PHP5 面向对象初步

Created by 刀客羽朋

前言 PHP5 面向对象设计

从 OOP 的视角看，不应区分语言。无论是 C++、无论是 Java、无论是.net 还有更多面向对象的语言，只要你了解了 OO 的真谛，便可以跨越语言，让你的思想轻松的跳跃。便没有对于 Java、.net 、 PHP 之间谁强谁弱的争执了。

希望这个介绍 PHP5 面向对象编程（OOP）的资料能让初学者受益，能让更多的 PHPer 开始转向 OO 的编程过程。

相对 PHP4，PHP5 在面向对象方面改变了很多。我们将只介绍 PHP5 环境下的面向对象。而我们必须改变自己来跟随 PHP5 的发展。如果代码结果在你的环境下执行不一致，请确认你的环境为 PHP5。

我们假设读者没有任何面向对象的知识，即使你是第一次听说 OOP，也可以读懂这文章。但我希望你必须对 PHP 有一些了解。

在后面我们将使用一些例子，来逐渐分析 PHP5 的 OO 基础。

面向对象只解决了两个问题，**代码的可扩展性 、 代码的可维护性。**

关于为什么要使用面向对象？在什么时候使用面向对象？这里不讨论。

感谢www.phpchina.com，编程软件环境使用的ZendStudio5.2。

不得不说句，php 越来越像 Java 了。

在本书之后，建议继续读www.PHPchina.com 翻译的 《PHP设计模式》。

仅以这些给我即将降生的 baby。

刀客羽朋
gaoxiangming@hotmail.com
2006-10-12

目录

第一章 PHP5 面向对象基础

1.1 类和对象

Everything is Object：万事万物皆对象。

面向对象的编程（OOP）思想力图使对计算机语言中对事物的描述与现实世界中该事物的本来面目尽可能的一致。

（面向对象语言与我们的生活是想通的，面向对象语言学习起来其实很简单。在应用中更符合我们的生活逻辑。）

类(Class)是用来描述一个对象(Object)：

类描述了每个对象应包括的数据

类描述了每个对象的行为特征

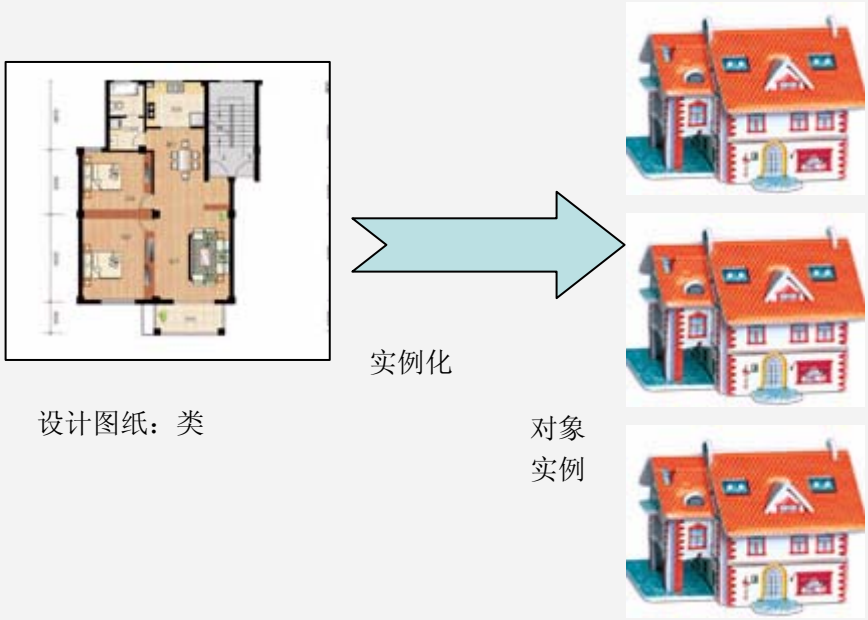
（如上所述，类仅仅是对象的描述，就好像楼房的设计图纸。）

Class/Object：类(class)和对象(object)是面向对象方法的核心概念。

类是对一类事物描述，是抽象的、概念上的定义；
（类好像是在图纸上设计的楼房，楼房设计出来了，但这个楼房并不存在。）

对象是实际存在的该类事物的每个个体，因而也称实例(instance)。
（对象是实实在在存在的，照着楼房的设计图纸，高楼盖起来，可以住进去了。在计算机中，可以理解为，在内存中创建了实实在在存在的一个内存区域存储着这个对象。）

创建对象的过程称为 创建对象 也称为实例化。
看下面的图示，一张楼房的图纸创建了多个别墅（对象）。
思考一下：
它们外观一样么？
它们结构一样么？
它们是一个对象么？

