多级设置解决方案

关于设置表的问题
1、平台中存在不同级别“业务主体对象”的设置，在设置项相同的情况下，如技师对某个开通项目的某个设置、技师对自
2、如果设置的权限开放给不同级别的“业务主体对象”，则需要平台来限制每一个设置项的可调整范围，这个又该怎么设
3、这个设置功能需要设计一个模块，按 io 模型分析，输入参数可能是业务主体对象 id，返回正确的设置集（如下单

一、需求分析

1. 业务场景

- 平台中存在不同级别"业务主体对象"的设置
- 在设置项相同的情况下,按优先级从大到小查找，“业务主体对象”也可能不存在:
  - 技师设置 -> 店铺设置 -> 代理设置 -> 平台设置 -> 默认设置
- 需要平台限制每个设置项的可调整范围
- 因“业务主体对象”存在配置继承关系，但不是强关联，所以查询查询”上级业务主体 id”的方法需要自己实现，然后根据上级业务主体 id 查询设置值

2. 功能要求

- 支持多级设置继承
- 灵活的权限控制
- 设置值范围限制
- 设置项分组管理
- 完整的设置历史记录

二、数据库设计

1. 设置组表(setting_groups)

```
CREATE TABLE `setting_groups` (
  `id` bigint unsigned NOT NULL AUTO_INCREMENT COMMENT '主键ID',
  `code` varchar(50) NOT NULL COMMENT '设置组编码',
  `name` varchar(50) NOT NULL COMMENT '设置组名称',
  `description` varchar(255) DEFAULT NULL COMMENT '设置组描述',
  `sort` int NOT NULL DEFAULT 0 COMMENT '排序',
  `created_at` timestamp NULL DEFAULT NULL,
  `updated_at` timestamp NULL DEFAULT NULL,
  PRIMARY KEY (`id`),
  UNIQUE KEY `uk_code` (`code`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci COMMENT='设置表';
```

2. 设置项表(setting_items)

```
CREATE TABLE `setting_items` (
  `id` bigint unsigned NOT NULL AUTO_INCREMENT COMMENT '主键ID',
  `group_id` bigint unsigned NOT NULL COMMENT '设置组ID',
  `code` varchar(50) NOT NULL COMMENT '设置项编码',
  `name` varchar(50) NOT NULL COMMENT '设置项名称',
  `description` varchar(255) DEFAULT NULL COMMENT '设置项描述',
  `value_type` varchar(20) NOT NULL COMMENT '值类型:string,number,boolean,json',
  `default_value` text DEFAULT NULL COMMENT '默认值',
  `min_value` decimal(10,2) DEFAULT NULL COMMENT '最小值(数值类型)',
  `max_value` decimal(10,2) DEFAULT NULL COMMENT '最大值(数值类型)',
  `options` json DEFAULT NULL COMMENT '可选值(json格式)',
  `sort` int NOT NULL DEFAULT 0 COMMENT '排序',
  `created_at` timestamp NULL DEFAULT NULL,
  `updated_at` timestamp NULL DEFAULT NULL,
  PRIMARY KEY (`id`),
  UNIQUE KEY `uk_code` (`code`),
  KEY `idx_group_id` (`group_id`),
  CONSTRAINT `fk_setting_items_group_id` FOREIGN KEY (`group_id`) REFERENCES `setting_g
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci COMMENT='设置项表';
```

3. 设置权限表(setting_permissions)

```
CREATE TABLE `setting_permissions` (
  `id` bigint unsigned NOT NULL AUTO_INCREMENT COMMENT '主键ID',
  `item_id` bigint unsigned NOT NULL COMMENT '设置项ID',
  `object_type` varchar(20) NOT NULL COMMENT '业务对象类型:PLATFORM,AGENT,SHOP,COACH',
  `can_edit` tinyint(1) NOT NULL DEFAULT 0 COMMENT '是否可编辑',
  `min_value` decimal(10,2) DEFAULT NULL COMMENT '最小值限制',
  `max_value` decimal(10,2) DEFAULT NULL COMMENT '最大值限制',
  `options` json DEFAULT NULL COMMENT '可选值限制',
  `created_at` timestamp NULL DEFAULT NULL,
  `updated_at` timestamp NULL DEFAULT NULL,
  PRIMARY KEY (`id`),
  UNIQUE KEY `uk_item_object` (`item_id`,`object_type`),
  KEY `idx_item_id` (`item_id`),
  CONSTRAINT `fk_setting_permissions_item_id` FOREIGN KEY (`item_id`) REFERENCES `setti
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci COMMENT='设置权限表';
```

