

[<- retour aux formations](#)

PLAN DU COURS



21 octobre 2021



117.028K

0

FORMATION LARAVEL : QUERY BUILDER

Dans les 3 chapitres précédents (Formulaire, Request et Validation) nous avons vu comment manipuler des données entrantes. Maintenant nous allons voir comment manipulant des données sortant de notre base de données. Nous allons garder notre exemple avec les livres.

Les pré-requis pour commencer ce chapitre sont de :

- créer une table 'books' (cf chapitre Migrations) avec les colonnes suivantes :
 - 'title' en char limité à 255
 - 'price' en decimal 4,2
 - 'author' qui sera la clé étrangère pour la table 'authors'
 - 'description' en char limité à 255 avec la valeur par défaut null
 - 'cover' en char limité à 255 avec la valeur par défaut null
- créer un model Book (cf chapitre sur Eloquent)
- créer un controller resource BookController (cf chapitre sur les Controllers)
- peupler la table books de la base de données (cf chapitre sur les Seeders et Factories)



- Nous aurons également besoin de la table “authors” qui sera en relation avec la table “books”

‘books.authors’ = ‘authors.id’ (cf chapitre sur les Relations).

Cela vous permettra de mettre en pratique rapidement les notions vues précédemment.

1. Sélectionner des résultats

Comme nous l’avons vu précédemment le Query Builder de Laravel est accessible via la façade DB et vous permet de sélectionner une table puis d’y chaîner des méthodes afin de modeler votre requête SQL comme bon vous semble.

PHP

```
// Rappel de la structure de départ
DB::table('books') // je prépare une requête pour la table 'books'
```

Nous allons maintenant voir quelques méthodes utiles concernant la sélection de données :

• get()

La méthode `get()` permet de récupérer un groupe de résultat. Elle retourne une Collection (de la Class `Illuminate\Support\Collection`) contenant chaque résultats obtenu par la requête. Chacun de ces résultats est une instance de la Class vide générique de PHP : `stdClass`.

Voici un aperçu de ce que cela donne :

PHP

```
$books = DB::table('books')->get();
dd( $books );
```



```

Collection {#322 ▼
  #items: array:10 [▼
    0 => {#329 ▼
      +"id": 1
      +"title": "Le monde comme volonté et comme représentation"
      +"price": "19.00"
      +"author": "1"
      +"description": "Lorem Ipsum..."
      +"cover": null
    }
    1 => {#331 ▼
      +"id": 2
      +"title": "Métaphysique de la mort, Métaphysique de l'amour"
      +"price": "1.00"
      +"author": "1"
      +"description": "Lorem Ipsum..."
      +"cover": null
    }
    2 => {#332 ►}
    3 => {#333 ►}
    4 => {#334 ►}
    5 => {#335 ►}
    6 => {#336 ►}
    7 => {#337 ►}
    8 => {#338 ►}
    9 => {#339 ►}
  ]
}

```

• where()

La méthode `where()` permet de poser une condition à la selection de résultat :

PHP

```

$expensives = DB::table('books')->where('price', '>', 20)->get();
dd( $expensives );

```

```

Collection {#322 ▼
  #items: array:3 [▼
    0 => {#329 ▼
      +"id": 8
      +"title": "Métamorphoses de l'âme et ses symboles"

```



```

        +"price": "20.50"
        +"author": "2"
        +"description": "Lorem ipsum..."
        +"cover": null
    }
    1 => {#323 ▶}
    2 => {#331 ▶}
]
}

```

• first()

La méthode `first()` a un fonctionnement similaire à `get()` mais ne récupère que le premier résultat correspondant à la requête.

PHP

```

$book = DB::table('books')->first();
dd( $book );

```

```

{#329 ▼
  +"id": 8
  +"title": "Métamorphoses de l'âme et ses symboles"
  +"price": "20.50"
  +"author": "2"
  +"description": "Lorem ipsum..."
  +"cover": null
}

```

S'il ne trouve aucun valeur `first()` retourne la valeur null.

