Eloquent ORM de Laravel

1. Objectifs

- A la fin de cet atelier vous serez capables de :
  - Comprendre et utiliser un Model
  - Créer et exécuter une Migration
  - Créer et exécuter un Seeder

2. Qu'est-ce que Laravel Eloquent?

- L'une des caractéristiques les plus importantes du framework laravel est qu'il est livré avec **ORM** (**O**bject **R**elation **M**apping) intégré appelé Eloquent **ORM**.
- Eloquent Object Relation Mapper est une fonctionnalité importante du framework Laravel, considérée comme une fonctionnalité puissante et coûteuse de Laravel.
- L'**ORM** fonctionne avec des objets de base de données et est utilisé pour établir des relations avec des tables de base de données. Chaque table de la base de données est mappée avec un **modèle** éloquent particulier.
- L'objet modèle contient diverses méthodes pour récupérer et mettre à jour les données de la table de base de données.
- Eloquent ORM peut être utilisé avec plusieurs bases de données en implémentant `ActiveMethod`. Cette fonctionnalité simplifie les tâches liées à la base de données, telles que la définition de relations, en définissant les tables de la base de données.
- Les opérations **CRUD** sous le mappeur objet-relationnel Eloquent (**ORM**) permettent aux développeurs Laravel de travailler plus facilement avec plusieurs bases de données.
- Eloquent **ORM** rend facile l'exécution des opérations CRUD (Créer, Lire, Mettre à jour, Supprimer) sur le modèle Laravel.
- Il effectue des opérations de création, de récupération, de mise à jour et de suppression (CRUD), ainsi que les mappages de modèles d'objets sur des tables de base de données. Il gère toutes les interactions de base de données requises pour les opérations CRUD.

3. Comment fonctionne Eloquent?

- Les développeurs peuvent travailler efficacement dans **Eloquent** avec plusieurs bases de données en utilisant une implémentation ActiveMethod.
- Il s'agit d'un modèle architectural où le modèle créé dans la structure **M**odel-**V**iew-**C** ontroller (**MVC**) correspond à une table dans la base de données.

apcpedagogie.com

- L'avantage est que les modèles effectuent des opérations de base de données courantes sans coder de longues requêtes SQL.
- Les modèles permettent d'interroger des données dans vos tables, ainsi que d'insérer de nouveaux enregistrements dans des tables.
- Le processus de synchronisation de plusieurs bases de données s'exécutant sur différents systèmes est simplifié. Il n'est pas du tout nécessaire d'écrire des requêtes SQL.
- Tout ce que vous avez à faire est de définir les tables de la base de données et les relations entre elles, et Eloquent fera le reste du travail.

4. Configuration de Laravel

- Pour apprécier pleinement l'utilité d'**Eloquent ORM**, il est indispensable de comprendre l'écosystème.
- Voici les étapes pour commencer:
  1. Installez Laravel depuis getcomposer.org
  2. Créer des **migration**s à l'aide de la console Artisan
  3. Créez des **modèle**s éloquents
  4. "**Seed**" la base de données
- Artisan Console est le nom de l'interface de ligne de commande fournie avec **Laravel**. Il fournit un certain nombre de commandes utiles à utiliser lors du développement de votre application.

A. Migration

1. **Présentation**
   - La migration est un processus de gestion de votre base de données en écrivant PHP plutôt que SQL. Il fournit également un moyen d'ajouter un contrôle de version à votre base de données.
   - La migration est un type de contrôle de version pour votre base de données. Il permet à une équipe de modifier le schéma de la base de données et de rester à jour sur l'état actuel du schéma. La migration est généralement associée à **Schema Builder** pour gérer facilement le schéma de votre application.

     > *Les Migrations sont une sorte de contrôleur de version pour la BD, permettant, à une équipe, de modifier et partager le schéma d'une BD facilement.*

2. **Créer une migration**

- Pour créer une migration, vous pouvez utiliser la commande sur l'Artisan CLI: `php` **`artisan make:migration create_users_table`**
- La migration sera placée dans votre dossier **database/migrations** et contiendra un horodatage qui permet au framework de déterminer l'ordre des migrations.