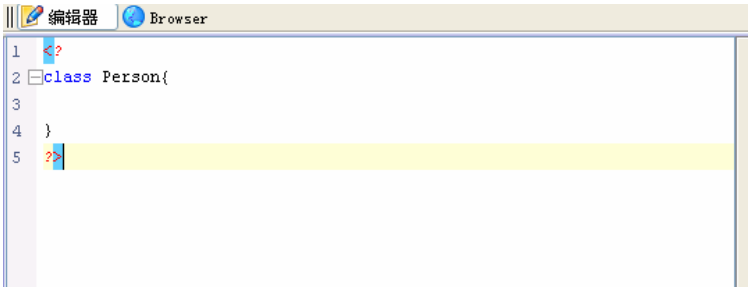


1.2 PHP5 中的类和对象

我们先建立一个基础的类。

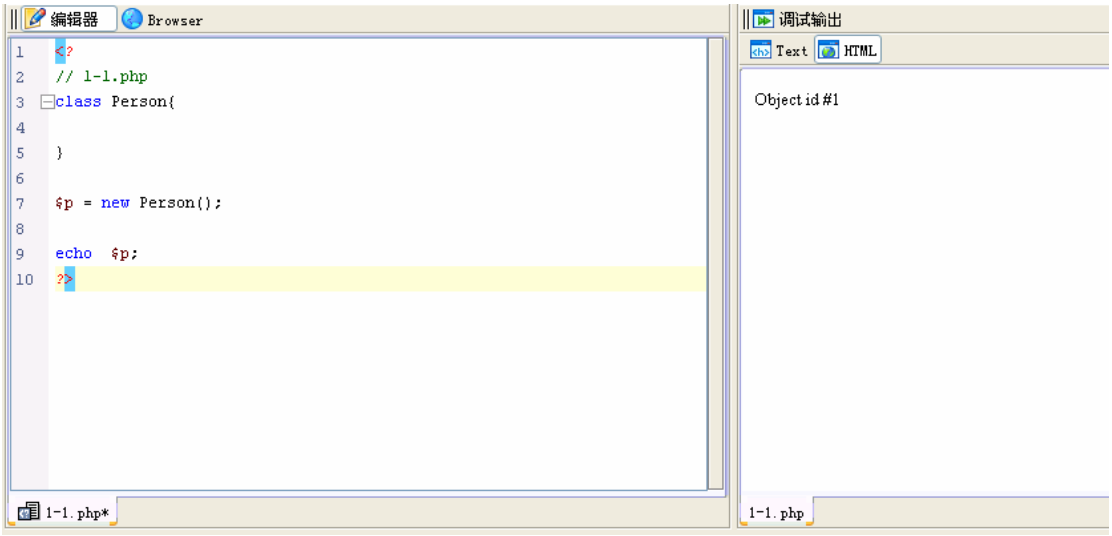
PHP 中使用关键字 **class** 来定义一个类。类的命名一般使用首字符大写，而后每个单词首字符大写连接的方式方便阅读。

例 1-1.PHP



这样，我们就拥有了第一个 PHP 类。

我们继续来使用这个类，使用 **new** 这个关键字创建对象，用 echo 打印$p。



我们定义了一个变量 $p ，使用 **new** 这个关键字创建了一个 Person 的对象。

打印变量$p，我们看到输出 Object id #1 提示这是一个对象。

> $p = new Person();也可以写成 $p = new Person;
> 但不建议使用后面的这种方式。

1.3 PHP5 中的属性

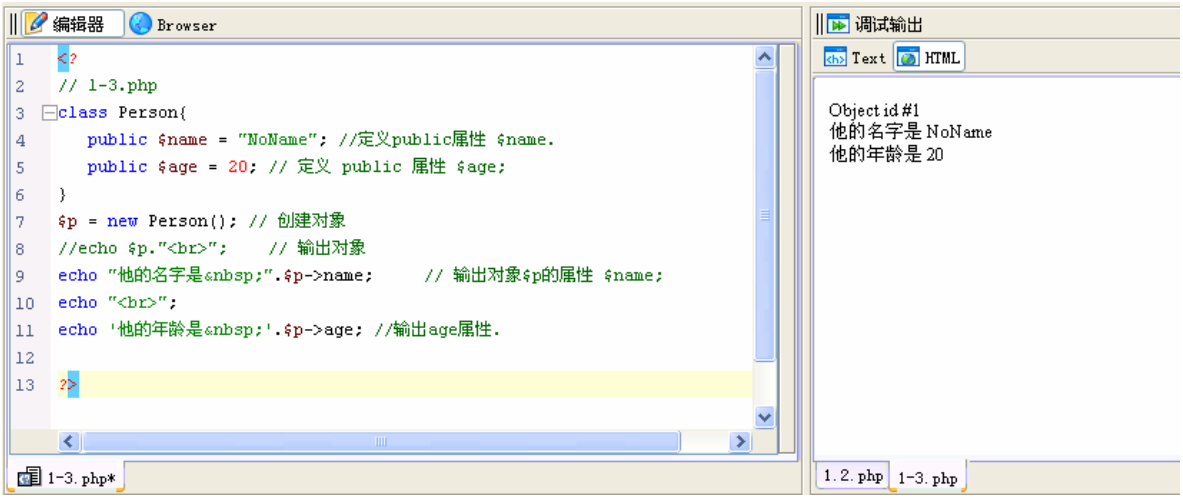
属性：用来描述对象的数据元素称为对象的属性（也称为数据/状态）

在 PHP5 中，属性指在 class 中声明的变量。在声明变量时，必须使用 public private protected 之一进行修饰，定义变量的访问权限。

- Public（公开）：可以自由的在类的内部外部读取、修改。
- Private（私有）：只能在这个当前类的内部读取、修改。
- Protected（受保护）：能够在这个类和类的子类中读取和修改。

