

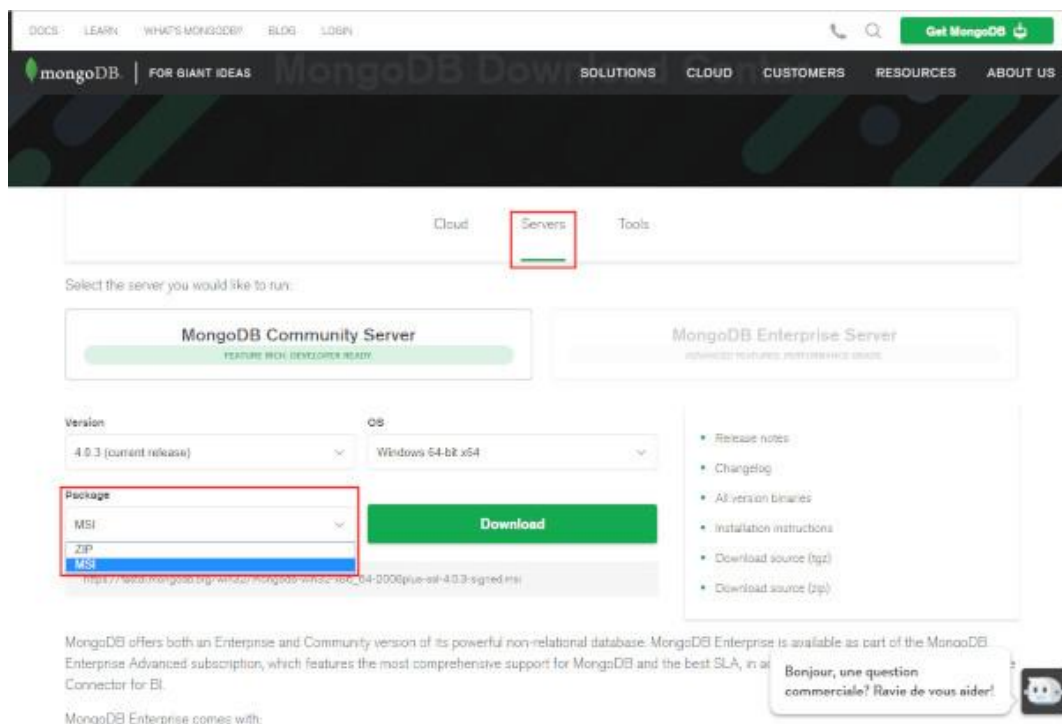
# Les bases de données NoSql : MongoDB

MongoDB est un système de base de données NoSql orientée *documents*. Le nom *Mongo* vient de l'anglais *humongous* qui signifie *énorme*.

MongoDB fonctionne sur le port 27017 et se manipule en lignes de commandes. Pour exécuter des requêtes, MongoDB utilise le langage Javascript. Les données sont stockées au format *BSON*, du JSON (revoir le cours sur AJAX) converti en binaire.

## Installation de MongoDB

- [Téléchargez MongoDB](#) : choisir l'onglet *Servers*, choisir *Community Server* pour Windows 64 et le **package MSI**.



- Suivre les instructions par défaut de l'installateur (cliquez sur le bouton **Complete**)
- Pour démarrer MongoDB, lancez le terminal Windows (touches **Windows** + **R**, puis taper **cmd**), saisir la commande suivante (changer l'emplacement et la version si nécessaire) :

```
cd C:\Program Files\MongoDB\Server\4.2\bin
```

Puis la commande de démarrage du **serveur Mongo** :

```
mongod
```

et pour terminer la commande de lancement de **l'interpréteur de commande** :

```
mongo
```

## Exécuter des requêtes

[Installez](#) la base MySQL **voitures** dans PhpMyAdmin et observez les relations entre les tables.

Ressources :

- [Collections](#)
- [CRUD](#)
- [Tutorialspoint](#)
- [Commandes de base](#)

## Créer une base de données

Pour créer et/ou utiliser une base MongoDB, saisir la commande **use** suivie du nom de la base de données (ici, la base *vehicules*) :

```
use voitures
```

Si la base n'existe pas, elle sera créée.

## Créer une collection

**Une collection** correspond à une table et se crée via la commande :

```
db.createCollection("modeles")
```

**modeles** est ici le nom de la collection.

Cette commande peut comporter [des options](#).

Il s'agit ici de la commande explicite de création d'une collection, mais un simple *insert* (voir point suivant) permet de créer une collection si elle n'existe pas encore.

# Insérer/modifier/supprimer un document

Un **document** correspond à un enregistrement dans une collection (ajout d'une ligne dans une table).

## Insérer un document

Un **document** correspond à un enregistrement dans une collection (ajout d'une ligne dans une table) :

Un 1er enregistrement :

```
db.modeles.insert({ "mod_nom": "C3",  
  "mod_marque": "Citroën",  
  "mod_cylindree": 1.0,  
  "mod_prix": 13500,  
  "mod_date_ajout": new Date(),  
})
```

Un second enregistrement :

```
db.modeles.insert({ "mod_nom": "Cactus",  
  "mod_marque": "Citroën",  
  "mod_cylindree": 1.0,  
  "mod_prix": 18470,  
  "mod_date_ajout": new Date(),  
  "mod_d_dispo": new Date("2018-10-20"),  
  "mod_options": [ "GPS", "Toit ouvrant", "Peinture métallisée"  
]  
})
```

## Chaînes et entiers

- Les chaînes sont entre guillemets (simples ou doubles)
- Les entiers et décimaux ne sont pas entre guillemets (colonne `mod_cylindree`)

## Identifiants

Un identifiant unique nommé `_id` est créé automatiquement pour chaque document (équivalent des clés primaires d'un SGBDR). Il est toutefois possible de le spécifier explicitement.

