

依赖注入

Edit by 柴佳卫

场景

- 有一个用户类,在登录的时候需要记录日志
- 记录日志分两种方式
 - 1.文件
 - 2.数据库

用户类

```
//用户
class User {
    protected $log;

    public function __construct() {
        $this->log = new DatabaseLog();
    }

    public function login() {
        echo 'login success ...' . PHP_EOL;
        $this->log->write();
    }
}
```

日志类

- 日志接口

```
// 日志接口  
interface Log {  
    public function write();  
}
```

- 文件日志

```
// 文件记录  
class FileLog implements Log {  
    public function write() {  
        echo 'file log write ...' . PHP_EOL;  
    }  
}
```

日志类

- 数据库日志

```
// 数据库记录日志
class DatabaseLog implements Log {
    public function write() {
        echo 'database log write ...' . PHP_EOL;
    }
}
```

V1

- 一般我们是需要哪种日志,就是使用哪个日志类

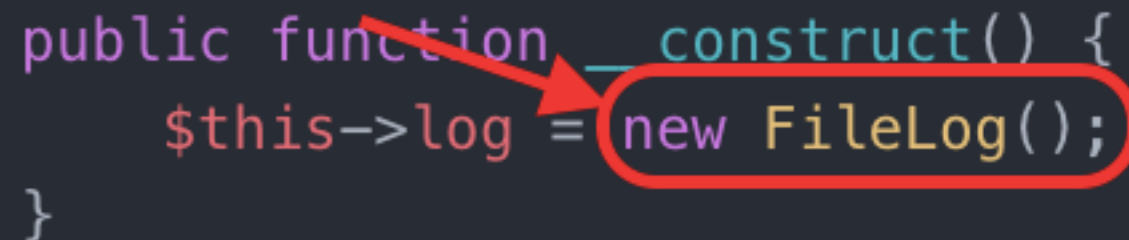
```
public function __construct() {  
    $this->log = new DatabaseLog();  
}
```

- 用户类使用场景

```
$user = new User();  
$user->login();
```

V1

- 问题: 如果我要使用文件进行日志记录呢?



```
public function __construct() {  
    $this->log = new FileLog();  
}
```

- 这样就必须要改动用户类了,不是很灵活

V2

- 于是改成了手工注入日志接口

```
public function __construct(Log $log) {  
    $this->log = $log;  
}
```

- 调用场景

```
$user = new User(new DatabaseLog());  
$user->login();
```


V2

- 一般情况下,做到这样已经很不错了,但是如果用户类又需要注入其他的接口呢?
 - 1.修改用户类构造函数,加入接口依赖
 - 2.场景类里再手工注入

V3

- 自动依赖注入
- 需要用到的知识点:

1. PHP反射机制 reflectionClass

2. `getConstructor()` 获取类的构造函数

3. `getParameters()` 获取方法的所有参数

V3

- 需要用到的知识点:

4. `getClass()` 获取所属类的信息

5. `newInstance()` 实例化反射对象

6. `newInstanceArgs(array())` 带参数实例化反射对象

make方法

- 传入需要实例化的类名称,返回实例化对象,并且自动解决构造函数的依赖

make方法

```
function make($concrete) {  
    $reflector      = new ReflectionClass($concrete);  
    $constructor    = $reflector->getConstructor();  
    if(is_null($constructor)) {  
        // 没有构造函数, 直接返回实例化对象  
        return $reflector->newInstance();  
    } else {  
        // 获取构造函数的参数  
        $parameters = $constructor->getParameters();  
        // 获取依赖的对象  
        $dependencies = getDependencies($parameters);  
        return $reflector->newInstanceArgs($dependencies);  
    }  
}
```

递归获取依赖

```
function getDependencies($parameters) {  
    $dependencies = [];  
    foreach($parameters as $parameter) {  
        $dependencies[] = make($parameter->getClass()->name);  
    }  
  
    return $dependencies;  
}
```

使用场景

- 更改用户类构造函数

```
public function __construct(FileLog $flog, DatabaseLog $dlog) {  
    $this->flog = $flog;  
    $this->dlog = $dlog;  
}
```

- 实例化用户类

```
$user = make(User::class);  
$user->login();
```

谢谢观看 ^ _ ^ @

满怀希望地旅行比到达终点更美好