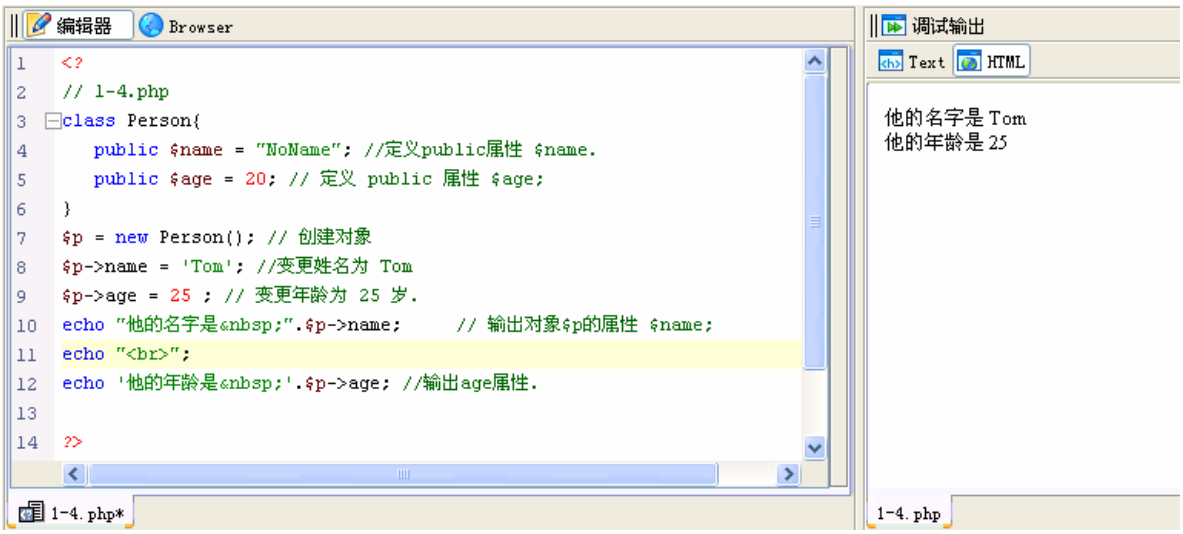
属性的使用：通过引用变量的 -> 符号调用变量指向对象的属性。
在方法内部通过 $this-> 符号调用同一对象的属性。

例 1-3：public 的属性。



Person 类有两个属性，$name 和 $age，在实例化后，使用$p->name 和 $p->age 打印出属性的内容。
当然，你可以在属性定义时不设置初始值，那样的话，就打印不出任何结果了。

例 1-4:
改变对象的属性，注意 8 行和 9 行代码，还有输出结果的变化。我们看到输出的属性值被改变了。

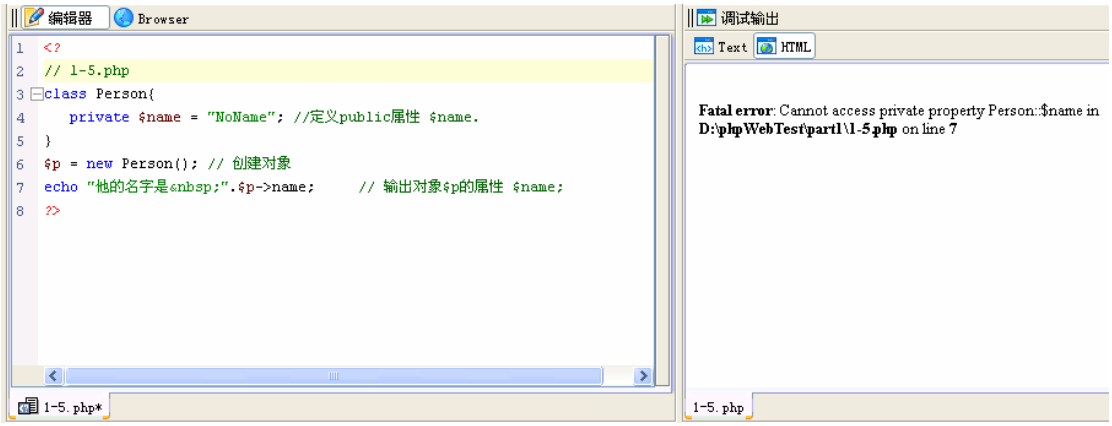


创建一个 Person 的对象，改变这个对象的属性。为它命名，查看它的名字。你就是机器里面这个 Person 对象的上帝，按照你定义的规则，这个实实在在在内存中的 Person 对象被创建了，而且它有了可以改变的属性。
现在，我们就是计算机世界的上帝，准备好创造世界吧。

