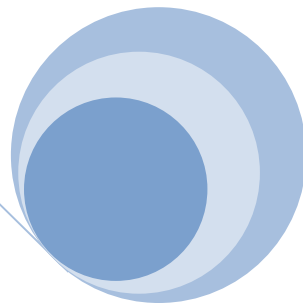
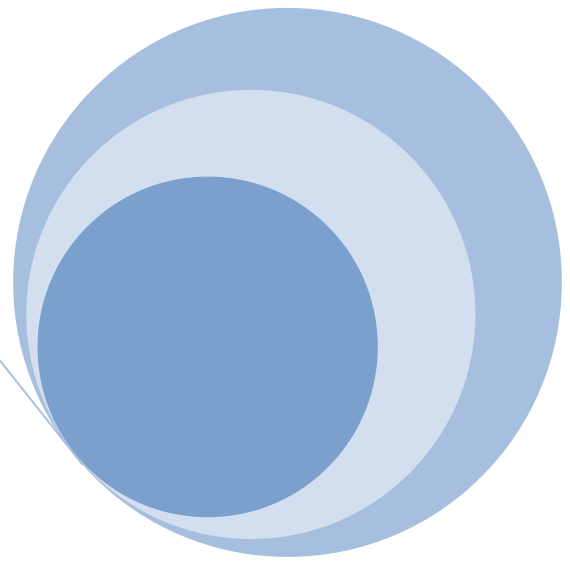


Geneager



Documentation

Version 1 du CMS

Gestionnaire de contenu gratuit pour la généalogie.

CYRIL HAVRET

21 December 2021

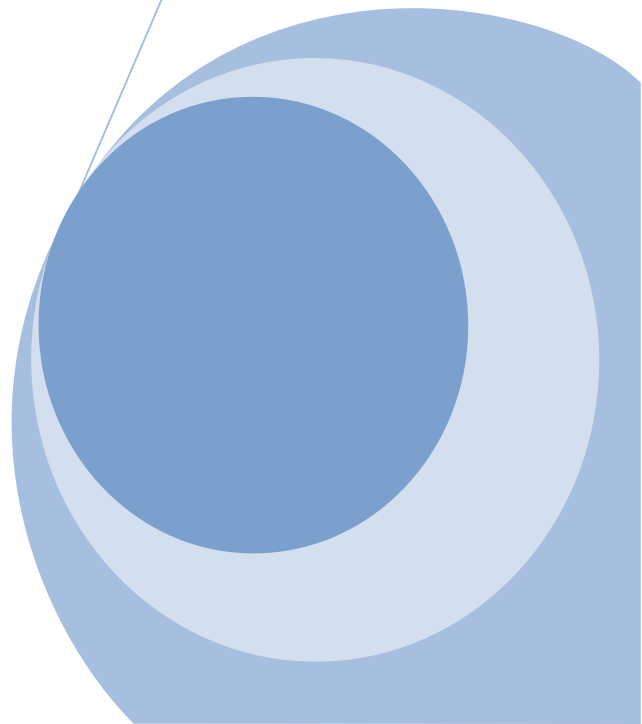


Table des matières

1 – GÉNÉRALITÉS.....	3
1.1 - INTRODUCTION.....	3
1.2 – LICENCE ET DROITS.....	3
2 – ARCHITECTURE.....	3
2.1 – ARCHITECTURE DES DOSSIERS.....	3
2.2 – AUTOLOADER.....	4
2.3 – CHARGEMENT INTELLIGENT DU MVC.....	4
3 – CONVENTION DE NOMMAGE.....	5
3.1 - GÉNÉRALITÉS.....	5
3.2 – MODELS, VIEWS ET CONTROLLERS.....	6
3.3 – NOMS RÉSERVÉS.....	6
3.3.1 – Les fichiers.....	6
3.3.2 – Les variables et constantes PHP.....	6
3.2.3 – Les « namespaces ».....	7
3.4 – MOTS CLÉS APPRÉCIÉS.....	7
4 – LA BASE DE DONNÉES.....	7
4.1 – GÉNÉRALITÉS.....	7
5 – PHP.....	8
5.1 – CRÉER UNE PAGE.....	8
5.1.1 - Généralités.....	8
5.1.2 – Ajouter d’une vue + titre + meta dans le controller.....	8
5.1.3 – Ajouter des ressources CSS et JS.....	9
5.1.4 – Changer le « header », la « navbar » et le « footer ».....	9
5.1.5 – Classes POST et GET.....	10
5.1.6 – Créer un formulaire.....	10
6 – MODS ET DÉPENDANCES.....	11
6.1 - APACHE2.....	11
6.2 - PHP MODS.....	11
7 – LIENS.....	11