## Clés étrangères

Les clés étrangères **n'existent plus** : en MySQL, la table `modeles` présente la colonne `mod_mar_id` dont la valeur renvoie à sa correspondance dans la table `marques`; en MongoDB, on stocke la valeur de la marque (`Citroën`) directement dans le document, et celle-ci sera répétée autant de fois qu'il y a d'occurrences (s'il y a 25

modèles de Citroën, on écrira 25 fois "mod\_marque": "Citroën" dans la collection voitures. Dans notre exemple la marque Citroën.

## Dates

- pour insérer la date du jour (par exemple pour la colonne `mod_date_ajout` : {  
`date: new Date()` }
- pour insérer une date avec une valeur : `new Date("2018-10-18")`, `new Date("2018-10-18T15:31:45")` (**Attention : ne pas oublier le T**).
- [documentation officielle](#)

## Valeurs nulles

Les colonnes avec une valeur nulle (`mod_date_dispo`) n'ont pas à être stockées : le champ n'existe pas pour l'occurrence ; MongoDB ne sortira que les occurrences pour lesquelles le champ existe.

## Valeurs uniques

Pour la colonne `mod_nom`, la valeur est unique (= un seul modèle peut porter ce nom, toutes marques confondues). En MongoDB, on spécifie l'unicité d'une valeur via la méthode `createIndex()` :

```
db.voitures.createIndex( { "mod_nom": 1 }, { unique: true } )
```

 (le 1 de `mod_nom` signifie ordre ascendant)

## Tableaux

On peut spécifier un tableau, c'est-à-dire une liste de valeurs, en remplacement d'informations contenues dans une table liée (jointure).

Par exemple, lier des commentaires à un article, ou, dans notre base *voitures*, spécifier les options disponibles pour un modèle.

1<sup>er</sup> cas : tableau comprenant une liste simple de valeurs :

```
"options": [ "GPS", "Toit ouvrant", "Peinture métallisée" ]
```

2<sup>ème</sup> cas : tableau comprenant une liste de paires clés/valeurs :

```
"options": [ { "opt_libelle" : "GPS", "opt_prix" : 450 },  
              { "opt_libelle" : "Toit ouvrant", "opt_prix" : 870 },  
              { "opt_libelle" : "Peinture métallisée", "opt_prix" : 275 }  
            ]
```

## Insertions multiples

Pour insérer plusieurs documents à la fois : cf. méthode [InsertMany\(\)](#).

Un tableau de ce type est appelé **sous-document** et peut recevoir un identifiant (`_id`).

# Modifier un document

On modifie un document avec la commande `update` combinées aux options `$set`, `$unset`. [exemples](#).

## Supprimer un document/un champ

On supprime un champ avec la commande `update` (bien `update` comme pour modifier) combinées aux options `$pop`, `$pull` et `$pullAll` : [exemples](#)

Pour supprimer un document complet : `db.nom_collection.remove()`, [exemples](#)

On peut aussi utiliser l'identifiant `_id` pour cibler un document en particulier.

## Sélectionner

### Sélectionner tous les documents d'une collection

```
db.nom_collection.find()
```

Equivalent SQL : `SELECT * FROM table`

### Sélectionner d'un document selon son identifiant

```
db.nom_collection.find( { _id : ObjectId("5bc9958add21f2c56cfa2d57") } )
```

**Attention à la casse : bien écrire `ObjectId` avec un O et un I majuscule.**

Equivalent SQL : `SELECT * FROM table WHERE id = 1`

### Sélectionner selon des critères précis

Par exemple, sélectionner les voitures de la marque *Citroën* :

Equivalent SQL : `SELECT * FROM table WHERE marque = 'Citroën'` ou, si tables liées, `WHERE mar_id = 1`

## Autres requêtes utiles

- `db` : afficher le nom de la base sur laquelle on travaille
- `show dbs` : afficher toutes les bases
- `db.dropDatabase()` : effacer la base sur laquelle on travaille
- `show collections` : liste les collections
- `db.oldname.renameCollection("newname")` : renommer une collection
- `db.nom_collection.drop()` : supprimer une collection (remplacer `nom_collection` par celle souhaitée)
- `db.nom_collection.count()` : compter le nombre de documents dans la collection
- `db.inventory.deleteMany({})` : supprimer tous les documents d'une collection (= vider une table en SQL)

Les bases de données NoSQL : MongoDB

Réalisation : Germain SIPIERE Formateur AFPA  
Guillaume DELACROIX Formateur AFPA  
13 avril 2023

**Afpa** 

- `db.collection.drop()` : supprimer l'intégralité d'une collection

## Outils

Il existe des interfaces utilisateur (équivalent de phpMyAdmin) permettant de gérer MongoDB, comme [Robo3T](#).

## Mise en pratique

- Installez MongoDB
- Installez Robot3T
- Créer une collection et des documents (4 ou 5 lignes) correspondant à la base de données de Jarditou.
- Exécuter une requête :
  - de sélection
  - d'insertion
  - de modification
  - de suppression

## Connexion PHP

Les lignes de commandes c'est sympa, mais on ne va pas pouvoir faire ça sur un site web ! Voyons comment utiliser MongoDB dans un projet PHP :

- [Exemple](#)