PHP反射在设计模式中的应用分享试题

1、填空题（每空2分）

1、ReflectionClass的getMethod方法是否可以访问类的一个private方法，

如果不可以访问说明原因 \_\_\_\_是\_\_\_\_\_\_\_\_\_\_，如果可以访问，是否可以对获取的方法执行invoke操作\_\_\_\_\_\_\_\_\_否\_\_\_\_\_\_。

1. 反射是指PHP在\_\_\_\_运行\_\_\_\_状态中，动态获取\_\_\_\_类\_\_\_\_、\_\_方法\_\_\_\_、\_\_\_属性\_\_\_\_、\_\_\_参数\_\_\_\_、\_\_\_\_注释\_\_\_等信息和动态调用对象的方法的功能。
2. 依赖注入通过以\_\_\_\_构造函数\_\_\_参数，\_\_\_方法\_\_\_\_或\_\_\_\_属性\_\_\_等方式将具体组件传递给依赖方。
3. 策略模式中体现了面向对象三大特性的\_\_\_\_\_多态性\_\_\_\_。
4. 注解是允许加在源代码中的一种特殊格式的语法元数据，PHP并没有专门的语言特性来支持注解，可以通过反射 API 中\_\_\_\_getDocComment\_\_\_\_方法来实现。
5. 现有调用第三方发送消息的类，要实现在每次请求之前记录请求参数的日志，在不改变发送消息类方法的情况下，可以采用\_\_\_\_\_代理模式\_\_\_\_\_\_\_\_实现。
6. 在现实生活中，我们想去某个地方的话。一般会在地图APP中输入起点和终点。然后选择出行方式，而每种方式价格可能不一样，可以用设计模式中的\_\_\_\_策略模式\_\_\_来实现。
7. PHP 可以用魔术方法\_\_\_call\_\_\_方法的方式来实现远程调用，很容易实现动态代理模式。
8. 在\_\_\_\_文档生成器\_\_\_，\_\_\_\_依赖注入\_\_\_等地方广泛应用反射API，可以通过反射深入到类的内部，但反射也会破坏对象的\_\_\_\_封装性\_\_\_，使不该暴露的方法被强制暴露了。

2、简答题（第一题10分，第二题10分，第三题10分，第四题20分）

1. PHP如何验证类的属性是私有的，判断User类的 $name属性 (写出一种方式即可)

(1)、$prop = get\_class\_vars(User::class);

array\_key\_exists('name', $prop);

(2)、$ref = new ReflectionClass(User::class);

$prop = $ref->getProperty('name');

$prop->isPrivate();

1. 简要说明策略模式的使用场景？

(1)、针对同一类型问题的多种处理方式，仅仅是具体行为有差别时

(1)、需要安全地封装多种同一类型的操作时

(1)、出现同一抽象类有多个子类，而又需要使用 if-else 或者 switch-case 来选择具体子类时

3、简要描述装饰器模式优缺点。

优点:

（1）和继承的共同特点就是扩展对象的功能，而装饰器模式比继承更加灵活，可以在不改变原类文件和使用继承的情况下，动态的扩展一个 对象的功能

（2）通过使用不同的具体装饰器类，及其不同的排列组合，可以产生出大量不同的组合

缺点:  
1)比继承更加复杂.，（2）会出现一些小类，过度使用会使程序变得复杂

4、请求地址 /index.php?controller=user&action=list，访问到UserController下到list方法，用代码简单写出请求实现过程。

（1）$controller= $\_GET['controller'];$action= $\_GET['action'];

$class = new ReflectionClass(ucfirst($controller).'Controller');

$controller = $class->newInstance();

if ($class->hasMethod($method)) {

$method = $class->getMethod($method);

$method->invoke($controller);

}

（2）$controller=ucfirst($controller).'Controller'

if (class\_exists($controller)) {

$controller= new $controller;

$controller->$action();

}