3. **`Exécuter une migration`**
   - L'exécution d'une Migration permet de créer la table dans la BD avec le schéma défini dans la Migration. La commande permettant d'exécuter les Migrations est :**`php artisan migrate`**
   - Il existe une commande qui permet de créer une Migration avec le Model :
   - **`php artisan make:model --migration NomModel`**
   - Ou bien :
   - **`php artisan make:model -m NomModel`**

B. **Models**

1. **`Présentation`**
   - Une fois que vous avez terminé la migration de votre base de données, le processus suivant est l'amorçage(seeding).
   - L'éloquent entre en scène puisque l'amorçage insère des enregistrements dans notre base de données.
   - Vous devrez donc créer vos modèles avant de pouvoir amorcer la base de données.
   - Chaque table de base de données a un modèle correspondant qui est utilisé pour interagir avec cette table.
   - Les modèles vous permettent de rechercher des données dans vos tables, ainsi que d'insérer de nouveaux enregistrements dans la table.
2. **`Définition`**
   - L'ORM Eloquent inclus avec Laravel fournit un outil simple pour travailler avec la BD.
   - A chaque table dans la BD doit correspondre un Model qui sera utilisé pour interagir avec la table.
3. **`Construire des modèles`**
   - Commande Artisan pour créer un modèle Article : **`php artisan make:model Article`**

apcpedagogie.com

> - *Par convention, les modèles commencent tous par une **majuscule**, étant donné qu'ils sont représentés par des classes et que les classes commencent toutes par une majuscule.*
> - *Il faudra donc privilégier les noms en notation **CamelCase**, comme par exemple ArticleJardin, plutôt que article_jardin.*
> - *De plus, le nom de la classe sera toujours **au singulier**, ce qui signifie qu'un article sera représenté par Article plutôt que par Articles.*

- Le **modèle** créé est ajouté dans le dossier app de l'application Laravel. Il s'agit pour l'instant d'une classe sans méthodes ni propriétés propres qui hérite de la classe Model.
- Les modèles sont des classes qui héritent de la classe Model. Ils représentent les entités du site. Ils sont enregistrés dans le dossier app.

4. **`Les conventions d'Eloquent pour les Models`**
    1. **Nom de la table**
        - Par convention le nom du Model est le singulier du nom de la table qui lui correspond dans la BD. Si le nom de la table est différent, il faut le spécifier dans le model : **`protected $table = 'my_clients';`**
    2. **Clé primaire**
        - Eloquent suppose également que chaque table possède une colonne de clé primaire appelée id. Vous pouvez définir une propriété $primaryKey pour remplacer cette convention : **`public $primaryKey = 'ncin' ;`**
        - De plus, Eloquent suppose que la clé primaire est une valeur entière incrémentée, ce qui signifie que, par défaut, la clé primaire sera automatiquement convertie en int.
        - Si vous souhaitez utiliser une **clé primaire** non incrémentée ou non numérique, vous devez définir la propriété publique **$incrementing** sur votre modèle sur **false**. **`public $incrementing = false ;`**
    3. **Timestamps**
        - Par défaut, Eloquent s'attend à ce que les colonnes created_at et updated_at existent sur vos tables. Si vous ne souhaitez pas que ces colonnes soient gérées automatiquement par Eloquent, définissez la propriété $timestamps sur votre modèle sur false: **`public $timestamps = false;`**