Private 修饰的属性，在当前对象以外不能访问。设置私有属性是为了进行数据的隐藏。

隐藏：指对象的一种保护机制，使得它的属性或方法不被外部的程序直接访问。

例 1-5：访问时会报错如下。



私有属性不能被外部访问，在下一节介绍这样做的好处。

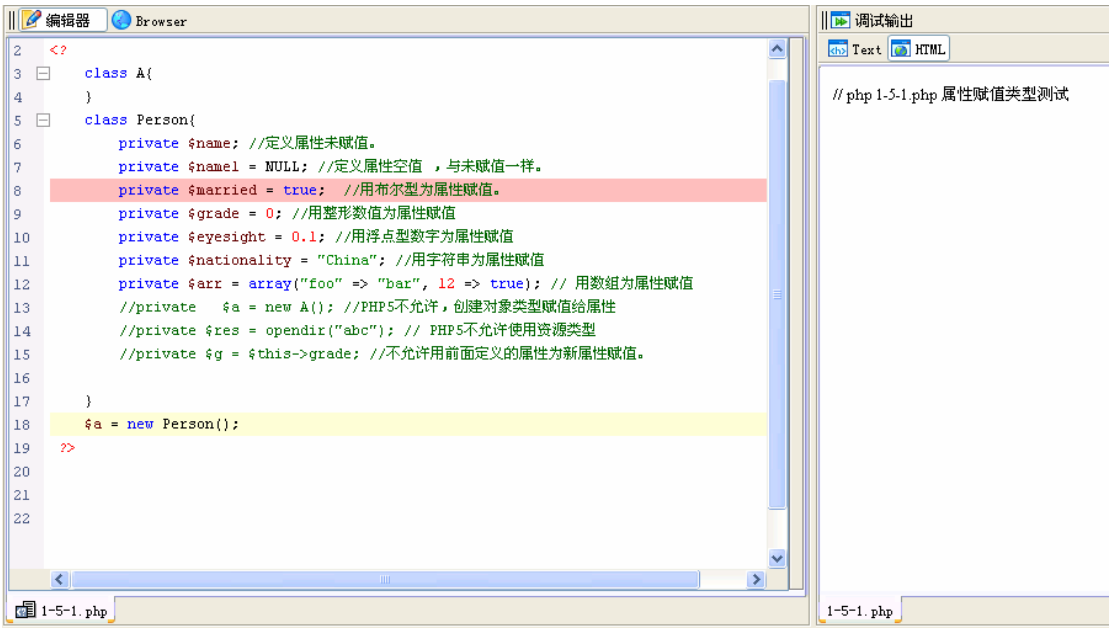
■ 属性的初值

在 PHP5 中，在属性定义可以不设置初值，或者赋予以下红色类型的初值。

PHP 中简单类型有 8 种，分别是：

- **数值类型**
  - **boolean** 布尔类型
  - **integer** 整型
  - **float** 浮点型，也称为 **double** 双精度浮点型
  - **string** 字符串
- **复合类型**
  - **array** 数组
  - **object** 对象
- **特殊类型**
  - resource 资源
  - **NULL**

例：1-5-1.php




```php
<?
    class A{
    }
    class Person{
        private $name; //定义属性未赋值。
        private $name1 = NULL; //定义属性空值 ，与未赋值一样。
        private $married = true;  //用布尔型为属性赋值。
        private $grade = 0; //用整形数值为属性赋值
        private $eyesight = 0.1; //用浮点型数字为属性赋值
        private $nationality = "China"; //用字符串为属性赋值
        private $arr = array("foo" => "bar", 12 => true); // 用数组为属性赋值
        //private   $a = new A(); //PHP5不允许，创建对象类型赋值给属性
        //private $res = opendir("abc"); // PHP5不允许使用资源类型
        //private $g = $this->grade; //不允许用前面定义的属性为新属性赋值。

    }
    $a = new Person();
?>
```

