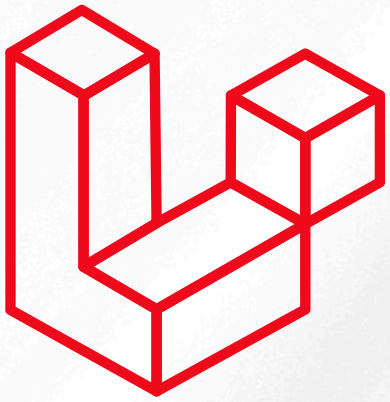

# Eloquent: Relationships

**Since relationships also serve as powerful query builders**

- **الـ** Relations

**العلاقات** (Relations) **في** Laravel **هي وسيلة لربط الجداول ببعضها في قاعدة البيانات بشكل بسيط وسهل. العلاقات بتخلي إدارة واسترجاع البيانات بين الجداول أسهل وأسرع، وده بيساعدك تبني تطبيقات قوية ومرتبة بطريقة احترافية.**

## ١. علاقة واحد لواحد (One-to-One)

**الفكرة هنـــا :**
**عندك جدول** users **وكل مستخدم له ملف شخصي خاص به في جدول** profiles .
**يعني كل سجل في جدول المستخدمين مرتبط بسجل واحد فقط في جدول الملفات الشخصية .**

```php
// In User.php
// Defines the relationship between a user and their profile as a one-to-one relationship
public function profile()
{
    return $this->hasOne(Profile::class);
}

// In Profile.php
// Defines the relationship between a profile and the user as a many-to-one relationship
public function user()
{
    return $this->belongsTo(User::class);
}
```

**الاستخدام العملي :**

```php
$user = User::find(1); // Finds the user with ID 1
echo $user->profile->bio; // Prints the bio from the user's profile
```

## ٢. علاقة واحد إلى متعدد (One-to-Many)

**الفكرة هنـــا :**
**كاتب واحد يقدر يكتب أكثر من مقال ، لكن المقال مرتبط بكاتب واحد فقط . هذه العلاقة تربط سجل واحد في جدول بعدة سجلات في جدول آخر.**

```php
// In Author.php
// Defines the relationship between an author and their posts as a one-to-many relationship
public function posts()
{
    return $this->hasMany(Post::class);
}

// In Post.php
// Defines the relationship between a post and its author as a many-to-one relationship
public function author()
{
    return $this->belongsTo(Author::class);
}
```

**الاستخدام العملي :**

```php
$author = Author::find(1);
foreach ($author->posts as $post) {
    echo $post->title;
}
```

## ٣. علاقة متعدد إلى متعدد (Many-to-Many)

**الفكرة هنـــــا :**

**الطلاب يقدروا يشتركوا في عدة دورات ، والدورة الواحدة يقدر يشترك فيها عدة طلاب . هنا نحتاج جدول وسيط (Pivot Table) مثل course_student علشان يحفظ الربط بينهم .**

```php
// In Student.php
// Defines the relationship between a student and their courses as a many-to-many relationship
public function courses()
{
    return $this->belongsToMany(Course::class);
}

// In Course.php
// Defines the relationship between a course and its students as a many-to-many relationship
public function students()
{
    return $this->belongsToMany(Student::class);
}
```

**الاستخدام العملي :**

```php
$student = Student::find(1);
foreach ($student->courses as $course) {
    echo $course->name;
}
```

## ٤. علاقة متعددة الأشكال (Polymorphic)

**الفكرة هنـــــا :**

**عندك ميزة الإعجابات (Likes) التي يمكن تطبيقها على منشورات (Posts) أو تعليقات (Comments). بدل ما تعمل جدول مختلف لكل نوع، تستخدم جدول واحد يحفظ الإعجابات مع تحديد النوع.**

```php
// In Like.php
// Defines a polymorphic relationship indicating that a Like can belong to any model (e.g., Post or Comment)
public function likeable()
{
    return $this->morphTo();
}

// In Post.php
// Defines a polymorphic one-to-many relationship indicating that a Post can have many likes
public function likes()
{
    return $this->morphMany(Like::class, 'likeable');
}

// In Comment.php
// Defines a polymorphic one-to-many relationship indicating that a Comment can have many likes
public function likes()
{
    return $this->morphMany(Like::class, 'likeable');
}
```

**الاستخدام العملي :**

```php
$post = Post::find(1);
foreach ($post->likes as $like) {
    echo $like->user_id;
}
```