• find()

La méthode `find()` permet de sélectionner un résultat d'après un id que l'on passe en paramètre :

PHP



```

$book = DB::table('books')->find(1);

```

```
dd( $book );
```

```
{#329 ▼
  +"id": 1
  +"title": "Le monde comme volonté et comme représentation"
  +"price": "19.00"
  +"author": "1"
  +"description": "Lorem Ipsum..."
  +"cover": null
}
```

2. Rangement et groupement

• orderBy()

Permet d'organiser les résultats en les rangeant dans un order ascendant (ASC) ou descendant (DESC) d'après une colonne de la table :

PHP

```
$books = DB::table('books')->orderBy('price', 'DESC')->get();
dd( $books );
```

```
Collection {#322 ▼
  #items: array:10 [▼
    0 => {#329 ▼
      +"id": 10
      +"title": "Névrose et Psychose"
      +"price": "21.99"
      +"author": "3"
      +"description": "Lorem Ipsum"
      +"cover": null
    }
    1 => {#332 ►}
    2 => {#332 ►}
    3 => {#333 ►}
    4 => {#334 ►}
```



```

5 => {#335 ▶}
6 => {#336 ▶}
7 => {#337 ▶}
8 => {#338 ▶}
9 => {#339 ▶}
]
}

```

• groupBy()

La méthode `groupBy` permet de grouper les résultats par valeur(s) d'une ou plusieurs colonnes.

PHP

```

$books = DB::table('books')->groupBy('author')->get();
dd( $books );

```

Collection {#322 ▼

#items: array:3 [▼

0 => {#323 ▼

```

+"id": 1
+"title": "Le monde comme volonté et comme représentation"
+"price": "19.00"
+"author": "1"
+"description": "Lorem ipsum..."
+"cover": null

```

}

1 => {#331 ▼

```

+"id": 4
+"title": "Dialectique du moi et de l'inconscient"
+"price": "19.99"
+"author": "2"
+"description": "Lorem ipsum..."
+"cover": null

```

}

2 => {#332 ▼

```

+"id": 6
+"title": "L'inconscient"
+"price": "7.50"
+"author": "3"
+"description": "Lorem ipsum..."
+"cover": null

```

}



```
]
}
```

Ici on se retrouve avec 3 résultats qui correspondent aux 3 premières concordances pour chaque nouvel auteur. Vous pouvez, si vous le souhaitez, entrer plusieurs paramètres dans cette méthode `groupBy` afin de faire un groupement par plusieurs colonnes.

Attention, en utilisant cette méthode avec une connexion via `mysql` il se peut que vous obteniez une erreur « Syntax error or access violation ». C'est que vous avez probablement une version de `mysql` supérieure à la 5.7.5, je vous invite donc à lire la documentation officielle de `mysql`.

• latest() et oldest()

Les méthodes `latest()` et `oldest()` rangent les données par date en prenant comme référence par défaut la colonne `'created_at'`. Si vous voulez appliquer ces méthodes à une autre colonne vous n'avez qu'à passer en paramètre le nom de celle-ci. Ici je n'ai pas mis de timestamps mais nous pouvons par exemple appliquer cette méthode avec la colonne `'id'`.

PHP

```
$books = DB::table('books')->latest('id')->first();
dd( $books );
```

```
{#331 ▼
  +"id": 10
  +"title": "Névrose et Psychose"
  +"price": "21.99"
  +"author": "3"
  +"description": "Lorem ipsum..."
  +"cover": null
}
```



On récupère en premier résultat l'id 10, nous avons 10 livres donc cela nous retourner bien le dernier résultat enregistré dans la base de données.

• inRandomOrder()

La méthode `inRandomOrder()` vous permet tout simplement de sortir vos résultats dans un ordre aléatoire.

3. Les jointures

C'est ici que notre table 'authors' va nous servir. J'espère que vous n'avez pas oublié de la créer 😊

• join()

La méthode `join()` permet de faire une jointure **'INNER JOIN'**.

PHP

```
$book = DB::table('books')->join('authors', 'books.author', '=', 'authors.id')->first()  
dd($book);
```

```
{#336 ▼  
  +"id": 1  
  +"title": "Le monde comme volonté et comme représentation"  
  +"price": "19.00"  
  +"author": 1  
  +"description": "Lorem Ipsum..."  
  +"cover": null  
  +"firstname": "Arthur"  
  +"lastname": "Schopenhauer"  
  +"birth": "1788-02-22"  
  +"death": "1860-09-21"  
  +"biography": "Lorem Ipsum..."  
}
```



On remarque que les colonnes correspondant à l'auteur de ce livre ont été ajoutées à notre objet de la class stdClass.

• leftJoin() et rightJoin()

Les méthodes leftJoin() et rightJoin() permettent d'effectuer les commandes SQL de jointure '**LEFT JOIN**' et '**RIGHT JOIN**'.

• crossJoin()

La méthode crossJoin() permet d'effectuer la commande SQL de jointure '**CROSS JOIN**'.

4. Insert, Update et Delete


• insert() et insertGetId()

La méthode insert() permet d'insérer une ligne dans une table. La méthode insertGetId() fait le même job que insert() mais en plus vous permet de récupérer l'id créé pour cette enregistrement. Bien évidemment insertGetId() ne marche qu'avec des tables ayant un id en auto-incrémentation.

