构造方法和继承

四种情况：

- 父无子无。
  - 父类没有构造方法，子类没有构造方法。
  - 实例化子类时，无需考虑构造方法问题。
- 父无子有。
  - 父类没有构造方法，子类有构造方法。
  - 实例化子类时，只要考虑子类的构造方法即可。
- 父有子无。
  - 父类有构造方法，子类没有构造方法。
  - 实例化子类时，只要考虑父类的构造方法即可。
- 父有子有。
  - 父类有构造方法，子类有构造方法。
  - 实例化子类时，父类、子类的构造方法都要考虑。

单独分析构造方法的原因

- 我不能确定**在实例化子类的时候，父类的构造方法会不会调用**。
- StudentController类继承自Controller类。
- 一开始的代码是这么写的：

```
class Controller
{
    public $smarty;
    public $app;
    public $request;
    public function __construct(App $app, Request $request, Smarty $smarty)
    {
        $this->app = $app;
        $this->request = $request;
        $this->smarty = $smarty;
        $template_dir = $app->getRootPath() . 'resources/views/';
        $compile_dir = $app->getRootPath() . 'storage/framework/views';
        $this->smarty->template_dir = $template_dir;
        $this->smarty->compile_dir = $compile_dir;
    }
}
class StudentController extends Controller{
    protected $request;
    public function __construct(Request $request)
    {
        $this->request = $request;
    }
    ...
}
```

- 在这个框架中，App类会根据PATH_INFO，通过依赖注入，自动实例化StudentController类。但是现在会一直报错，一直提示$this->smarty为null。也就是说也就是说父类中的smarty没有实例化，**即父类中的构造方法没有调用**。当时我也不知道为啥没有实例化，但是想到了那句"子中构造方，须将参递父。"
- 于是就在子类的构造方法中加上了parent::__construct()。代码如下。

```
class StudentController extends Controller
{
    protected $request;
    public function __construct(Request $request, App $app, Smarty $Smarty)
    {
        parent::__construct($Smarty, $app, $request);
        $this->request = $request;
    }
}
```

- 修改代码之后，确实能跑了。但是今天又遇到了这个问题，就必须打破砂锅问到底了。
- 今天无意间注意到了子类中有构造方法，又想到了之前做的笔记：

> 所谓的重载或覆盖，并不是将父类的方法覆盖。也可以这么理解：**如果子类有调用该方法时，如果在自己类里找不到这个方法，那么它会去找父类里，看有没有这个方法**。这也是多态的理解

- 似乎有了点感觉，就去确认在实例化子类的时候，父类的构造方法会不会调用。事实证明，在实例化子类的同时，**父类的构造方法会调用。**
- 于是就把StudentController类中的构造方法删掉了，而且原来的代码也有点小问题，父类和子类中的request属性重复了。

```
class StudentController extends Controller
{
    // 没有构造方法。
    ...
}
```

- StudentController类中的构造方法删掉之后，代码就能正常的跑起来了。
- 总结：
    - 在实例化子类的时候，如果子类有构造方法，就会立刻调用该方法，如果在自己类里找不到该方法，那么它会去找父类里，看有没有这个方法。
- 其他例子：（验证该问题时自己写的例子）

```php
<?php
// 验证在实例化子类的时候，父类的构造方法会不会调用。
class A
{
}
class B
{
    public function test()
    {
        echo '这是B中的test方法';
    }
}
class C
{
}
class Controller
{
    public $a;
    public $b;
    public function __construct(A $a, B $b)
    {
        $this->a = $a;
        $this->b = $b;
    }
    public function getA()
    {
        return $this->a;
    }
    public function getB()
    {
        return $this->b;
    }
}
class StudentController extends Controller
{
    public $c;
    public function __construct(A $a, B $b, C $c)
    {
        parent::__construct($a, $b);
        $this->c = $c;
    }
}
class App
{
    public $instances = [];
    public static $instance;
    public function make($className)
    {
        if (!isset($instances[$className])) {
            // 创建对象反射类。
            $reflect = new ReflectionClass($className);
```

```
            $constructor = $reflect->getConstructor();
            if (!$constructor) {
                // 如果没有构造方法，那么就不需要传入参数，但是newInstanceArgs方法必须传入数组参数。
                $args = [];
            } else {
                $args = [];
                $params = $constructor->getParameters();
                foreach ($params as $v) {
                    // 获取类型提示类
                    $class = $v->getClass();
                    if ($class) {
                        // 如果是类型提示类
                        $args[] = $this->make($class->getName());
                    }
                }
            }
            $this->instances[$className] = $reflect->newInstanceArgs($args);
        }
        return $this->instances[$className];
    }
    public static function getInstance()
    {
        if (!isset(self::$instance)) {
            self::$instance = new static();
        }
        return self::$instance;
    }
}

class Main
{
    public static function client()
    {
        $app = App::getInstance();
        $className = 'StudentController';
        $obj = $app->make($className);
        // var_dump($obj);
        $obj->getB()->test();
    }
}
Main::client();
// 输出：这是B中的test方法
```

note路径：

- D:\phpStudy\PHPTutorial\WWW\index\myframe2\note\construct