注意：

在上面例子中，第 13 行，尝试创建对象并将值赋予属性$a 会报错。

第 14 行，建立资源并复制给$res 会出现错误。

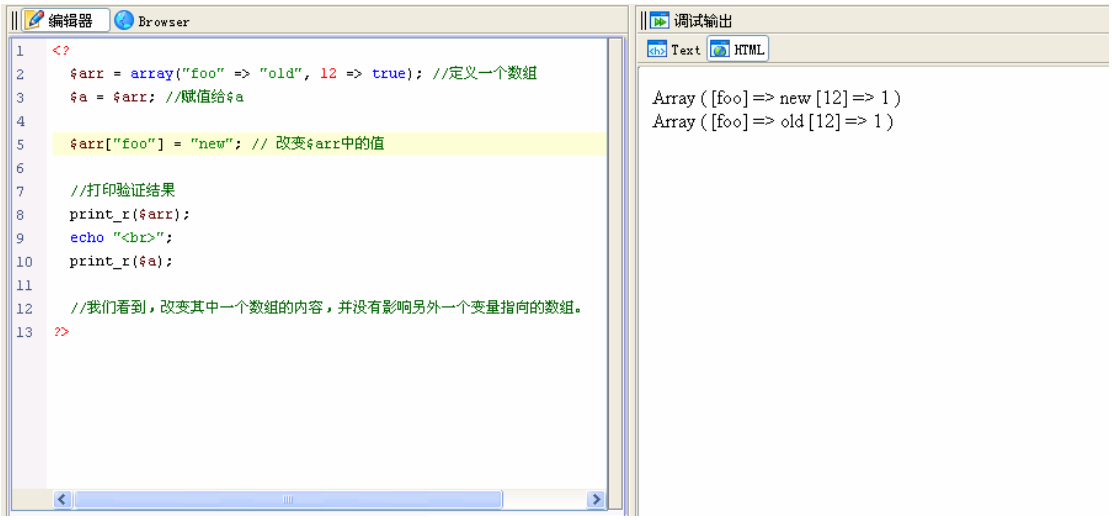
第 15 行，使用上面定义的属性为新属性赋值也会产生错误。

（在 Java 中，可以作 13 行和 15 行这样的操作。

PHP5 中定义属性的默认值，被限制到最简单的方式。其它的操作，交给构造方法操作，后面内容中将讲解构造方法。）

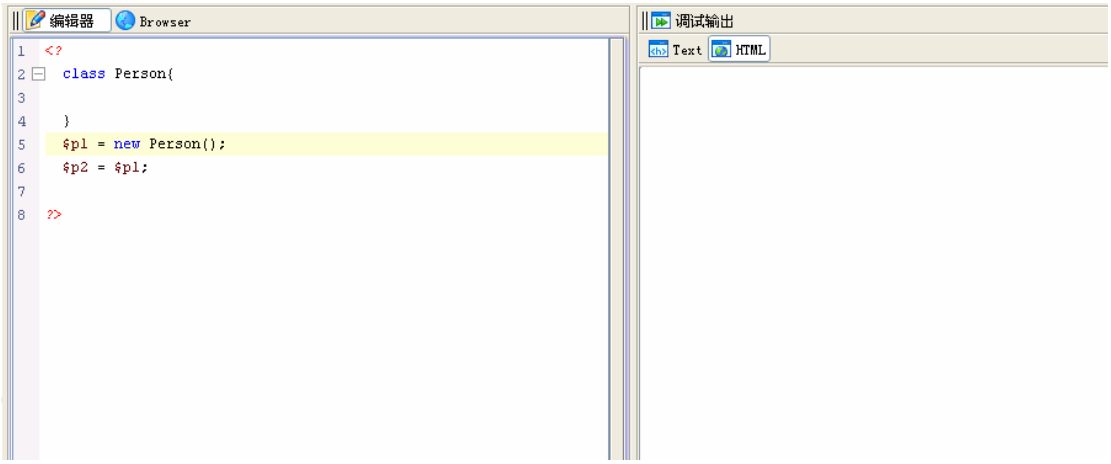
- 变量与引用变量

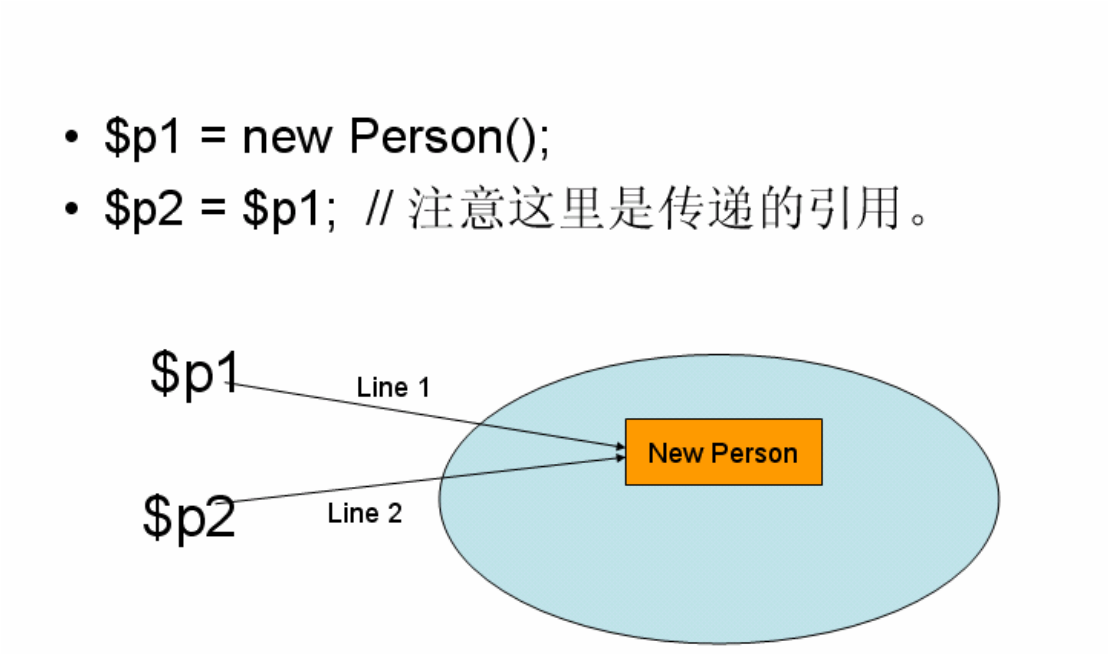
普通变量间的传值方法，就是值的赋值。比如数组。



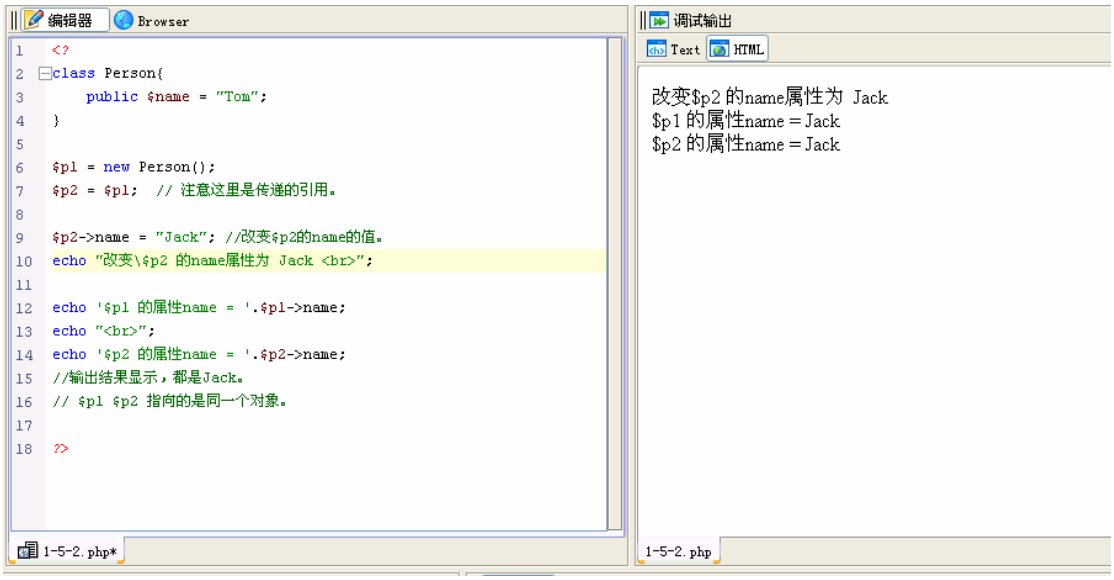

```php
<?
  $arr = array("foo" => "old", 12 => true); //定义一个数组
