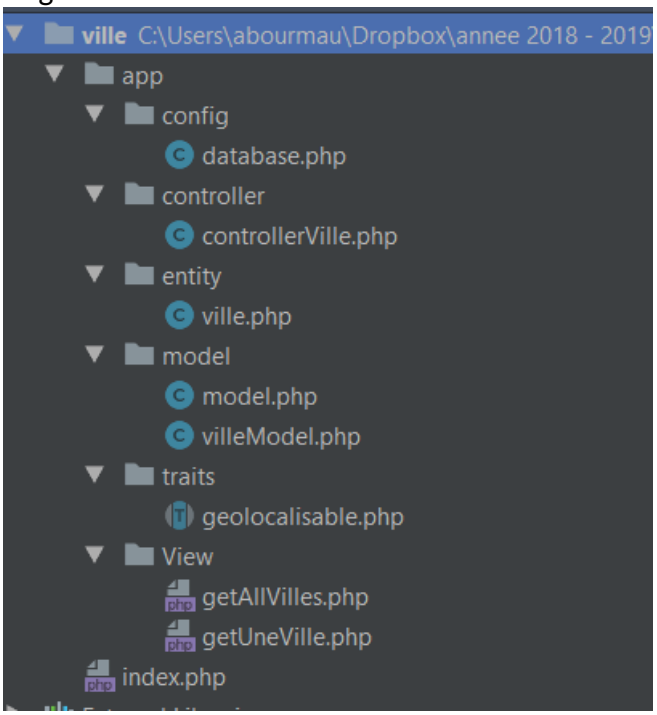
Architecture et dev web
TP 8 - PHP Objet

Lors de ce Tp, nous souhaitons :

- réutiliser les concepts de classes, d'hydratation, d'autoload vus lors du TP6
- réutiliser les concepts d'organisation, de namespace et de traits vus lors du TP7
- découper notre projet en MVC

Le but est d'afficher des données stockées en BD sur des Villes.

Vous trouverez ci-dessous l'organisation des dossiers et des fichiers :



Toutes les classes ainsi que e trait auront un namespace correspondant à leur chemin dans les fichiers.

Le fichier index possédera la fonction autoload vue précédemment et sera appelée ici.

I. Création de la base de données

- Créez la base de données « voyage »
- Importez dans cette BD, le fichier voyage.sql du réseau

II. Trait Geolocalisable

- Récupérez le trait de la semaine dernière géolocalisable

III. Database

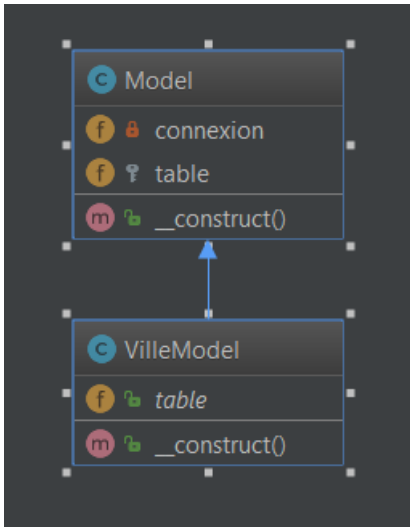
- Récupérez la classe DataBase du TP6 et modifiez le nom de la base de données

IV. Classe Ville

- Dans le dossier entity, créez la classe ville.
- Cette classe possède les attributs (id, nom, population et pays_id)
- Cette classe utilise le trait
- Cette classe possède la méthode hydrate et utilise cette méthode dans le constructeur
- Cette classe redéfinie la méthode __toString() permettant de faire afficher les données de celle ci
- Testez dans l'index cette classe

V. les classes du dossier Model

Nous souhaitons créer nos deux classes ainsi :

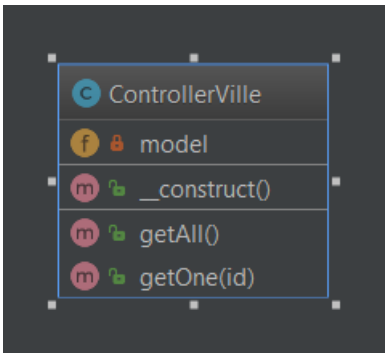


La classe VilleModel va hériter de la classe Model. La classe Model correspond à notre SCRUD (Search, Create, Read, Update, Delete)

- Créez la classe Model qui possède les 2 attributs connexion et table.
- Créez le constructeur qui permet d'initialiser la connexion en utilisant la classe DataBase et sa fonction de connexion.
- Créez une fonction généraliste find() (select *) qui renvoie un tableau associatif de tous les enregistrements de n'importe qu'elle table. Votre requête devra récupérer tous les champs de la table $this→table
- Créez la classe VilleModel qui hérite de la classe Model
- Créez son constructeur qui utilise le constructeur parent et qui initialise l'attribut table avec le nom de la classe associée ici « ville »
- Créez la méthode findAll() qui appelle la méthode find() du model et qui renvoie un tableau d'objet de type Ville

VI. ControllerVille

- Créez la classe ControllerVille.



- Cette classe permettra d'appeller le modèle VilleModel via l'attribut model
- Créez la méthode getAll() qui utilise la méthode findAll() de la classe VilleModel. Cette méthode inclura (include ou require) la vue getAllVilles.php

VII. View

- Créez le fichier getAllVilles.php qui correspond à l'affichage sous forme de liste des différentes villes stockées dans la base de données.

VIII. Index.php

- Créez un controller et utilisez la méthodes getAll().

IX. Ajout

Vous devez à présent avoir la vue de toute la base de données. Nous souhaitons maintenant pouvoir voir uniquement le contenu d'une Ville. Le but est d'ajouter une méthode getOne($id)
au ControllerVille

- Ajoutez cette méthode qui fera appel à la méthode findOne($id) de ModelVille qui fera elle même appel à la méthode read($id) de la classe Model
- testez

X. Routage

Selon l'url, nous souhaitons maintenant utiliser la méthode du contrôleur

- pour l'url index.php alors on redirige vers le getAll() du contrôleur
- pour l'url index.php?id=2 alors on redirige vers getOne() du contrôleur

Implémentez ce système de routage dans l'index.php

XI. sélection d'un pays

- Nous souhaitons faire appel à la méthode getAllPays($paysID) lorsque l'url sera index.php?pays=1

Il serait intéressant de créer dans la classe Model,une fonction généraliste nommée find(array $data) qui selon les $data réalisera une requête différente.

Exemple du tableau $data

```
$data=array(
                    "conditions"=>
                    "fields"=>
                    "limit"=>
                    "order"=>
                    "othertable"=>
)
```

- Créez cette méthode et testez