Cours Laravel - Relations en Base de Données

Ce support présente le fichier fourni concernant les différentes relations en base de données avec Laravel Eloquent.

```php
<?php
// https://laravel.com/docs/12.x/eloquent-relationships
/*
Relations en Base de Données avec Laravel Eloquent

One to One (1:1)
---------------
Une entité est liée à une seule autre entité. Exemple : Un utilisateur a un profil (User -> Profile)
*/
// Un utilisateur a un profil.
// User.php
public function profile()
{
    return $this->hasOne(Profile::class);
}

// Profil.php
// Un profil appartient à un utilisateur.
public function user()
{
    return $this->belongsTo(User::class);
}
/*
One to Many (1:N)
-----------------
Une entité peut être liée à plusieurs autres. Exemple : Un utilisateur peut avoir plusieurs articles
*/
// Un utilisateur a plusieurs posts
public function posts()
{
    return $this->hasMany(Post::class);
}
/*
Many to One (N:1)
-----------------
L'inverse du One to Many : plusieurs entités appartiennent à une seule. Exemple : Plusieurs articles
*/
// Plusieurs posts appartiennent à un user.
// Post.php
public function user()
{
    return $this->belongsTo(User::class);
}

/*
Many to Many (N:N)
------------------
Deux entités peuvent avoir plusieurs occurrences liées via une table pivot. Exemple : Un utilisateur
*/
// Un utilisateur peut avoir plusieurs rôles
// User.php
public function roles()
{
    return $this->belongsToMany(Role::class);
}

/*
Has One Through
---------------
Une entité est reliée à une autre à travers une troisième. Exemple : Un utilisateur peut récupérer la
*/
// Un User peut récupérer la ville de son entreprise
// User.php
public function city()
{
    return $this->hasOneThrough(City::class, Company::class);
}

/*
```

```
Has Many Through
----------------
Une entité est liée à plusieurs autres à travers une table intermédiaire. Exemple : Un pays a plusieu
*/
// Un Country a plusieurs orders via ses users
// Country.php
public function orders()
{
    return $this->hasManyThrough(Order::class, User::class);
}

/*
Polymorphic One to One
----------------------
Une relation où une seule entité peut être liée à plusieurs types de modèles. Exemple : Une image peu
*/
// Une Image peut appartenir à un User ou à un Post.
// Image.php
public function imageable()
{
    return $this->morphTo();
}

// User.php et Post.php
public function image()
{
    return $this->morphOne(Image::class, 'imageable');
}

/*
Polymorphic One to Many
-----------------------
Un modèle peut avoir plusieurs entités polymorphiques. Exemple : Des commentaires peuvent être attach
*/
// Des comments peuvent être attachés à des posts ou des videos.
// Comment.php
public function commentable()
{
    return $this->morphTo();
}

// Post.php et Video.php
public function comments()
{
    return $this->morphMany(Comment::class, 'commentable');
}

/*
Polymorphic Many to Many
------------------------
Une entité peut être associée à plusieurs autres via une table pivot polymorphique. Exemple : Un tag
Exemple : avec Spatie, la table model_has_role est un relation polymorphic. Un User peut avoir un ro
*/
// Un Tag peut être lié à plusieurs posts ou videos.
// Tag.php
public function posts()
{
    return $this->morphedByMany(Post::class, 'taggable');
}

public function videos()
{
    return $this->morphedByMany(Video::class, 'taggable');
}

// Post.php et Video.php
public function tags()
{
    return $this->morphToMany(Tag::class, 'taggable');
}
```