400x300 avec un redimensionnement basic réalisé par le navigateur et un fichier qui pèse bien plus lourd sur le serveur et au moment de l'envoi vers un navigateur.

Une correction gamma peut être appliquée à une image avec la fonction suivante :

```
bool imagegammacorrect(ressource image,float inputgamma,float outputgamma)
```

Il est aussi possible d'appliquer une rotation à une image…

```
resource imagerotate(ressource src_im,float angle,int bgd_color,int option)
```

Pour découvrir les nombreuses fonctions disponibles, consultez le site www.php.net

3. Le JSON

Pour terminer ce cours, nous allons voir comment créer et manipuler un fichier de données JSON avec un script PHP.

3.1 Introduction au format JSON

JSON (JavaScript Object Notation) est un format qui permet de représenter de l'information structurée.

Ce format est de plus en plus utilisé et vient remplacer progressivement le format XML.

Les avantages sont :
- Génération et analyse facile pour un ordinateur
- Syntaxe simple
- Structure en arborescence
- Beaucoup moins « verbeux » que le XML
- Format ouvert
- Permet de manipuler plusieurs types de données : chaînes de caractères, nombres, tableaux, objets, booléens.

Quelques inconvénients :
- La structure doit être connue avant l'utilisation
- Le fichier doit être encodé en utf-8

JSON se base sur deux structures :
- Une collection de couples nom/valeur
- Un tableau contenant des valeurs

Un objet est un ensemble de couples nom/valeur, il commence par une accolade gauche et termine par une accolade droite.

Voici un premier exemple de fichier au format JSON :

```
{
    "nom":"Forest Gump",
    "realisateur":"Robert Zemeckis",
    "annee":1994,
    "genre":"Comédie dramatique"
}
```

Ce fichier contient un objet contenant des informations suivantes :
-un nom
- un réalisateur
- une année
- un genre

On peut ainsi deviner que cet objet représente un film, mais rien ne peut l'indiquer au niveau du format.

3.1.1 La syntaxe

La syntaxe est assez similaire au langage Javascript, elle est basée sur l'utilisation de signes de ponctuation. Le tableau ci-dessous liste les signes les plus importants.

Signe	Explications
{…}	Les accolades permettent de définir un objet
"nom" :"valeur"	Un couple nom/valeur utilise les guillemets et les double-points
[…]	Les crochets permettent de définir un tableau
,	La virgule permet de séparer les couples nom/valeur d'un tableau ou d'un objet.

3.1.2 Exemple commenté

Maintenant que nous avons vu la syntaxe, voici un fichier au format JSON un peu plus complet :

```
{
    "films" : [
        {
            "nom":"Forest Gump",
            "realisateur":"Robert Zemeckis",
            "annee":1994,
            "genres": ["Comédie", "Drame"]
        },
        {
            "nom":"Free Guy",
            "realisateur":"Shaw Levy",
            "annee":2021,
            "genres": ["Comédie", "Action", "Aventure"]
        }
    ]
}
```

Dans cet exemple, nous avons notre objet principal définit avec les accolades (obligatoire), qui contient un couple nom/valeur (qui se nomme aussi « membre ») dont le nom est « films ».

La valeur de ce membre est un tableau d'objet (présence de crochets []).

Chaque objet contient un nom, un réalisateur, une année et un tableau de genres (tableau de chaine texte).

3.2 Lire et afficher le contenu d'un fichier JSON en PHP

Les fonctions utilisées pour la création ou la lecture d'un fichier au format JSON en PHP sont :

- *json_encode()* : retourne la représentation JSON d'une valeur. Cette fonction va encoder la valeur passée en paramètre en format JSON.
- *json_decode()* : décode une chaine JSON. Cette fonction récupère une chaine encodée au format JSON et la convertit en une variable PHP.
- *file_get_contents()* : cette fonction permet de lire un fichier et de mettre le contenu dans une variable.

