Table des matières

[1. Fonctionnement de base. 2](#_Toc32840106)

[2. Créer une vue 2](#_Toc32840107)

[3. Créer une fonction 2](#_Toc32840108)

[4. Créer un objet 2](#_Toc32840109)

[5. Configuration 3](#_Toc32840110)

# Fonctionnement de base.

Toutes les URL passent par le fichier index.php.

Le fichier charge les fonctions,

* Cherche un contrôleur, et la vue associé.
* Une vue si pas de contrôleur
* Le traitement d’un formulaire si pas de vu
* Erreur 404 si rien n’est trouvé.

# Créer un contrôleur et une vue

Pour créer une vue est simple :

Dans le dossier contrôleurs, créer le contrôleur .php associé à la vue

« exemple.php »

Dans le dossier partials créer la vue avec le même nom que le contrôleur avec une extension .phtml et un \_ devant.

« \_exemple.phtml »

/!\ la case est prise en compte !

Toutes fonctions /variable créer dans le contrôleur sont accessibles dans la vue.

# Créer une fonction

Aller dans le fichier functions/function.php

Ecrire la fonction souhaitée si elle n’existe pas.

Elle sera accessible depuis n’importe ou sauf dans index.php ou elle ne peut être appelé qu’APRES l’appel du fichier functions.php

# Créer un objet

Placer les objets dans le dossier class. Pas besoin de « require » pour appeler un objet, ils sont appelés par défaut sauf dans index.php. L’appel d’une classe ne peut précéder l’appel des fonctions.

Si il y a besoin d’accéder à la BDD, la classe doit être un extend de BDD

# Configuration

Le fichier config/config.php sert à configurer la BDD, et les variables globales (constante).

Sont inclus de basse :

define("HOST", $host); retourn l’url de base (127.0.0.1 ou [www.cheezpa.com](http://www.cheezpa.com)) sans rien de plus.

define("RACINE\_SRV", $racine\_srv); retourn c://wamp64/www/nomProjet

define("FUNCTIONS\_PATH", $functions\_path); Emplacement des fonctions

define("FUNCTIONS\_URL", $functions\_URL); URL des fonctions

define("CONTROLLER\_PATH", $controleur\_path); emplacement des controleurs

define("CLASS\_URL", $class\_url); URL des class

define("CLASS\_PATH", $class\_path); Repertoire des Class

Répertoire des images en fonction de la taille désirée.

define("IMAGES\_PATH", $images\_path);

define("IMAGES\_URL", $images\_URL);

define('IMAGE\_PORTABLE\_URL', $imagesPortable\_URL);

define("IMAGE\_TABLETTE\_URL", $imagesTablette\_URL);

define("IMAGE\_FIXE\_URL", $imagesfixe\_URL);

define("PARTIAL\_PATH", $partial\_PATH);

define("FORMS\_PATH", $form\_path);

define("ADRESSEMAIL", $adresseMail); doit être renseigné sur les 1er ligne du fichier config

define("CSS\_URL", $css\_URL);

define("JS\_URL", $js\_URL);

On utilise le \_PATH quand php doit accéder à un fichier. Et \_URL quand le navigateur doit accéder à un fichier (ex : images)

# Créer des liens internes

<a href="maPage">lien</a> Pas d'extensions !

<picture>

<source srcset="<?php echo IMAGE\_TABLETTE\_URL;?>nomImage.jpg" media="(min-width : 576px)">

<source srcset="<?php echo IMAGE\_FIXE\_URL;?>nomImage.jpg" media="(min-width : 768px)">

<img src="<?php echo IMAGE\_PORTABLE\_URL;?>nomImage.jpg" alt="Blabla">

</picture>

Appeler un script JS

<script src="<?php echo JS\_URL;?>monScript.js"></script>

# Plugins Members

Dossier members à ajouter dans le dossier plugins.

Le dossier crée la variable $member. Il ne faut donc pas l'écraser dans un autre plugin ou autre !

Le formulaire de contact doit impérativement contenir le champ 'connect'. Il est inclus dans le generateur disponible via fonction echo $member->generatConnectForm().

Le formulaire doit être adapté, il ne ressort que les 2 inputs.