1 – Généralités

1.1 - Introduction

Gestionnaire de contenu GENEAGER est destiné pour la généalogie et ayant pour but de démocratiser d'avantage l'indépendance numérique dans le monde de la généalogie: offrant ainsi donc entre-autres un meilleur contrôle des données saisies (la révocation par exemple).

A terme un installateur sera inclus pour démocratiser d'avantage « en ciblant » les utilisateurs plus novices.

1.2 – Licence et droits

L'ensemble des codes du projet sont sous licence Creative Commons 4.0 (CC BY-NC-SA 4.0), cela signifie que :

- **Vous pouvez partager** (copier et distribuer et communiquer le matériel par tous moyens et sous tous formats)
- **vous pouvez adapter:** vous devez distribuer vos contributions sous la même licence que l'original.
- **Vous ne pouvez pas** utiliser les réalisations à des fins commerciales

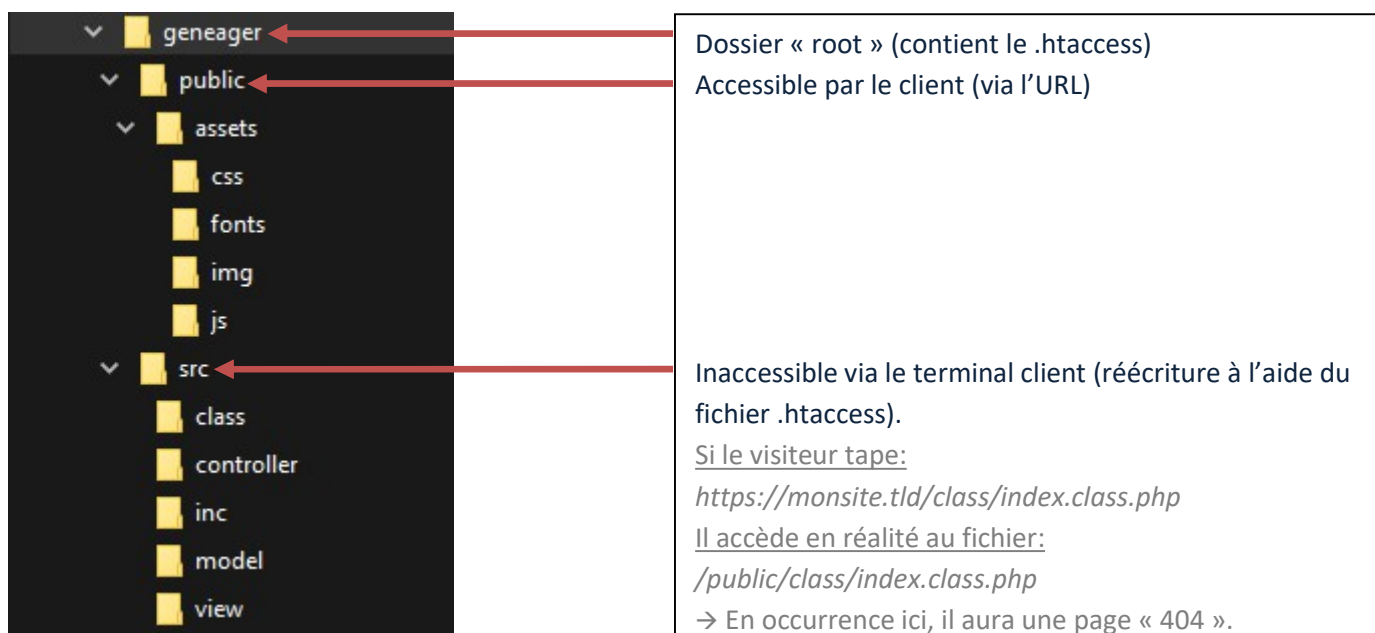
Plus d'informations sur la licence sur la page suivante: <https://creativecommons.org/licenses/by-nc-sa/4.0/legalcode>

