Objet : documentation technique cetcal.site - Apprendre à maintenir et créer une nouvelle page sur www.cetcal.site

Auteur	Date	Version
Thomas	3 juillet	Initiale v0.1

A propos de ce document

Pour faciliter une intégration développeur ou informaticien au projet, **ce document prend la forme d'un didacticiel**.

Codes couleurs (ce document utilise un code couleur pour faciliter sa lecture). Voici :

- text simple.
- **Elément important à retenir ou noter.**
- **Travaux pratique ! Ou TODO (chose à faire) pour suivre le didacticiel sereinement.**

Introduction projet

Projet **CETCAL ou *Castillonnais en transition, Circuits Alimentaires Locaux***.

Pour en savoir plus sur le projet : demander la documentation et les spécifications existantes.

Tour d'horizon technique

Langages / technologies utilisées :

- **Développement web – langages et frameworks :** Php 7.2, Jquery, Javascript natif, Bootstrap 4 (css responsive), sql MariaDb et MySql (quasi identique). PHPMailer. Composer pour les dépendances.
- **Config côté serveur :** Php 7.2, serveur Web apache, MySql (mariadb en environnement de développement local). PhpMyAdmin.
- **Le poste de développement local + repository :** https://github.com/j-fish/cetcal, Apache NetBeans pour le Php et la liaison avec le repository GitHub, Sublime Text pour le développement JS, Jquery, Php, Sql, Css, HTML5, Bootstrap 4. Config 100% Linux (dist Linux MINT).

Architecture générale

Tous **l'applicatif web est développé sur le model Modèle-Vue-Contrôleur**, ou **MVC**. Il faut vous familiariser avec cette architecture (wikipédia) si le MVC ne vous est pas familier.

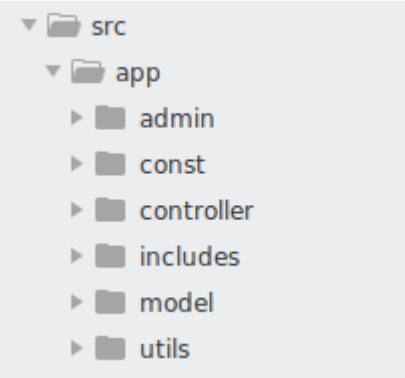
MVC dans les grandes lignes : découpper le code en couches indépendantes (principe du "*divide and conquer*"). La couche **Vue** contient l'HTML, et les variables Php (lecture seule). La couche **Modèle** contient le code d'interface avec la base de donnée ainsi que les DTO (Data Tansfer Objects : objets Php qui aident à véhiculer les données entre couches). La

couche **Contrôleur** contient le code de liaison entre la Vue et la couche Modèle – elle assure la navigation, le contrôle des données entrantes, sortantes, les règles de gestion.

Se renseigner sur les notions de MVC, DTO. L'exception est les DTO contenus dans la couche modèle : étant des objets conteneurs des varibales, cette couche est la seule qui est visible par toutes les autres couches.

En parralèle du MVC, une couche **utilitaire** et **constantes** sont utilisées dans une logique de "couteau suisse" (Lire des fichiers, traitement de chaines de caractères ect).

Voici un aperçu visuel (couche modèle = includes) :



Prise en main des Sources du projet

TODO : télécharger le code en format .zip depuis le github du projet (url dans chapître précédent). Sauvegarder le zip (et extraire le projet dans un répertoire de travail poste local si vous utilisez Sublime Text. Pour NetBeans : import projet depuis zip).

TODO : importer le projet dans Sublime TEXT ou NetBeans.

Le projet respecte les normes de développement Web :

- **code sources dans le répertoire /*src/**
- **Ressources (images, fichiers de données, bref tout élément qui n'est pas du code informatique) dans le répertoire /*res/***

Architecture détaillée (exemple de page web)

Toutes les pages du projet sont construites sur une logique modulaire. Ceci permet d'éviter les redondances de code et de rendre le code plus flexible au changement (entres autres). **Exemple de la page d'accueil de cetcal.site (les modules sont listés par ordre de chargement). Le fichier maître qui inclu ces chargements est et sera toujours index.php**

Nom du module dans index.php	Description	Couche MVC
cet.qstprod.startup.php	Gestion des actions communes à toutesles pages : la session entres autres est gérée dans ce module pour toutes les pages.	/
Import contrôleur associé à la page	Pour appel depuis la Vue	C
include.cet.qstprod.navbar.php	Barre de menu.	V
include.cet.qstprod.'.$statut.'.php	La variable $status = nom de la page encours	V
include.cet.qstprod.footer.php	Pied de page	V
include.cet.qstprod.modal.*.php	Import des popups de type fenêtre modale	V

Noter que dans le code d'une page ou de module, seule existe une référence à un contrôleur de la couche *Controller*. Il faudra continuer de respecter cette architecture dans les maintenances ou développements futurs.