3.2.1 Lecture

Nous allons écrire un script qui permet de lire un fichier au format JSON puis de récupérer le contenu afin de le convertir en une variable PHP.

```php
<?php
/* Récupération du contenu du fichier .json */
$contenu_fichier_json = file_get_contents('3.1.2.json');

// Vérification
var_dump($contenu_fichier_json);

// Décodage du json
$objet = json_decode($contenu_fichier_json);

// Vérification
var_dump($objet);
?>
```

Le résultat de ce script affiche :

```
D:\Windows\Download\UwAmp\www\3.2 Lecture fichier json.php:6:string '{
    "films" : [
        {
            "nom":"Forest Gump",
            "realisateur":"Robert Zemeckis",
            "annee":1994,
            "genres": ["Comédie", "Drame"]
        },
        {
            "nom":"Free Guy",
            "realisateur":"Shaw Levy",
            "annee":2021,
            "genres": ["Comédie", "Action", "Aventure"]
        }
    ]
}
' (length=383)

D:\Windows\Download\UwAmp\www\3.2 Lecture fichier json.php:12:
object(stdClass)[3]
  public 'films' =>
    array (size=2)
      0 =>
        object(stdClass)[1]
          public 'nom' => string 'Forest Gump' (length=11)
          public 'realisateur' => string 'Robert Zemeckis' (length=15)
          public 'annee' => int 1994
          public 'genres' =>
            array (size=2)
              ...
      1 =>
        object(stdClass)[2]
          public 'nom' => string 'Free Guy' (length=8)
          public 'realisateur' => string 'Shaw Levy' (length=9)
          public 'annee' => int 2021
          public 'genres' =>
            array (size=3)
              ...
```

La fonction *file_get_contents*() charge le contenu du fichier JSON dans une chaine de caractères nommée "$contenu_fichier_json".

La fonction *var_dump()* nous permet de vérifier le contenu de la variable : c'est bien une chaine de caractères (string).

Nous appelons ensuite la fonction *json_decode()* qui va convertir le format JSON en une variable PHP. Nous pouvons vérifier le contenu ensuite avec la fonction *var_dump()*.

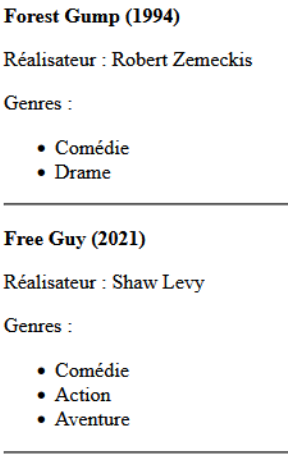
3.2.2 Affichage

Nous avons donc maintenant un objet PHP (variable $objet) qui contient les données du fichier JSON.

Nous pouvons maintenant afficher les informations dans une page web.

```
<?php
/* Récupération du contenu du fichier .json */
$contenu_fichier_json = file_get_contents('3.1.2.json');
// Décodage du json
$objet = json_decode($contenu_fichier_json);
?>
<!DOCTYPE html>
<html lang="fr">
    <head>
        <meta charset="utf-8">
        <title>Affichage du contenu du fichier JSON</title>
    </head>
    <body>
       <?php

               foreach($objet->films as $film)
               {
                       echo "<p><strong>".$film->nom." (".$film->annee.")</strong></p>";
                       echo "<p>Réalisateur : ".$film->realisateur."</p>";
                       echo "<p>Genres : </p>";
                       echo "<ul>";
                       foreach($film->genres as $genre)
                              echo "<li>".$genre."</li>";
                       echo "</ul>";
                       echo "<hr/>";
               }
       ?>
    </body>
</html>
```

L'exécution du script affiche :



3.3 Créer un fichier JSON

Dans cette dernière partie, nous allons créer un fichier JSON à partir d'une structure de données PHP.

Nous allons créer le fichier JSON vu précédemment contenant une liste de films.