4. 设置值表(setting_values)

```
CREATE TABLE `setting_values` (
  `id` bigint unsigned NOT NULL AUTO_INCREMENT COMMENT '主键ID',
  `item_id` bigint unsigned NOT NULL COMMENT '设置项ID',
  `object_type` varchar(20) NOT NULL COMMENT '业务对象类型:PLATFORM,AGENT,SHOP,COACH',
  `object_id` bigint unsigned NOT NULL COMMENT '业务对象ID',
  `value` text NOT NULL COMMENT '设置值',
  `created_at` timestamp NULL DEFAULT NULL,
  `updated_at` timestamp NULL DEFAULT NULL,
  PRIMARY KEY (`id`),
  UNIQUE KEY `uk_item_object` (`item_id`,`object_type`,`object_id`),
  KEY `idx_object` (`object_type`,`object_id`),
  CONSTRAINT `fk_setting_values_item_id` FOREIGN KEY (`item_id`) REFERENCES `setting_it
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci COMMENT='设置值表';
```

三、代码实现

1. Model 定义

```php
<?php
namespace App\Models;

use Illuminate\Database\Eloquent\Model;

class SettingGroup extends Model
{
    protected $fillable = ['code', 'name', 'description', 'sort'];

    public function items()
    {
        return $this->hasMany(SettingItem::class, 'group_id');
    }
}

class SettingItem extends Model
{
    protected $fillable = [
        'group_id', 'code', 'name', 'description',
        'value_type', 'default_value', 'min_value',
        'max_value', 'options', 'sort'
    ];

    protected $casts = [
        'options' => 'json'
    ];

    public function group()
    {
        return $this->belongsTo(SettingGroup::class);
    }

    public function permissions()
    {
        return $this->hasMany(SettingPermission::class, 'item_id');
    }

    public function values()
```

```
    {
        return $this->hasMany(SettingValue::class, 'item_id');
    }
}
```

2. Service 实现

```php
<?php
namespace App\Services;

class SettingService
{
    /**
     * 获取设置值
     */
    public function getValue($code, $objectType, $objectId)
    {
        // 获取设置项
        $item = SettingItem::where('code', $code)->first();
        if (!$item) {
            throw new Exception('设置项不存在');
        }

        // 查找设置值
        $value = SettingValue::where([
            'item_id' => $itemId,
            'object_type' => $objectType,
            'object_id' => $objectId
        ])->value('value');

        // 返回设置值或默认值
        return $value ?? $item->default_value;
    }

    /**
     * 更新设置值
     */
    public function setValue($code, $objectType, $objectId, $value)
    {
        // 验证和权限检查...

        SettingValue::updateOrCreate(
            [
                'item_id' => $item->id,
                'object_type' => $objectType,
                'object_id' => $objectId
            ],
            ['value' => $value]
```

```
        );
    }
}
```

四、使用示例

1. 定义设置项

```php
// 设置组
$orderGroup = SettingGroup::create([
    'code' => 'order',
    'name' => '订单设置'
]);

// 设置项
SettingItem::create([
    'group_id' => $orderGroup->id,
    'code' => 'min_amount',
    'name' => '最低订单金额',
    'value_type' => 'number',
    'default_value' => 100,
    'min_value' => 0,
    'max_value' => 1000
]);
```

2. 设置权限

```php
// 平台可以设置0-1000
SettingPermission::create([
    'item_id' => $item->id,
    'object_type' => 'PLATFORM',
    'can_edit' => true,
    'min_value' => 0,
    'max_value' => 1000
]);

// 代理可以设置50-500
SettingPermission::create([
    'item_id' => $item->id,
    'object_type' => 'AGENT',
    'can_edit' => true,
    'min_value' => 50,
    'max_value' => 500
]);
```

3. 使用设置值

```php
// 获取设置值
$minAmount = $settingService->getValue(
    'min_amount',
    'COACH',
    $coachId
);

// 新设置值
$settingService->setValue(
    'min_amount',
    'AGENT',
    $agentId,
    200
);
```