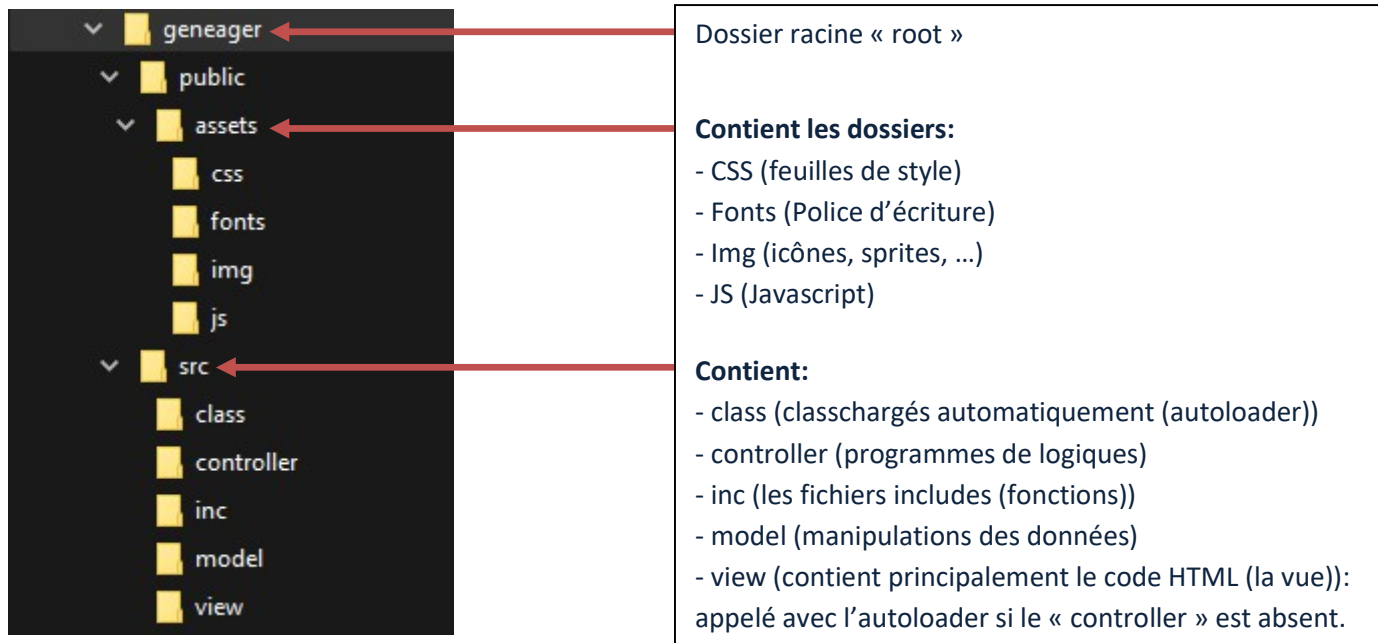
2 – Architecture

L'architecture du CMS est de type MCV et utilise un système de «routeur PHP ».

La prise en charge des règles (.htaccess) est requise pour le bon fonctionnement du système.

2.1 – Architecture des dossiers





2.2 – Autoloader

Le système d'autoloader permet de charger automatiquement les fichiers de classes.

