Understanding Laravel's `withCount()` with Query Scopes

What is `withCount()` in Laravel?

`withCount()` is a Laravel method used to count related records and include the count as a column in the query result.

Basic Usage

```php
Post::withCount('reviews')->get();
```

This adds a `reviews_count` column to each `Post` instance, showing how many related `reviews` exist.

Passing Relationships as an Array in `withCount()`

It is possible to pass relationships as an array inside `withCount()`. This allows applying conditions on the counted relationship:

```php
Post::withCount([
    'reviews' => function (Builder $q) {
        $q->where('created_at', '>=', '2024-01-01'); // Count only reviews from 2024 onwards
    }
])->get();
```

- The **array key** `'reviews'` represents the relationship.
- The **closure** modifies the query, filtering `reviews` by date before counting them.

Using `withCount()` in a Query Scope

A query scope encapsulates query logic inside a model for reuse.

Example: `scopePopular` for Sorting by Review Count

```php
use Illuminate\Database\Eloquent\Builder;

public function scopePopular(Builder $query, $from = null, $to = null): Builder
{
    return $query->withCount([
        'reviews' => function (Builder $q) use ($from, $to) {
            if ($from && !$to) {
                $q->where('created_at', '>=', $from);
            } elseif (!$from && $to) {
                $q->where('created_at', '<=', $to);
            } elseif ($from && $to) {
                $q->whereBetween('created_at', [$from, $to]);
            }
        }
    ])->orderBy('reviews_count', 'desc');
}
```

How It Works

- **Counts `reviews`** associated with each model instance.
- **Filters reviews** by creation date (`$from`, `$to`).
- **Orders results** by `reviews_count` in descending order (most reviewed first).

Using the Scope in Queries

Fetch Most Popular Posts (All Time)
```php
$popularPosts = Post::popular()->get();
```

Fetch Most Popular Posts Within a Date Range
```php
$popularPosts = Post::popular('2024-01-01', '2024-02-01')->get();
```

Fetch Popular Posts Since a Specific Date
```php
$popularPosts = Post::popular('2024-01-01')->get();
```

Fetch Popular Posts Until a Specific Date
```php
$popularPosts = Post::popular(null, '2024-02-01')->get();
```

Alternative Approach: Using a Separate Relationship
If you need multiple filtered counts, you can define different relationships in your model:

```php
public function recentReviews()
{
    return $this->hasMany(Review::class)->where('created_at', '>=', now()->subMonth());
}
```

Then, use:

```php
Post::withCount('recentReviews')->get();
```

Conclusion

Passing relationships as an **array** inside `withCount()` is valid. Allows **modifying count queries** with conditions. Encapsulating logic in a **query scope** makes it reusable. Great for sorting models by related record counts dynamically!