TODO : Je vous propose maintenant d'ouvrir le fichier racine index.php qui gère tous les imports et la logique modulaire décrite ci-dessus.

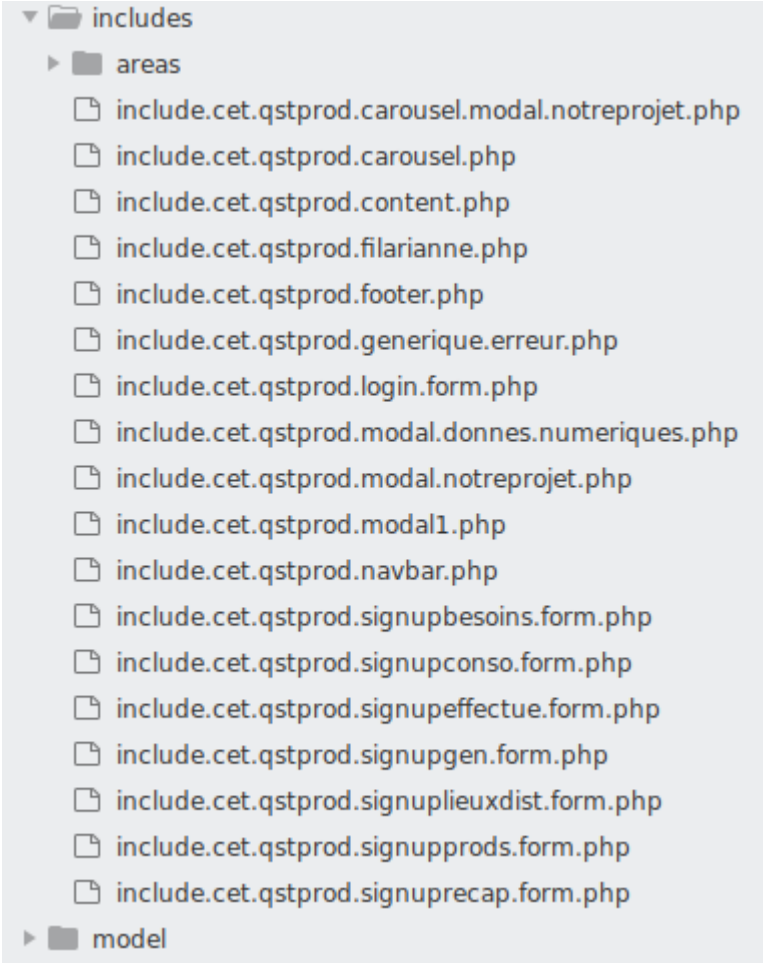
Créer un nouvelle page modulaire

Toujours créer la Vue en premier.

Respecter la nomenclature : `include.cet.qstprod.[nom].[de ma page].php`

include indique la couche Vue. Les préfixes cet.qstprod indiquent qui et quel projet – repectivement **cet=castillonnais en transition, qstprod=questionnaire producteurs**.

Dans le projet voici toutes les Vues à ce jour :



TODO : créer “mapagetest” dans la couche Vue (dossier includes ci-dessus).

Résultat attendu : `include.cet.qstprod.mapagetest.php`

TODO :

- **ajouter du code html à la nouvelle page.**
- **Charger la page pour vérifier le résultat.** Pour cela, indiquer le module à charger via paramètre GET dans l’url. Copier cet url dans votre navigateur : http://localhost/?statut=mapagetest

Que c'est-il passé ? :

- Dans index.php, le paramètre GET *statut* est récupéré depuis l'url. Ensuite le module de la couche Vue est chargé automatiquement. **TODO : comprendre ce fonctionnement dans la page maître index.php**.

Couche Contrôleur : gérer la navigation

TODO : créer une nouvelle page dans la couche Vue pour naviguer depuis "mapagetest" (appliquer les pricipes appris).

TODO : créer le script php contrôleur. Respecter la nomenclature `cet.qstprod.controller.mapagetest.php`

TODO : ajouter un lien <a href=> à la nouvelle page pour joindre le contrôleur de "mapagetest". Le href du lien = $_SERVER['DOCUMENT_ROOT'].'/src/app/controller/cet.qstprod.controller.mapagetest.php'

Couche DTO

Dans la couche Contrôleur

Dans la couche Modèle

Bonnes pratiques