构造方法的依赖注入：

需要用到的函数：

- ReflectionClass类。
  - 构造方法：
    - public ReflectionClass::__construct(mixed $argument)
      - $argument: 既可以是包含**类名的字符串（string）也可以是对象（object）**。
      - 换句话说，在实例化ReflectionClass类的时候需要传一个**类名参数**或者**对象**参数。
  - 功能：ReflectionClass 类报告了一个类的有关信息。
- ReflectionClass::getConstructor
  - 参数：此函数没有参数。
  - 功能：获取已**反射的类**的构造函数。
  - 返回值：
    - 若该反射的类有构造函数：一个 ReflectionMethod 对象。
      - 例子：
        - 反射test类：
          - class {
            public function __construct(b $b, demo $demo)
            {...}
            }
        - object(ReflectionMethod)#3 (2) {
          ["name"]=>
          string(11) "__construct"
          ["class"]=>
          string(4) "test"
          }
    - 若该反射的类没有构造函数：返回 null。
  - 语法：public ReflectionClass::getConstructor(): ReflectionMethod
- ReflectionClass::newInstanceArgs
  - 参数：以 array 形式传递到类的构造函数。
  - 功能：从给出的参数创建一个新的类实例。
  - 返回值：类的新实例。
  - 语法：public ReflectionClass::newInstanceArgs(array $args = ?): object
- getParameters
  - 参数：无参。
  - 功能：获取方法的参数。
  - 返回值：以数组的形式返回方法的参数。

- ReflectionParameter::getClass
  - 参数：此函数没有参数。
  - 功能：获得类型提示类。
  - 返回值：一个 ReflectionClass 对象。
    - 格式：object(ReflectionClass)#6 (1) {
      ["name"]=>
      string(1) "b"
      }
  - 语法：public ReflectionParameter::getClass(): ReflectionClass
- ReflectionClass::getName
  - 参数：无参。
  - 功能：获取类名。
  - 返回值：类名。
  - 语法：public ReflectionClass::getName(): string

make方法：

- 函数描述：
  - 根据类名实例化对象。
- 1、反射目标类：
- 2、问题1：有没有构造方法？
  - 没有：直接创建一个新的类实例。
  - 有：获取反射函数的参数
    - 将参数转换为类。

toObj方法：

- 函数描述：
  - 将数组中的参数挨个实例化类。
- 1、问题1：参数数组是否为空。
  - 如果为空返回值为[]（空数组）
- 2、问题2：参数数组的第一个元素是否是类型提示类。
  - 如果不是：
    - 继续递归剩余的数组。
  - 如果是：
    - 根据类名实例化第一个元素，(即make该函数)，继续递归剩余的数组。

代码：

```
class obj
{
    public function make($className)
    {
        // 1、反射。
        $reflect = new ReflectionClass($className);
        // 有没有构造方法
        $constructor = $reflect->getConstructor();
        if (!$constructor) {
            // 从给出的参数创建一个新的类实例。
            return $reflect->newInstanceArgs();
        }
        // 获取反射函数的参数：
        $params = $constructor->getParameters();
        $args = $this->toObj($params);
        return $reflect->newInstanceArgs($args);
    }
    public function toObj($params)
    {
        if (empty($params)) {
            return [];
        }
        // 获取数组的第一个元素。
        $car = array_shift($params);
        // 获取第一个元素的类型提示类。
        $class = $car->getClass();
        if (!$class) {
            return $this->toObj($params);
        } else {
            // 第一个元素是类，实例化该类。
            $obj = $this->make($class->getName());
            return array_merge([$obj], $this->toObj($params));
        }
    }
}
```

note路径：

- D:\phpStudy\PHPTutorial\WWW\index\aaa\依赖注入\note.md