基于 Laravel 包的多级设置解决方案

一、可选包分析

1. spatie/laravel-settings

优点:

- 支持强类型设置
- 支持缓存
- 可以按组分类
- 支持加密
- 文档完善
- 活跃维护

缺点:

- 不支持多级继承
- 不支持权限控制

2. anlutro/laravel-settings

优点:

- 简单易用
- 支持多种存储驱动
- 轻量级

缺点:

- 功能较简单
- 不支持类型检查
- 维护不够活跃

3. 推荐方案

使用 spatie/laravel-settings 作为基础,通过扩展实现多级继承和权限控制功能。

二、实现方案

1. 安装基础包

```
composer require spatie/laravel-settings
```

2. 创建设置类

```
<?php

namespace App\Settings;

use Spatie\LaravelSettings\Settings;

class CommissionSettings extends Settings
{
    public float $default_rate;
    public float $min_rate;
    public float $max_rate;

    // 定义分组
    public static function group(): string
    {
        return 'commission';
    }
}
```

3. 实现多级继承

```
<?php

namespace App\Services;

use Spatie\LaravelSettings\Settings;
use Illuminate\Support\Facades\Cache;

class HierarchicalSettingsService
{
    private const CACHE_PREFIX = 'settings:';
    private const CACHE_TTL = 3600;

    /**
     * 获取设置值(支持多级继承)
     */
    public function getValue(string $settingsClass, string $key, string $objectType, in
    {
        $cacheKey = $this->getCacheKey($settingsClass, $key, $objectType, $objectId);

        return Cache::remember($cacheKey, self::CACHE_TTL, function () use ($settingsCl
            // 1. 查找当前对象的设置
            $value = $this->getObjectValue($settingsClass, $key, $objectType, $objectId
            if ($value !== null) {
                return $value;
            }

            // 2. 查找上级对象的设置
            $parentObject = $this->getParentObject($objectType, $objectId);
            if ($parentObject) {
                return $this->getValue(
                    $settingsClass,
                    $key,
                    $parentObject['type'],
                    $parentObject['id']
                );
            }

            // 3. 返回默认值
            return app($settingsClass)->$key;
        });
    }
```

```
/**
 * 设置值
 */
public function setValue(string $settingsClass, string $key, string $objectType, in
{
    // 权限检查
    if (!$this->checkPermission($settingsClass, $key, $objectType)) {
        throw new \Exception('没有权限修改此设置');
    }

    // 值范围检查
    if (!$this->validateValue($settingsClass, $key, $value, $objectType)) {
        throw new \Exception('设置值超出允许范围');
    }

    // 保存设置
    $this->saveObjectValue($settingsClass, $key, $objectType, $objectId, $value);

    // 清除缓存
    $this->clearCache($settingsClass, $key, $objectType, $objectId);
}

/**
 * 获取对象的设置值
 */
private function getObjectValue(string $settingsClass, string $key, string $objectT
{
    return \DB::table('settings')
        ->where('group', app($settingsClass)->group())
        ->where('name', $key)
        ->where('object_type', $objectType)
        ->where('object_id', $objectId)
        ->value('payload');
}

/**
 * 保存对象的设置值
 */
private function saveObjectValue(string $settingsClass, string $key, string $object
{
    \DB::table('settings')->updateOrInsert(
        [
                'group' => app($settingsClass)->group(),
                'name' => $key,
                'object_type' => $objectType,
                'object_id' => $objectId,
            ],
            [
                'payload' => $value,
                'updated_at' => now(),
            ]
        );
    }

    /**
     * 获取上级对象
     */
    private function getParentObject(string $objectType, int $objectId): ?array
    {
        // 实现获取上级对象的逻辑
        // 例如: COACH -> SHOP -> AGENT -> PLATFORM
        switch ($objectType) {
            case 'COACH':
                $shopId = \DB::table('coaches')
                    ->where('id', $objectId)
                    ->value('shop_id');
                return $shopId ? ['type' => 'SHOP', 'id' => $shopId] : null;

            case 'SHOP':
                $agentId = \DB::table('shops')
                    ->where('id', $objectId)
                    ->value('agent_id');
                return $agentId ? ['type' => 'AGENT', 'id' => $agentId] : null;

            case 'AGENT':
                return ['type' => 'PLATFORM', 'id' => 0];

            default:
                return null;
        }
    }

    /**
     * 检查权限
     */
    private function checkPermission(string $settingsClass, string $key, string $object
    {
        // 实现权限检查逻辑
        return true;
    }

    /**
     * 验证值范围
     */
    private function validateValue(string $settingsClass, string $key, $value, string $
    {
        // 实现值范围验证逻辑
        return true;
    }

    /**
     * 获取缓存键
     */
    private function getCacheKey(string $settingsClass, string $key, string $objectType
    {
        return self::CACHE_PREFIX . "{$settingsClass}:{$key}:{$objectType}:{$objectId}"
    }

    /**
     * 清除缓存
     */
    private function clearCache(string $settingsClass, string $key, string $objectType,
    {
        Cache::forget($this->getCacheKey($settingsClass, $key, $objectType, $objectId))
    }
}
```