  $a = $arr; //赋值给$a

  $arr["foo"] = "new"; // 改变$arr中的值

  //打印验证结果
  print_r($arr);
  echo "<br>";
  print_r($a);

  //我们看到，改变其中一个数组的内容，并没有影响另外一个变量指向的数组。
?>
```

而指向对象的变量，是一个引用变量。在这个变量里面存储的是所指向对象的内存地址。引用变量传值时，传递的是这个对象的指向。而非复制这个对象。





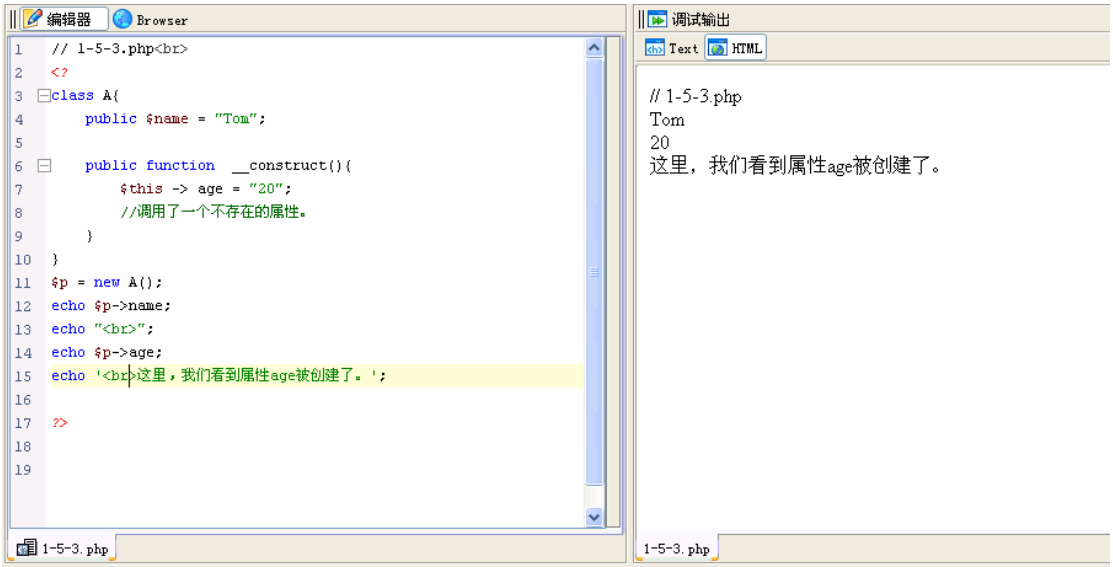
例 1-5-1.php 说明了这个问题。



- 属性的扩充

$this 指当前对象。
$this-> 调用当前对象的 属性或者方法。

在类中使用$this-> 调用一个未定义的属性时，PHP5 会自动创建一个属性供使用。
这个被创建的属性，默认的方法权限是 public。
例：1-5-3.php