Vous devez créer un sous dossier portant le nom du « namespace » dans le dossier « src/class » si vous vous utilisez un « namespace »

2.3 – Chargement intelligent du MVC

Le système de chargement intelligent est une fonctionnalité qui permet de charger les bons fichiers en fonction de l'URL.

Par exemple « <https://monsite.tld/home> » appellera les fichiers suivants :

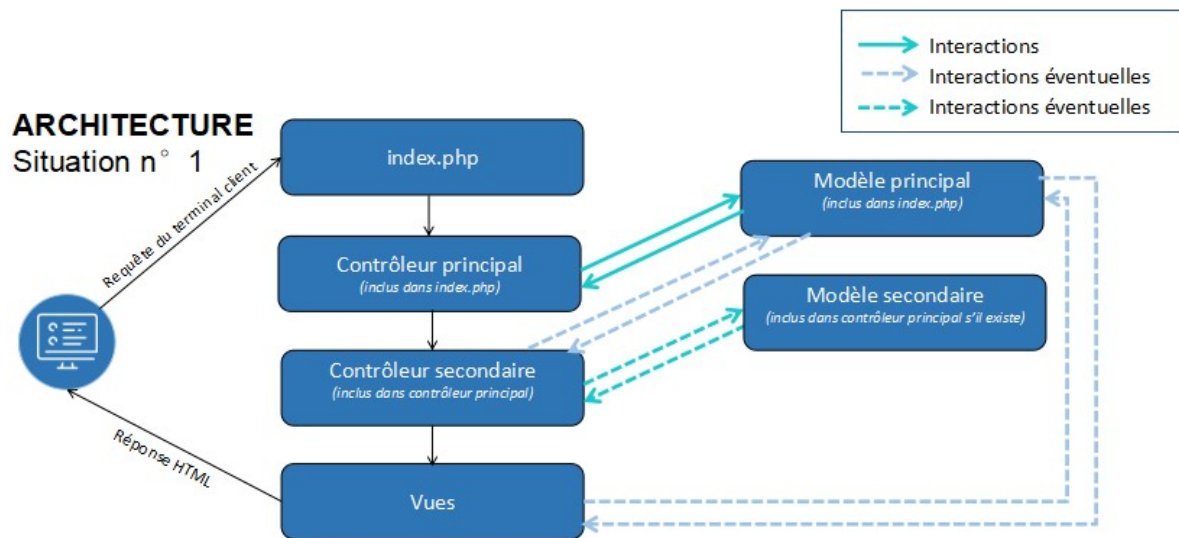
- /src/model/home.php

- /src/view/home.php (si le « controller » est absent)

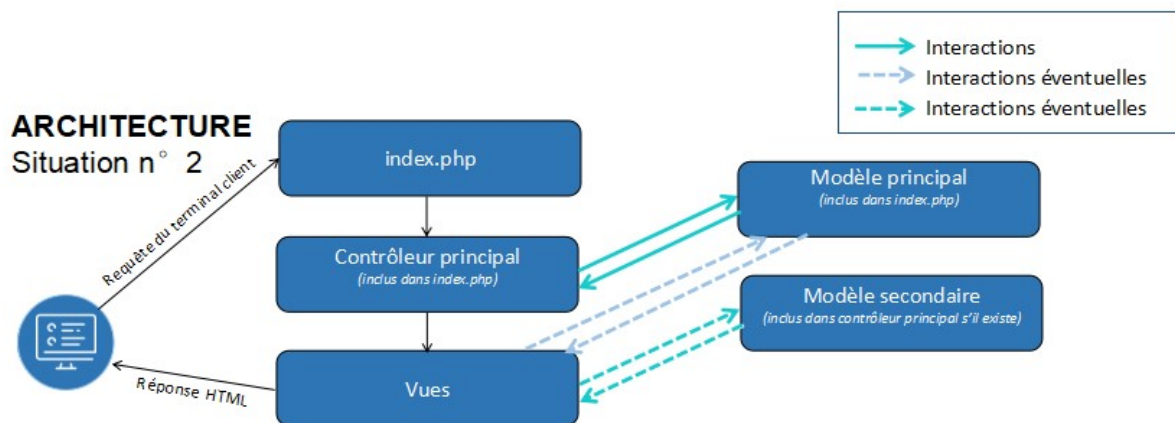
- /src/controller/home.php (programme logique devant appeler la « view »)

Si la vue n'existe pas alors la vue « 404 » est appelée.

