Présentation du mini-framework.

Tout d'abord inutile d'ajouter à la main des class, le spl_autoload le fait deja tout seul.

Les URL:

Les URL sont essentielles, c'est elles qui passent les paramètres au controller.
exemple:
monsite.com/admin/test/lihfgre/sdgfdhg

Va appeller dans le controller admin, la méthode test et lui passe en argument lihfgre.
Le $args[0] sera donc « lihfgre ». Le $args[1] sera « sdgfdhg ».
Les paramètres en POST sont accessibles comme dans un script classique.

Si aucun controller n'est passé en parametre, cela va chercher l'index.
Si le controller n'existe pas, le 404 est chargé

Modèle:

Classique, toutes les tables de la base de données ont une représentation en objet. Donc chaque modele de donnée se retrouve dans modele/class/monobjet.modele.class.php

Tout les objets « modeles » sont des extends de la class « bdd », ce qui permet une initialisation rapide ainsi qu'un enregistrement sans tapper de sql (en général).

Vue:

La vue est appelée depuis le controller, on lui passe des variables en instanciant une vue (view) depuis le controller.

Depuis le controller, il faut faire un
$view = new view("admin", "auth", "admin.notconnected.layout");
$view->assign("variable", « contenu de ma variable");

Le premier paramètre est le controller, le second l'action en cours, la troisième le layout.

Le layout contient donc en général la trame de page (menu, footer, head …).
L'action en cours est incluse dans ce layout.

Les objets, sauvegarde, instanciation:

$monobject->save(« matable »);

instance un objet depuis le base de donnée:

$monobject-> new object;
$monobject->getOneBy(« valeur a chercher », « colonne de la base », « table de bdd », « order »);
$monobject->setFromBdd($monobject->result);

J'ai maintenant accès a mon objet et a ses methodes.

Pour faire une requête et récupérer plusieurs résultats :
$monobject-> new object;
$mesobjets = $monobject->getResults(« valeur a chercher », « colonne de la base », « table de bdd », « order »);

J'ai accès avec un while a mes objets:
foreach ($mesobjets as $key => $value) {

}

Pour faire des requêtes un peu spéciales :

$monobject->requete(« ma requête sql »);

Pour enlever les accents d'une chaine de caractere:
fonctions::remove_accents

pour checker si un tableau est serialize en base (mais en théorie c'est dan sl'instanciation des objets) :
fonctions::is_serialized

pour checker un mail :
fonctions::valideEmail

pour sanitizer une chaine:
fonctions::sanitize