TP 01 – MongoDB

Création d'une application CRUD avec MongoDB et PHP

---

1. Objectifs du TP

L'objectif de ce premier travail pratique est de découvrir **MongoDB**, une base de données NoSQL orientée documents, et de l'utiliser dans une application web PHP.

Les objectifs principaux étaient les suivants :

- Découvrir le fonctionnement de MongoDB et son modèle de données
- Manipuler MongoDB via une interface graphique et via le shell
- Importer un jeu de données réel
- Utiliser MongoDB depuis une application PHP
- Implémenter un **CRUD complet**
- Mettre en place une **pagination** et une **recherche** (bonus)

---

2. Présentation de MongoDB

MongoDB est une base de données **orientée documents**.
Contrairement aux bases relationnelles (MySQL, PostgreSQL), elle ne repose pas sur des tables et des relations, mais sur :

- des **bases**
- des **collections**
- des **documents** au format BSON (JSON enrichi)

Chaque document peut avoir une structure flexible, ce qui permet une grande souplesse dans la modélisation des données.

Exemple de document MongoDB utilisé dans ce projet :

```
{
  "_id": ObjectId("65f8a4c8e12d4a0012345678"),
  "titre": "Chroniques médiévales",
  "auteur": "Anonyme",
  "siecle": "XIV"
}
```

---

3. Mise en place de l'environnement

3.1 Utilisation de Docker

L'environnement de travail repose sur Docker Compose, ce qui permet de garantir :

- un environnement reproductible
- une isolation complète des services
- une configuration identique pour tous les étudiants

Les services utilisés pour ce TP sont notamment :

- MongoDB
- Mongo GUI
- PHP 8.3
- Twig
- Composer

MongoDB est accessible via Docker et visualisable grâce à Mongo GUI sur le port 4321.

3.2 Import des données OpenData

Les données utilisées proviennent de l'OpenData de Clermont Auvergne Métropole :

- Catalogue des manuscrits de la bibliothèque du patrimoine.

L'importation a été réalisée via l'outil mongoimport :

```
mongoimport --collection=tp \
mongodb://zz3f3:easyma@localhost:27017/tp \
--authenticationDatabase=admin \
--file=catalogue-manuscrits-bibliotheque-patrimoine.json \
--jsonArray
```

Cette commande permet d'importer directement un fichier JSON dans une collection MongoDB.

---

4. Connexion à MongoDB dans l'application PHP

4.1 Centralisation de la configuration

La connexion à MongoDB est centralisée dans le fichier init.php. Les paramètres de connexion sont stockés dans un fichier .env, ce qui permet :

- de ne pas exposer les identifiants
- de modifier la configuration sans toucher au code

4.2 Fonction de connexion MongoDB

```php
function getMongoDbManager(): Database
{
    $client = new MongoDB\Client("mongodb://{$_ENV['MDB_USER']}:
{$_ENV['MDB_PASS']}@{$_ENV['MDB_SRV']}:{$_ENV['MDB_PORT']}");
    return $client->selectDatabase($_ENV['MDB_DB']);
}
```

Cette fonction :

- crée un client MongoDB
- sélectionne la base de données
- retourne un objet Database utilisable dans toute l'application

5. Architecture de l'application

L'application suit une architecture simple :

- PHP pour la logique métier
- Twig pour les templates HTML
- MongoDB pour le stockage des données

Chaque action du CRUD correspond à un fichier PHP :

- app.php : liste des livres
- create.php : création
- get.php : consultation
- update.php : modification
- delete.php : suppression

6. Implémentation du CRUD

6.1 Affichage de la liste des livres

La liste des livres est récupérée depuis MongoDB et transmise au template Twig.

```
$list = $manager
    ->selectCollection('tp')
    ->find([], ['sort' => ['_id' => -1]])
    ->toArray();
```

Dans le template Twig :

```
{% for document in list %}
<tr>
    <td>{{ document.titre }}</td>
    <td>{{ document.auteur }}</td>
    <td>{{ document.siecle }}</td>
</tr>
{% endfor %}
```

6.2 Création d'un livre

La création se fait via un formulaire HTML envoyé en POST.

```
$book = [
    'titre' => $title,
    'auteur' => $author,
    'siecle' => $siecle,
];

$manager->selectCollection('tp')->insertOne($book);
```

MongoDB génère automatiquement l'identifiant _id.

6.3 Consultation d'un livre

La consultation d'un livre se fait à partir de son identifiant :

```
$entity = $collection->findOne([
    '_id' => new ObjectId($_GET['id'])
]);
```

Cette fonctionnalité permet d'afficher les détails complets d'un manuscrit.

6.4 Mise à jour d'un livre

```
$collection->updateOne(
    ['_id' => new ObjectId($id)],
    ['$set' => [
        'titre' => $title,
        'auteur' => $author,
        'siecle' => $siecle
    ]]
);
```

MongoDB met à jour uniquement les champs concernés, sans modifier le reste du document.

6.5 Suppression d'un livre

```
$collection->deleteOne([
    '_id' => new ObjectId($_GET['id'])
]);
```

La suppression est déclenchée via un bouton dans la liste principale.

7. Pagination (Bonus)

Afin d'améliorer les performances et l'ergonomie, une pagination a été ajoutée.

```php
$page = $_GET['page'] ?? 1;
$perPage = 15;

$collection->find([], [
    'skip' => ($page - 1) * $perPage,
    'limit' => $perPage
]);
```

Cette approche évite de charger toute la collection en mémoire.

8.Conclusion du TP1

Ce premier TP a permis de :

- Comprendre le fonctionnement de MongoDB
- Mettre en œuvre une base NoSQL dans une application réelle
- Implémenter un CRUD complet
- Découvrir l'intérêt de la pagination et du filtrage
- Comparer l'approche NoSQL avec les bases relationnelles classiques

Cette base applicative a ensuite servi de fondation pour les TPs suivants, notamment l'ajout d'un cache Redis et d'un moteur de recherche ElasticSearch.