4. 设置平台默认分账比例

```php
// 设置平台默认分账比例
$settingService->setValue(
    'commission_rate',
    'PLATFORM',
    0,
    0.7  // 平台默认70%分账比例
);

// 设置某个代理商的分账比例
$settingService->setValue(
    'commission_rate',
    'AGENT',
    $agentId,
    0.75  // 该代理商区域75%分账比例
);

// 获取技师当前位置的分账比例
$commissionRate = $settingService->getValue(
    'commission_rate',
    'COACH',
    $coachId,
);

// 实时计算技师订单分账
class OrderService
{
    public function calculateCommission($orderId)
    {
        $order = Order::find($orderId);
        $coach = $order->coach;

        // 获取订单创建时技师所在位置的分账比例
        $commissionRate = app(SettingService::class)->getValue(
            'commission_rate',
            'COACH',
            $coach->id
        );

        // 计算分账金额
        $commissionAmount = $order->amount * $commissionRate;
```

```php
        return $commissionAmount;
    }
}

// 监听技师位置变化
class CoachLocationListener
{
    public function handle(CoachLocationChanged $event)
    {
        $coach = $event->coach;

        // 获取新位的设置
        $newSettings = app(SettingService::class)->getValue(
            'commission_rate',
            'COACH',
            $coach->id,
            $event->latitude,
            $event->longitude
        );

        // 可以在这里处理设置变化后的业务逻辑
        // 比如通知技师费率变化等
    }
}
```

五、扩展功能实现

1. 设置项缓存机制

```php
<?php
namespace App\Services;

use Illuminate\Support\Facades\Cache;

class SettingCacheService
{
    const CACHE_PREFIX = 'setting:';
    const CACHE_TTL = 3600; // 1小时

    /**
     * 获取缓存的设置值
     */
    public function getCachedValue($code, $objectType, $objectId)
    {
        $cacheKey = $this->getCacheKey($code, $objectType, $objectId);

        return Cache::remember($cacheKey, self::CACHE_TTL, function () use ($code, $obj
            return app(SettingService::class)->getValue($code, $objectType, $objectId);
        });
    }

    /**
     * 清除设置值缓存
     */
    public function clearCache($code, $objectType, $objectId)
    {
        $cacheKey = $this->getCacheKey($code, $objectType, $objectId);
        Cache::forget($cacheKey);
    }

    /**
     * 生成缓存key
     */
    private function getCacheKey($code, $objectType, $objectId)
    {
        return self::CACHE_PREFIX . "{$code}:{$objectType}:{$objectId}";
```

```
    }
}
```

2. 设置值变更日志

```php
<?php
// 设置值变更日志表
CREATE TABLE `setting_value_logs` (
    `id` bigint unsigned NOT NULL AUTO_INCREMENT COMMENT '主键ID',
    `item_id` bigint unsigned NOT NULL COMMENT '设置项ID',
    `object_type` varchar(20) NOT NULL COMMENT '业务对象类型',
    `object_id` bigint unsigned NOT NULL COMMENT '业务对象ID',
    `old_value` text DEFAULT NULL COMMENT '原值',
    `new_value` text NOT NULL COMMENT '新值',
    `operator_id` bigint unsigned NOT NULL COMMENT '操作人ID',
    `operator_type` varchar(20) NOT NULL COMMENT '操作人类型',
    `created_at` timestamp NULL DEFAULT NULL,
    PRIMARY KEY (`id`),
    KEY `idx_item_object` (`item_id`,`object_type`,`object_id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci COMMENT='设置值变更日志

// 设置值变更日志服务
class SettingLogService
{
    /**
     * 记录设置值变更
     */
    public function logValueChange($itemId, $objectType, $objectId, $oldValue, $newValu
    {
        SettingValueLog::create([
            'item_id' => $itemId,
            'object_type' => $objectType,
            'object_id' => $objectId,
            'old_value' => $oldValue,
            'new_value' => $newValue,
            'operator_id' => Auth::id(),
            'operator_type' => Auth::user()->type
        ]);
    }

    /**
     * 获取设置值变更历史
     */
    public function getValueHistory($itemId, $objectType, $objectId)
    {
        return SettingValueLog::where([
```

```
            'item_id' => $itemId,
            'object_type' => $objectType,
            'object_id' => $objectId
        ])->orderBy('created_at', 'desc')->get();
    }
}
```

3. 设置模板功能

```php
<?php
// 设置模板表
CREATE TABLE `setting_templates` (
    `id` bigint unsigned NOT NULL AUTO_INCREMENT COMMENT '主键ID',
    `name` varchar(50) NOT NULL COMMENT '模板名称',
    `description` varchar(255) DEFAULT NULL COMMENT '模板描述',
    `object_type` varchar(20) NOT NULL COMMENT '适用对象类型',
    `settings` json NOT NULL COMMENT '设置项值(json格式)',
    `created_at` timestamp NULL DEFAULT NULL,
    `updated_at` timestamp NULL DEFAULT NULL,
    PRIMARY KEY (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci COMMENT='设置模板表';

// 设置模板服务
class SettingTemplateService
{
    /**
     * 创建设置模板
     */
    public function createTemplate($name, $objectType, array $settings)
    {
        return SettingTemplate::create([
            'name' => $name,
            'object_type' => $objectType,
            'settings' => json_encode($settings)
        ]);
    }

    /**
     * 应用设置模板
     */
    public function applyTemplate($templateId, $objectType, $objectId)
    {
        $template = SettingTemplate::findOrFail($templateId);
        $settings = json_decode($template->settings, true);

        foreach ($settings as $code => $value) {
            app(SettingService::class)->setValue($code, $objectType, $objectId, $value);
        }
```

```
    }
}
```