apcpedagogie.com

C. Seeders

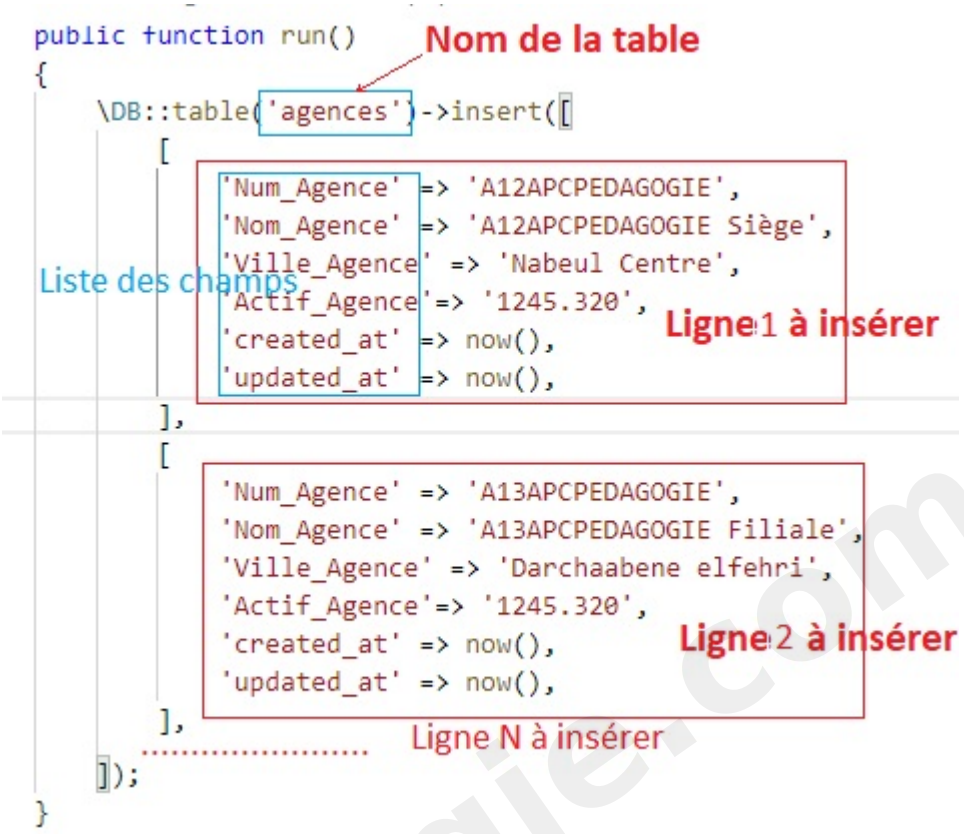
1. Présentation

- Laravel comprend une méthode simple pour remplir votre BD avec des données pour le test. Toutes les classes `Seed` sont stockées dans le répertoire **database/seeds**.
- En plus de proposer des migrations Laravel permet aussi la population (**seeding**), c'est à dire un moyen simple de remplir les tables d'enregistrements.
- Par convention, les noms des classes de Seeding sont écrits dans ce format : **UsersTableSeeder**.
- Pour générer un Seeder, il suffit d'exécuter la commande suivante : `php artisan make:seeder UsersTableSeeder`
- Une classe Seeder ne contient qu'une seule méthode par défaut: `run()`, dans laquelle vous pouvez insérer des données dans votre BD comme vous le souhaitez.
- Vous pouvez utiliser le Query Builder pour insérer manuellement des données ou vous pouvez utiliser les Eloquent Model Factories.
- Laravel intègre une bibliothèque très utile dans ce contexte nommé **Faker**, qui permet de générer des fausses données à insérer dans la BD.

2. Générer le fichier de seeds

- Un fichier de seeds sera créé à l'aide de la commande : **`php artisan make:seeder TablesTableSeeder`**
- Le fichier ainsi généré sera placé dans le dossier **database\seeds**, sous le dossier du projet.
- Éditer le fichier de seeds
- Éditez ce fichier pour y ajouter les données désirées à l'aide de **DB::table()**.

apcpedagogie.com



3. **Exécuter tous les fichiers de seeds du fichier DatabaseSeeder.php**

   - Pour exécuter tous les fichiers de seeds exécutez la commande suivante :`php artisan db:seed`

     > *Lorsqu'une table contient une clé étrangère, ses données initiales doivent être ajoutées après celles de la table contenant la clé primaire de la relation.*

   - Si une table contenait déjà des données, le fichier de seeds ajoutera les nouvelles données à la suite des données existantes.

Cours Laravel

Source : https://apcpedagogie.com/eloquent-orm-de-laravel · téléchargé le 08/11/2026 · © APC Pédagogie · apcpedagogie.com