1.4 PHP5 中的方法

方法：对对象的属性进行的操作称为对象的方法（也称为行为/操作）。

过程 函数 方法

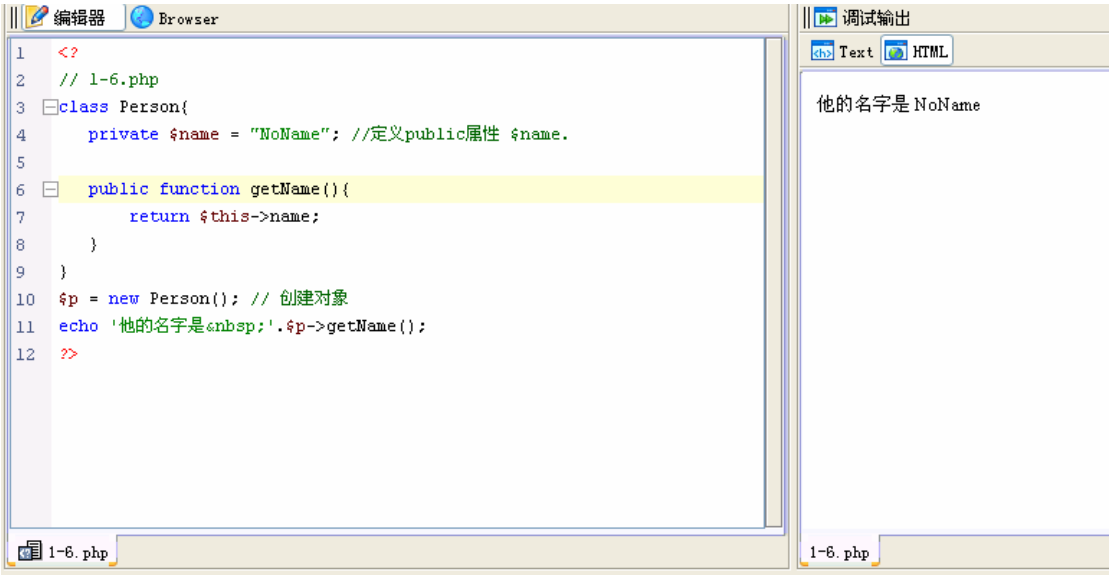
过程：过程是编制程序时定义的一个语句序列，用来完成某种指定的操作。

函数：函数有返回值，也是定义的语句序列。

方法：在面向对象概念中，类里面的一段语句序列。

一般来说，在面向对象概念中，函数和方法两个名词是通用的。

例 1-6：通过方法读取属性。



上面的例子将属性设置为 private ，同时声明了 public 的 getName()方法，用来获取属性$name 的值，调用 getName()方法就会通过 return $this->name 返回 $name 的值。

注意：这里，方法内部调用本地属性时，使用 $this->name 来获取属性。在这个例子中，设置了公开的 getName()方法，即用户只能获取$name，而无法改变他的值。这就是封装的好处。

- **封装指的是将对象的状态信息（属性）和行为（方法）捆绑为一个逻辑单元的机制。**
- **PHP5 中通过将数据封装、声明为私有的(private)，再提供一个或多个公开的（public）方法实现对该属性的操作，以实现下述目的：**
  - 隐藏一个类的实现细节；
  - 防止对封装数据的未经授权的访问。使用者只能通过事先定制好的方法来访问数据，可以方便地加入控制逻辑，限制对属性的不合理操作；
  - 有利于保证数据的完整性；
  - 便于修改，增强代码的可维护性；

- **方法的参数**

通过方法定义时的参数，可以向方法内部传递变量。

如下第 5 行，定义方法时定义了方法参数$_a,。 使用这个方法时，可以向方法内传递参数变量。方法内接受到的变量是局部变量，仅在方法内部有效。可以通过向属性传递变量值的方式，让这个变量应用于整个对象。

```
3  class Person{
4      private $a;
5      function setA($_a){
6          $this->a = $_a; // 局部变量$_a 传值给属性$a;
7      }
8      function getA(){
9          return $this->a; // 返回属性$a;
10     }
11 }
12
13 $p = new Person(); // 创建Person的对象。
14 $p->setA(100);
15 echo $p->getA();
```

```
100
```

如果声明这个方法有参数，而调用这个方法时没有传递参数，或者参数数量不足，系统会报出错误。

第 14 行调用方法 setA 的时候，没有传递参数。

编辑器 Browser

```
1  <? // 1-6-2.php
2
3  class Person{
4      private $a;
5      function setA($_a){
6          $this->a = $_a; // 局部变量$_a 传值给属性$a;
7      }
8      function getA(){
9          return $this->a; // 返回属性$a;
10     }
11 }
12
13 $p = new Person(); // 创建Person的对象。
14 $p->setA();
15 echo $p->getA();
16 ?>
17
18
19
```

1-6-2.php

调试输出

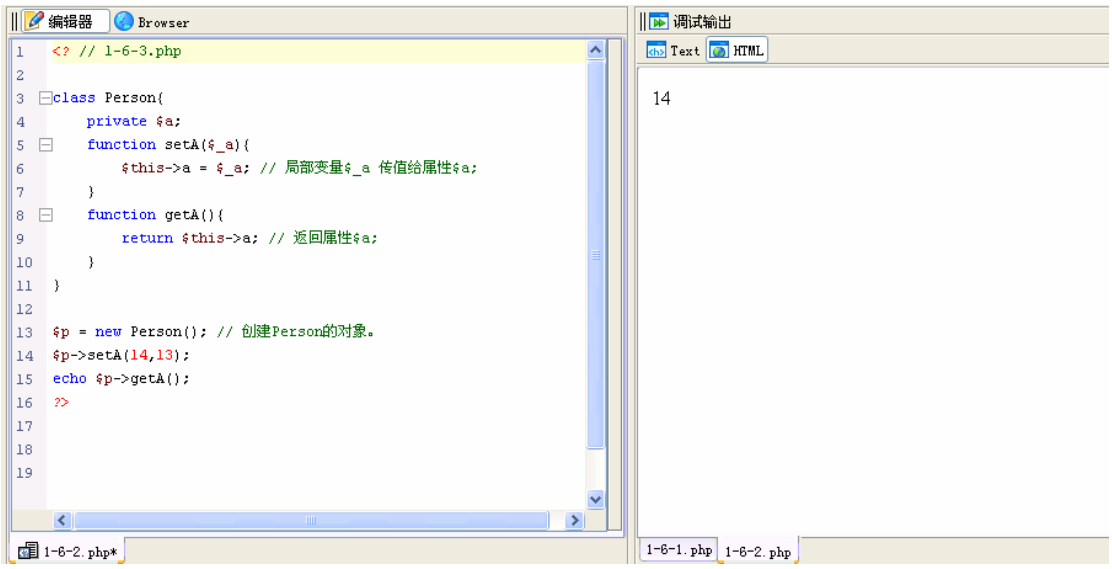
Text HTML

Warning: Missing argument 1 for Person::setA(), called in D:\phpWebTest\part1\1-6-2.php on line 14 and defined in **D:\phpWebTest\part1\1-6-2.php** on line **5**