Toutes les classes générales (utilisées dans plusieurs « controllers ») devront être rangées dans le dossier dédié nommé « class » (situé dans le dossier « src »).



Situation 1: le contrôleur secondaire existe



Situation 2: le contrôleur secondaire n'existe pas

3 – Convention de nommage

3.1 - Généralités

Nous utiliserons l'**anglais** et **uniquement des caractères alphanumériques** (chiffre et lettres **non-accentués**) pour l'ensemble des nommages.

Nous commencerons chaque nommage tel qu'il soit commence par une minuscule. Le 2^{ème} mot (particule) commencera avec une majuscule et sans espace ni « underscore » (tiret bas).

Exemples:

- `maClassIdentity`
- `exampleFile.php`



Exception

Cependant nous utiliserons tout de même des noms de variables composés séparés de tirés bas (« underscore ») pour les « variables globales » afin de réduire les risques de conflit (ex: \$meta_keyword).

On évitera également les déterminants du type « the ».

Exemples:

- ~~theIdentityCard~~
- ~~theExampleFile.php~~

Et privilégierons le singulier au pluriel.

Exemple: ~~identityCards.php~~ → identityCard.php

3.2 – Models, views et controllers

Chaque fichier MCV devra être rangé dans son dossier attitré avec la bonne double extension.

Exemples:

- /model/model/home.php
- /src/view/home.php
- /src/controller/home.php

3.3 – Noms réservés

3.3.1 – Les fichiers

Les noms de fichiers à 3 chiffres dans les dossiers MVC (controller, model, view) **sont réservés aux pages erreurs** (principalement 2xx, 4xx, 5xx).

Exemple: /src/view/403.php doit-être réservé pour la page d’erreur « 403 – Access Forbidden »

3.3.2 – Les variables et constantes PHP

3.2.2.1 – Constantes


Constantes	Commentaire
MVC	Contient le chemin d’accès des fichiers MVC, mais aussi des classes « génériques » et les fichiers « include »
ENCODE	Contient l’encodage de caractères (utilisé pour la déclaration dans le HTML + fonctions PHP de base).

3.2.2.2 – Les variables

Variables	Commentaire
\$db_	Les variables commençants par db_ sont réservés pour la base de données.
\$obj_	Les variables commençant par « \$obj_ » sont réservées pour les objets.
\$meta_	Réservé pour le titre de la page et balises « meta » (description, keyword,).
\$include_MVC	Réservé aux fichiers à charger (view, model, controller)
\$include_JsCss	Réservé aux fichiers additionnels (JS, CSS) à ajouter dans le « head » du squelette HTML
\$include_header	A définir dans le contrôleur secondaire pour modifier l'en-tête visuelle par défaut
\$include_menu	A définir dans le contrôleur secondaire pour modifier le menu par défaut
\$include_footer	A définir dans le contrôleur secondaire pour modifier le pied de page par défaut

3.2.3 – Les « namespaces »

Le namespace « gng » est réservés pour les classes de bases du CMS.

 Certains exemples dans la documentation contient le namespace « geneager » : il s'agit de l'ancien nom. Utilisez donc bien le namespace « gng »

Si vous développez un ou une dépendance ou un module complémentaire (bien que cette fonctionnalité n'existe pas encore) pour ce CMS évitez de l'utiliser ce namespace réservé afin d'éviter les conflits

3.4 – Mots clés appréciés

Mot-clé	Exemple	Commentaire
list	/src/class/userList.class.php	C'est plus évocateur de que « users.class.php »
view		
search		
display		

4 – La base de données

4.1 – Généralités

Pour les noms de tables nous reprendrons la même convention de nommage des fichiers (caractères alphanumériques non accentués uniquement, première lettre en minuscule, premières lettre en majuscule pour les noms composés).

