链滴

设计模式 - 观察者模式的批量应用

作者：henryspace

原文链接：https://ld246.com/article/1649416027428

来源网站：链滴

许可协议：署名-相同方式共享 4.0 国际 (CC BY-SA 4.0)

观察者模式

当对象间存在一对多关系时，则使用观察者模式（Observer Pattern）。比如，当一个对象被修改时则会自动通知依赖它的对象。观察者模式属于行为型模式。

1，设置处理事件的接口

```
// Obserable.php
namespace App\Services\Observer;

/**
 * 处理观察者事件接口
 * Interface Obserable
 * @package App\Services\Observer
 */
interface Obserable {

    public function handle();

}
```

2, 设置事件过滤器，便于处理不同的事件

```
// EventFilter.php
namespace App\Services\Observer;


/**
 * 定义观察者触发的事件，并做转发
 *
 * Class EventFilter
```

```
 * @package App\Services\Observer
 */
class EventFilter implements Obserable {

    // 订单下单事件
    const EVENT_ORDER_SUBMIT = 'orderSubmit';

    // 订单退款事件
    const EVENT_ORDER_REFUND = 'orderRefund';

    // ... ...

    /**
     * 注册的观察者
     * @var
     */
    public $observer;

    /**
     * 注册的观察者绑定事件
     * @var
     */
    public $event;

    /**
     * 注册的观察者绑定事件参数
     * @var
     */
    public $post;


    /**
     * 触发处理
     * @return mixed
     */
    public function handle()
    {
        return call_user_func_array([$this->observer, $this->event], [$this]);
    }

}
```

3, 设置观察者单一事件的对象集注册器

```
// Observed.php
namespace App\Services\Observer;


class Observed
{

    /**
     * 观察的注册主体对象集
     * @var array
```

```
     */
    private $_observers = [];

    /**
     * 观察的事件
     * @var string
     */
    public $event;

    /**
     * 注册的观察者绑定事件参数
     * @var
     */
    public $post;


    public function register($observer)
    {
        $this->_observers[] = $observer;
    }


    /**
     * 触发执行
     */
    public function trigger()
    {
        $posts = $this->post;
        $filter = new EventFilter();
        foreach ($this->_observers as $observer) {

            $filter->observer = $observer;
            $filter->event = $this->event;
            $filter->post = $posts;

            $filter->handle();
            $posts = $filter->post;
        }
        $this->post = $posts;

        return $this;
    }

}
```

4，具体场景调用（以下单事件为例）

```
$post = $request->all();
DB::beginTransaction();
try {
    $observed = new Observed();
    $observed->event = EventFilter::EVENT_ORDER_SUBMIT;
    $observed->post = $post;
    $observed->register(new ObserveUserAddress());
```

```
    $observed->register(new ObserveShoppingCart());
    $observed->register(new ObserveLogistics());
    $observed->register(new ObserveOrder());
    $observed->register(new ObserveOrderItem());
    $observed->trigger();
} catch (\Exception $e) {
    DB::rollBack();
    return error($e->getMessage());
}
DB::commit();
```

意图： 定义对象间的一种一对多的依赖关系，当一个对象的状态发生改变时，所有依赖于它的对象都到通知并被自动更新。

主要解决： 一个对象状态改变给其他对象通知的问题，而且要考虑到易用和低耦合，保证高度的协作。

何时使用： 一个对象（目标对象）的状态发生改变，所有的依赖对象（观察者对象）都将得到通知，行广播通知。