PHP

```
$insert = DB::table('authors')->insertGetId([
    'firstname' => 'Jean-Paul',
    'lastname'  => 'Sartre',
    'birth'     => '1905.06.21',
]);
dd($insert); // nous retourne : 4
```

• update()

La méthode update() permet de mettre à jour des données. Par exemple  dans l'exemple du dessus, nous avons oublié d'ajouter la date de décès de

l'auteur (oui c'est pas très joyeux mais c'est pour l'exemple ! ^^)

```
DB::table('authors')->where('id', 4)->update([
    'death' => '1980.05.15'
]);
```

• updateOrCreate()

La méthode `updateOrCreate()` permet de chercher un enregistrement pour le mettre à jour, mais si celui-ci n'existe pas et bien elle le crée. La méthode prend en paramètre 2 tableaux, le premier sert de conditions pour trouver l'enregistrement en question, le second permet de préciser les données à mettre à jour.

PHP

```
DB::table('authors')->insertOrUpdate(
    [ 'firstname' => 'Carl Gustav', 'lastname' => 'Jung' ],
    [ 'biography' => 'Lorem Ipsum...' ]
);
```

• incrementing() et decrementing()

Les méthodes `incrementing()` et `decrementing()` permettent respectivement d'incrémenter ou de décrémenter une valeur numérique d'un enregistrement.

• delete()

La méthode `delete()` permet de supprimer un enregistrement

PHP

```
DB::table('authors')->where('id', 4)->delete();
// Nous venons de supprimer l'enregistrement d'id 4 dans notre table 'authors'
```



5. Opérations numérique

Vous pouvez également appliquer des opérations numériques existantes dans les commandes SQL :

PHP

```
$max_price = DB::table('books')->max('price');  
$min_price = DB::table('books')->min('price');  
$average = DB::table('books')->avg('price');  
$sum = DB::table('books')->sum('price');
```

À savoir !

Nous venons de voir un ensemble de méthode appliquées à la façade DB mais sachez que ces méthodes sont également disponible via le model en question (ici Book par exemple).

Donc les 2 lignes de codes suivantes sont possibles :

PHP

```
$via_model = Book::where('id', 1)->first();  
$via_facade_db = DB::table('books')->where('id', 1)->first();
```

Mais ne revoient pas la même chose attention ! Via le model cela vous retourne un objet de la class Book et via la façade DB cela vous retourne un objet de la class stdClass.

PHP

```
dd( $via_model, $via_facade_db );
```

```
Book {#366 ▼  
  #guarded: []  
  +timestamps: false  
  #connection: "mysql"  
  #table: "books"  
  #primaryKey: "id"
```



```

#keyType: "int"
+incrementing: true
#with: []
#withCount: []
#perPage: 15
+exists: true
+wasRecentlyCreated: false
#attributes: array:7 [▼
  "id" => 1
  "title" => "Le monde comme volonté et comme représentation"
  "price" => "19.00"
  "author" => 1
  "description" => "Lorem Ipsum..."
  "cover" => null
]
#original: array:7 [▶]
#changes: []
#casts: []
#dates: []
#dateFormat: null
#appends: []
#dispatchesEvents: []
#observables: []
#relations: []
#touches: []
#hidden: []
#visible: []
#fillable: []
}

```

```

{#331 ▼
  +"id": 1
  +"title": "Le monde comme volonté et comme représentation"
  +"price": "19.00"
  +"author": 1
  +"description": "Lorem Ipsum..."
  +"cover": null
}

```

Voilà nous venons de voir diverses méthodes dans ce chapitre. Cette liste est non-exhaustive, je vous invite à vous rendre sur la documentation officielle de Laravel pour en connaître d'avantage. Nous venons cependant d'en voir suffisamment



pour que vous commenciez à être bien familiarisé avec le QueryBuilder et à comprendre son fonctionnement. 😊

Passons au chapitre suivant !

Chapitre précédent

Chapitre suivant

BESOIN D'UNE FORMATION PERSONALISÉE ?

Contactez-moi ici


Avis

4.9

Votre note

Votre avis





Votre nom

Envoyer votre avis

18 novembre 2022

Merci pour le partage 😊

—Dimitri

3 août 2022

Très complet, merci

—Christian

25 mai 2022

L'un des meilleurs cours de laravel !

Merci bien

—Estimé Gliti

13 avril 2022

Le cour est bien représenté et bien détaillé.

Merci.

—Petit-Homme wadly



9 avril 2022

Merci

—Anis kadri

Envie de travailler avec moi ?

Commencez votre projet

© 2018 - 2023. Walker Spider, Agence de développement web
Conditions générales de ventes - Politique de confidentialité - Archives