4. 数据库迁移

```php
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

return new class extends Migration
{
    public function up()
    {
        Schema::create('settings', function (Blueprint $table) {
            $table->id();
            $table->string('group');
            $table->string('name');
            $table->string('object_type');
            $table->unsignedBigInteger('object_id');
            $table->text('payload');
            $table->timestamps();

            $table->unique(['group', 'name', 'object_type', 'object_id']);
        });

        Schema::create('setting_permissions', function (Blueprint $table) {
            $table->id();
            $table->string('group');
            $table->string('name');
            $table->string('object_type');
            $table->boolean('can_edit')->default(false);
            $table->json('constraints')->nullable();
            $table->timestamps();

            $table->unique(['group', 'name', 'object_type']);
        });
    }
};
```

5. 使用示例

```php
<?php

namespace App\Http\Controllers;

use App\Settings\CommissionSettings;
use App\Services\HierarchicalSettingsService;

class SettingsController extends Controller
{
    private $settingsService;

    public function __construct(HierarchicalSettingsService $settingsService)
    {
        $this->settingsService = $settingsService;
    }

    /**
     * 获取分佣比例
     */
    public function getCommissionRate($coachId)
    {
        $rate = $this->settingsService->getValue(
            CommissionSettings::class,
            'default_rate',
            'COACH',
            $coachId
        );

        return response()->json(['rate' => $rate]);
    }

    /**
     * 设置分佣比例
     */
    public function setCommissionRate($coachId, Request $request)
    {
        $this->settingsService->setValue(
            CommissionSettings::class,
            'default_rate',
            'COACH',
            $coachId,
            $request->input('rate')
        );

        return response()->json(['message' => '设置成功']);
    }
}
```

三、扩展功能

1. 添加设置项验证器

```php
<?php

namespace App\Settings\Validators;

class CommissionSettingsValidator
{
    public function validateDefaultRate($value, $objectType)
    {
        $constraints = [
            'PLATFORM' => ['min' => 0, 'max' => 1],
            'AGENT' => ['min' => 0.1, 'max' => 0.9],
            'SHOP' => ['min' => 0.2, 'max' => 0.8],
            'COACH' => ['min' => 0.3, 'max' => 0.7],
        ];

        if (!isset($constraints[$objectType])) {
            return false;
        }

        return $value >= $constraints[$objectType]['min']
            && $value <= $constraints[$objectType]['max'];
    }
}
```

2. 添加设置变更日志

```
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Model;

class SettingLog extends Model
{
    protected $fillable = [
        'group',
        'name',
        'object_type',
        'object_id',
        'old_value',
        'new_value',
        'operator_id',
        'operator_type'
    ];
}
```

3. 添加设置导出功能

```php
<?php

namespace App\Exports;

use Maatwebsite\Excel\Concerns\FromCollection;

class SettingsExport implements FromCollection
{
    private $objectType;
    private $objectId;

    public function __construct($objectType, $objectId)
    {
        $this->objectType = $objectType;
        $this->objectId = $objectId;
    }

    public function collection()
    {
        return \DB::table('settings')
            ->where('object_type', $this->objectType)
            ->where('object_id', $this->objectId)
            ->get();
    }
}
```

四、优势与特点

1. 基于成熟的 Laravel 包开发,可靠性高
2. 支持多级设置继承
3. 支持缓存机制
4. 支持权限控制
5. 支持值范围验证
6. 支持设置变更日志
7. 支持导入导出
8. 代码结构清晰,易于扩展

五、注意事项

1. ��� 要合理设计缓存策略,避免缓存击穿
2. 权限控制要考虑安全性
3. 值范围验证要考虑业务规则
4. 要注意性能优化,避免多次查询数据库
5. 建议添加单元测试保证代码质量

六、后续优化方向

1. 添加设置模板功能
2. 支持设置项版本控制
3. 添加设置项依赖关系
4. 优化缓存策略
5. 添加更多导出格式支持
6. 完善单元测试