Il est assez logique d'utiliser un objet Film afin de définir les informations représentant un film :

- Un nom
- Un réalisateur
- Une année
- Une liste de genres

```
class Film
{
        public $nom;
        public $realisateur;
        public $annee;
        public $genres;

        function __construct($nom, $realisateur, $annee, $genres)
        {
                $this->nom = $nom;
                $this->realisateur = $realisateur;
                $this->annee = $annee;
                $this->genres = $genres;
        }
}
```

Nous allons également créer une classe Fichier, qui va contenir notre tableau d'objet Film.

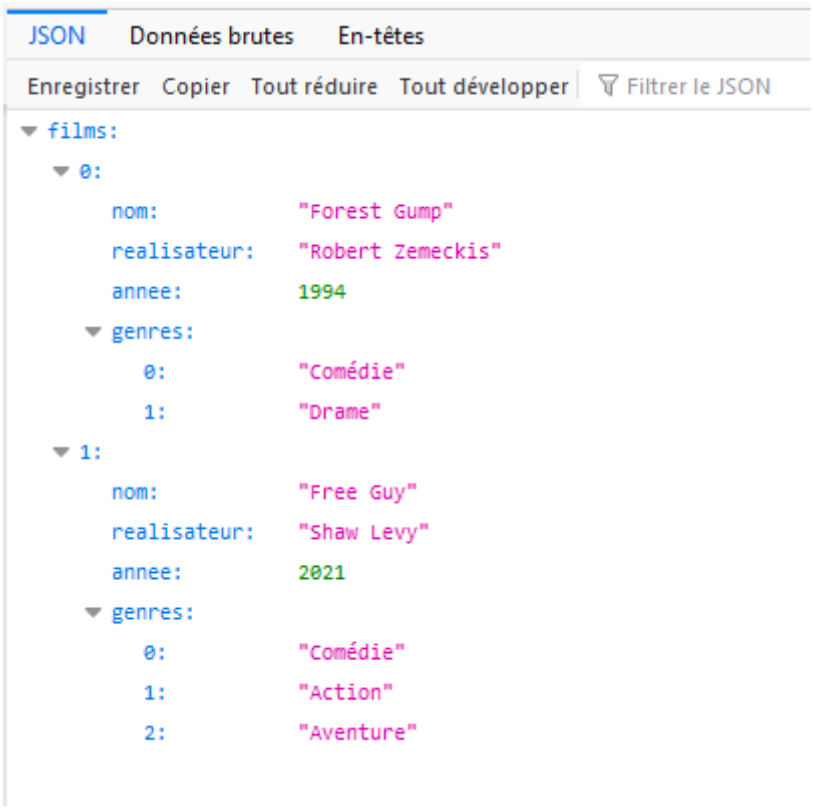
```
class Fichier
{
        public $films;
}
```

Notre script va afficher le fichier directement dans le navigateur, nous ajoutons donc en début du script un en-tête http définissant le contenu (« application/json »).

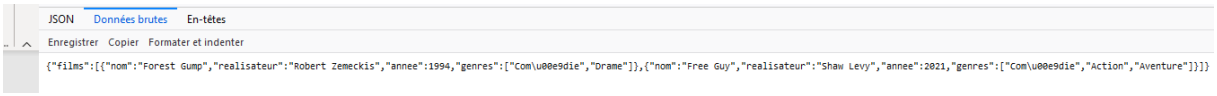
Nous utilisons la fonction *json_encode()* afin d'encoder notre structure de données au format JSON.

```
<?php


// Content type
header('Content-type: application/json');


// Inclure les classes ici


$fichier = new Fichier();


// Création du tableau de films
$mesFilms = array();
// Ajout du premier film
array_push($mesFilms, new Film("Forest Gump", "Robert Zemeckis", 1994, array("Comédie",
"Drame")));
// Ajout du deuxième film
array_push($mesFilms, new Film("Free Guy", "Shaw Levy", 2021, array("Comédie", "Action",
"Aventure")));


$fichier->films = $mesFilms;


// Affichage dans le navigateur
echo json_encode($fichier);


?>
```

Le navigateur interprète le format sans problème :



Les données brutes correspondantes sont :



Nous reverrons le format JSON lors de notre webconférence portant sur la production de documents avec PHP.