4. 设置导入导出功能

```php
<?php
namespace App\Services;

use Illuminate\Support\Facades\Storage;
use League\Csv\Reader;
use League\Csv\Writer;

class SettingImportExportService
{
    /**
     * 导设置值
     */
    public function exportSettings($objectType, $objectId)
    {
        $settings = SettingValue::where([
            'object_type' => $objectType,
            'object_id' => $objectId
        ])->with('item')->get();

        $csv = Writer::createFromString('');
        $csv->insertOne(['设置项编码', '设置项名称', '置值']);

        foreach ($settings as $setting) {
            $csv->insertOne([
                $setting->item->code,
                $setting->item->name,
                $setting->value
            ]);
        }

        return $csv->toString();
    }

    /**
     * 导入设置值
     */
    public function importSettings($objectType, $objectId, $file)
    {
        $csv = Reader::createFromPath($file->getPathname());
        $csv->setHeaderOffset(0);
```

```
        foreach ($csv as $record) {
            app(SettingService::class)->setValue(
                $record['设置项编码'],
                $objectType,
                $objectId,
                $record['设置值']
            );
        }
    }
}
```

5. 设置值计算功能

```php
<?php
namespace App\Services;

class SettingCalculatorService
{
    /**
     * 计算设置值
     */
    public function calculateValue($code, $objectType, $objectId, array $params = [])
    {
        $item = SettingItem::where('code', $code)->first();
        if (!$item) {
            throw new Exception('设置项不存在');
        }

        // 获取基础值
        $baseValue = app(SettingService::class)->getValue($code, $objectType, $objectId

        // 根据不同计算类型处理
        switch ($item->calculate_type) {
            case 'percentage':
                return $this->calculatePercentage($baseValue, $params);
            case 'formula':
                return $this->calculateFormula($baseValue, $params);
            case 'range':
                return $this->calculateRange($baseValue, $params);
            default:
                return $baseValue;
        }
    }

    /**
     * 计算百分比
     */
    private function calculatePercentage($baseValue, array $params)
    {
        $percentage = $params['percentage'] ?? 100;
        return $baseValue * ($percentage / 100);
    }

    /**
```

```
     * 计算公式
     */
    private function calculateFormula($baseValue, array $params)
    {
        $formula = $params['formula'] ?? '';
        // 替换公式中的变量
        $formula = strtr($formula, [
            '{value}' => $baseValue,
            '{param1}' => $params['param1'] ?? 0,
            '{param2}' => $params['param2'] ?? 0
        ]);

        // 使用eval计算公式(注意安全性)
        return eval("return {$formula};");
    }

    /**
     * 算范围值
     */
    private function calculateRange($baseValue, array $params)
    {
        $min = $params['min'] ?? PHP_FLOAT_MIN;
        $max = $params['max'] ?? PHP_FLOAT_MAX;

        return max($min, min($max, $baseValue));
    }
}
```

使用示例:

```php
// 1. 使用缓存服务
$settingValue = app(SettingCacheService::class)->getCachedValue('min_amount', 'SHOP', $

// 2. 记录变更日志
app(SettingLogService::class)->logValueChange(
    $item->id,
    'SHOP',
    $shopId,
    $oldValue,
    $newValue
);

// 3. 使用设置模板
$template = app(SettingTemplateService::class)->createTemplate(
    '标准店铺设置',
    'SHOP',
    [
        'min_amount' => 100,
        'max_amount' => 1000
    ]
);
app(SettingTemplateService::class)->applyTemplate($template->id, 'SHOP', $shopId);

// 4. 导入导出设置
$csv = app(SettingImportExportService::class)->exportSettings('SHOP', $shopId);
app(SettingImportExportService::class)->importSettings('SHOP', $shopId, $request->file(

// 5. 计算设置值
$actualValue = app(SettingCalculatorService::class)->calculateValue(
    'commission_rate',
    'COACH',
    $coachId,
    ['percentage' => 80] // 80%
);
```