链滴

# 面向对象的六大原则（单一职责原则）

作者：qingshui666
原文链接：https://ld246.com/article/1583938570459
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)

当我们要审视判断事物的好坏时，无论如何我们都需要有一个标准。而作为一个程序员我们也需要有个标准去判断代码结构设计的优劣。而在我们设计程序时这个标准正是面向对象的六大原则。

- 单一职责原则（S）
- 开闭原则（O）
- 里氏替换原则（L）
- 依赖倒置原则（D）
- 接口隔离原则（I）
- 合成复用原则
- 迪特米法则

# 单一职责原则

单一职责原则理解起来非常简单，一个人应该干好自己的本职工作就是遵循了单一职责原则，一个类做属于这个类的事情也是遵循了单一职责原则。

## 违反单一职责原则会存在什么问题

1. 代码无法复用
2. 调度混乱（不知道这个类到底是用来做什么的）
3. 难以拓展维护

我们看一个违反单一原则的类，看看这样的设计是否也存在你的项目中

```
<php
class OrderService
{
    //获取数据库连接
    public function getConnention()
    {

    }

    //获取订单
    public function getOrder()
    {

    }

    //创建JSON
    public function createJson()
    {

    }

    //返回订单JSON
    public function responeJson()
    {
```

```
    }
}
?>
```

我们可以看到 OrderService这个类它完成了几种职责

- 获取数据库连接
- 获取订单号
- 构建订单JSON
- 返回JSON

当一个类需要 获取数据库连接时或者我需要构造一个JSON时，我去创建一个 OrderService显然是不理的

这时候我们需要怎么去改进这样的设计呢?

```
Class DB
{
    //获取数据库连接
    public function getConnention()
    {

    }
}

Class OrderService
{
    private $db

    public function __construct(DB $db)
    {
        $this->db = $db;
    }

    //获取订单
    public function getOrder()
    {

    }
}

Class Json
{
    //创建订单JSON
    public function createOrderJson()
    {

    }

    //返回订单JSON
    public function responeJson()
    {
```

```
    }
}
```

重构完成以后

DB类负责和数据库进行交互

OrderService类负责订单相关的逻辑

Json类负责Json的构建和响应

每个类都各司其职业，当我需要某一个相关的方法时我可以单独把一个类拿出来，当我需要修改某些能时，能影响到这个类只有这个类的相关逻辑，而不再需要修改多个类。

最后我们遵循单一职责去写代码时记得

# 一个类只干一种事