Exemple:

- MysqlTable
- birthPlace

5 – PHP

5.1 – Créer une page

5.1.1 - Généralités

Pour créer une page il suffit de créer vos fichiers MCV dans leurs dossiers respectifs.

Exemple:

/src/model/croplImage.model.php → Le « model » (classes respectives à la page)

/src/view/croplImage.php → La « vue » (principalement le HTML)

/src/controller/croplImage.php → Le « controller » (programme logique)

Pour afficher la page ici nous devons appeler l'adresse « <https://monsite.tld/croplImage> ».

→ Si le controller « croplImage.php » est supprimé le « router » ouvrira le « model » (s'il existe) puis la « view ».

→ Si le controller existe il faudra appeler la vue dans le controller (en effet si le « controller » existe le « router » laissera alors le programme logique choisir).

Concernant les autres dossiers, pour rappel:

Dossier	Nommage	Commentaire
/src/inc/	*.inc.php	Fonctions ou vues commune (comme un menu par exemple commun à plusieurs pages)
/src/class/	*.class.php	Classes PHP communes à plusieurs fichiers

5.1.2 – Ajouter d'une vue + titre + meta dans le controller

Exemple simple d'ajout d'une vue (initialisée par le controller)

```

src > controller > test.controller.php > ...
1 <?php
2 $meta_title = "My title test";
3 $meta_description = "Just a description";
4 $meta_keyword = "test, test, test";
5 if(1==1){
6     \geneager\mcv::addView("test");
7 }
8

```

Titre (balise title)

Meta description

Meta mots-clés

Ici je charge ma vue (/src/view/test.php)

Namespace
geneager -> gng
classe PHP
fonction (statique)
vue à afficher

Pour modifier ou ajouter la balise « meta robots » il suffira juste d'utiliser la variable « meta_robots » et d'y assigner la valeur désirée.

Pour ajouter une vue il suffira de copier-coller la même ligne (6 de l'exemple ci-dessus) et de remplacer « test » par le nom de l'autre vue.

Exemple: `\gng\mvc::addView("test2");`

 **Les meta-tags et le titre doit-être inclus dans le « controller » pour être prises en compte.**

 **Les fichiers sont chargés dans l'ordre d'ajout !**

5.1.3 – Ajouter des ressources CSS et JS

Pour ajouter une feuille de style CSS ou JS il suffit de saisir le nom de la classe « additionalJsCss » précédé de l'espace nom « gng » puis faire appel à la fonction « set » comme dans l'exemple suivant :

```
\geneager\additionalJsCss::set("style2.css");  
\geneager\additionalJsCss::set("monJS.js");
```

 **geneager -> gng**

 **Seuls les extensions « CSS » et « JS » sont acceptés.**

 **Les fichiers doit-être appelés dans dans le « controller » pour être pris en compte.**

5.1.4 – Changer le « header », la « navbar » et le « footer »

```
$include_header = "header2";  
$include_navbar = "navbar2";  
$include_footer = "footer2";
```

Pour changer il suffit de (re)définir la bonne variable.

Pour modifier les valeurs par défaut appliquées sur tout le site il faut modifier les variables dans le fichier de config.

Pour modifier le « header », la « navbar » ou le « footer » d'une page spécifique il suffit juste de redéfinir la variable concernée dans le « controller » de la page.

Pour ne pas afficher l'en-tête, la barre de navigation ou le pied de page il suffit de mettre la valeur « none ».

 **Si la valeur est « footerAdmin » c'est alors le fichier « src/inc/footerAdmin.inc.php » qui sera appelé.**

 **Le fichier par défaut n'est pas chargé si le fichier redéfini n'existe pas.**

5.1.5 – Classes POST et GET

Depreacted

5.1.6 – Créer un formulaire

Contrairement à de nombreuses fonctions de base, je dois tout d'abord créer un nouvel objet pour pouvoir créer un formulaire !

En effet car une page peut contenir plusieurs formulaires.

Pour cela je dois créer une variable suivie d'un « new », du « namespace » et de la classe « form ».

Dans la classe « form » il est obligatoire de spécifier la méthode et la cible où les données seront envoyées.

Pour ajouter un nouvel élément à mon formulaire je dois faire appel à la fonction « setElement » de ma classe « form ».

La fonction « setElement » doit contenir obligatoirement 2 paramètres.

Le premier doit contenir le nom de balise (ex : « input », « textarea », « select », ...), le second doit contenir tous les attributs sous forme de tableau.

Exemple:

```
$formLogin = new \gng\form(array( // i declare my new object
    "method" => "post", // i give the method attr
    "action" => "?", // i give action attr
    "class"=>"login", // i give className ou className list (not required)
));


$formLogin->setElement("input", array( // here i give the type of tag
    "type" => "text", // i give the type of input
    "placeholder" => "Nom d'utilisateur", // i set a placeholder
    "name" => "username", // i give a className
    "required" => "required", // i add the attr required
    "minlength" => \gng\db::getParameter("usernameMinLength"), // i add the attr minlength
    "maxlength" => \gng\db::getParameter("usernameMaxLength") // i add the attr maxlength
));

$formLogin->setElement("input", array(
    "type" => "password",
    "placeholder" => "Mot de passe",
    "name" => "password",
    "required" => "required",
    "minlength" => \gng\db::getParameter("passwordMinLength"),
    "maxlength" => \gng\db::getParameter("passwordMaxLength")
));

$formLogin->setElement("input", array(
    "type" => "submit",
    "value" => "Connexion",
    "name" => "submit",
    "class" => "btn btn-primary" // i add a class to the element
));
```

 Les éléments apparaissent dans l'ordre dans lesquels ils sont créés.

 Pour éviter les conflits, bugs il est conseillé d'utiliser les entités html pour les guillemets (simple, doubles ou « alt gr 7 »).

 L'attribut « name » est requis pour tous les éléments si vous souhaitez utiliser la méthode « check ».

Pour un textarea

```
$myForm->setElement("textarea", array( //textarea
    "value" => "test textarea",        // je peux spécifier sa valeur
    "maxlength" => "200"                // je lui fixe un attribut. ici l'attribut "maxlength" avec une valeur à "200"
));
```

Pour ajouter des «options» à un menu select il suffit de créer un tableau nommé « option». Dans la clef («key») il est conseillé de mettre la « value » et de mettre la valeur associé en tant que texte.

```
$formLogin->setElement("select", array(
    "class" => "myClass",
    "option" => array(
        "1" => "Accueil",
        "2" => "Paramètres"
    )
));
```

Pour afficher le formulaire il suffit juste de faire appel à l'objet et la méthode display.

⚠ Les navigateurs omettent les listes à choix multiples («select multiple») si l'utilisateur sélectionne aucune valeur. Cette omission peut donc entraîner une erreur inattendue lors de l'utilisation de la méthode check. Par conséquent l'attribut « required » et un choix « nul » est imposé par la méthode « display » si l'attribut « required » n'est pas renseigné dans l'objet.

```
echo $myForm->display();
```

6 – Mods et dépendances

6.1 - Apache2

Mod	Commentaire
RewriteEngine	Est requis

6.2 - PHP mods

Mod	Commentaire
Imagick	Fortement recommandé
curl	Fortement recommandé

7 – Liens

URL	Commentaire
https://cyril.ovh	Site du fondateur du projet
https://www.apachefriends.org/	Mettez en place votre serveur maison facilement sous Windows avec XAMPP (Apache, MySQL, PHP, PhpMyAdmin)
https://visualstudio.microsoft.com/	Visual Studio: éditeur de codes complet et gratuit