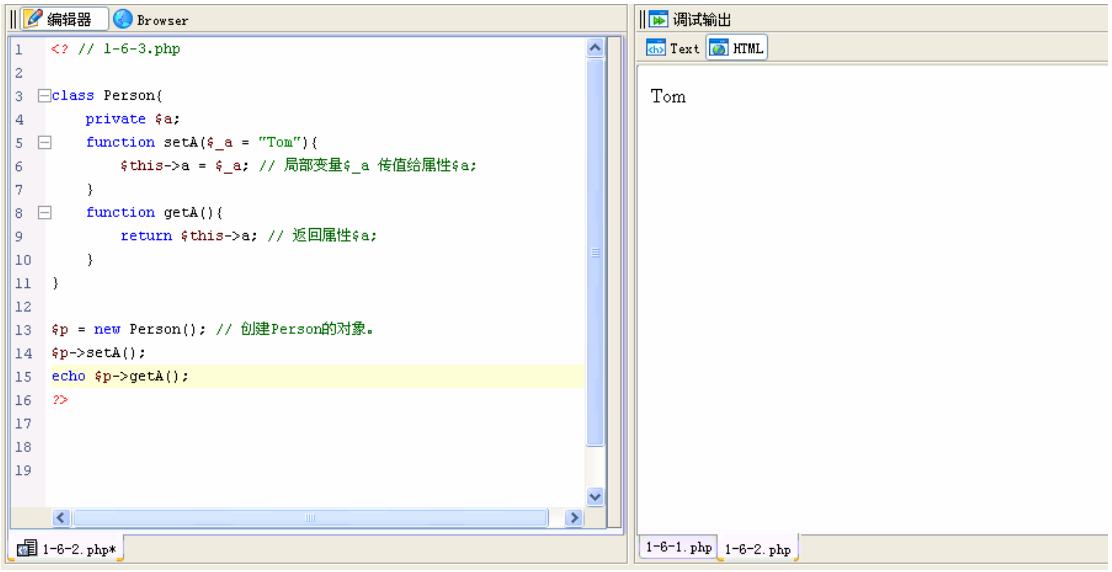
1-6-1.php 1-6-2.php

如果参数数量超过方法定义参数的数量，PHP 就忽略多于的参数。不会报错。

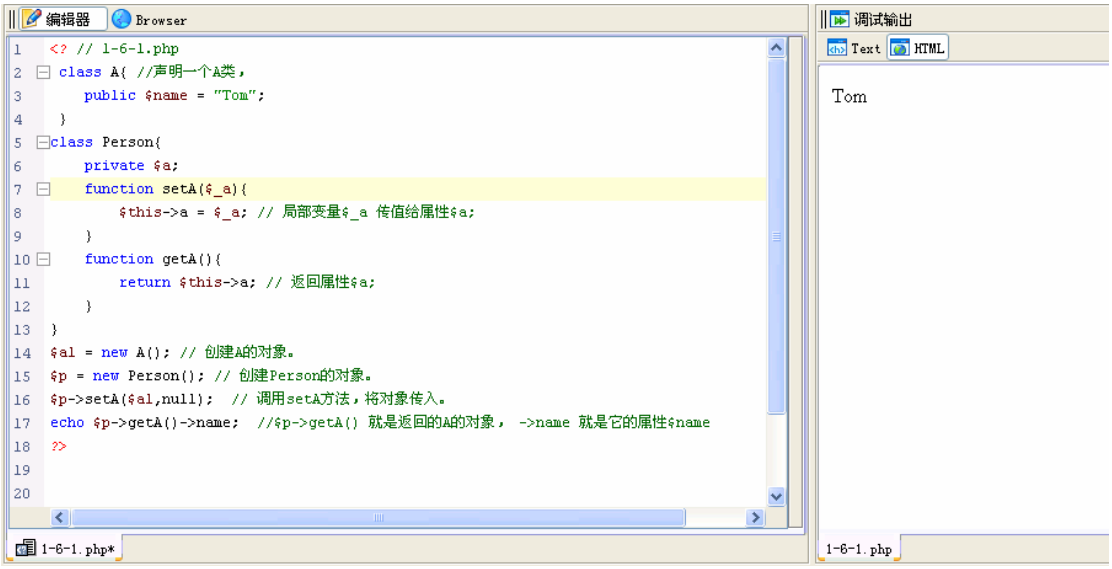
注意第 14 行。



可以在函数定义时为参数设定默认值。
在调用方法时，如果没有传递参数，将使用默认值填充这个参数变量。



可以向一个方法内部传递另外一个对象的引用变量。
在方法内部，这个引用可以一直传递，在需要的时候，调用这个对象的属性和方法。
例 1-6-1.php



再次提示

在 PHP5 中，指向对象的变量是引用变量。在这个变量里面存储的是所指向对象的内存地址。引用变量传值时，传递的是这个对象的指向。而非复制这个对象。

这与其它类型赋值有所不同。

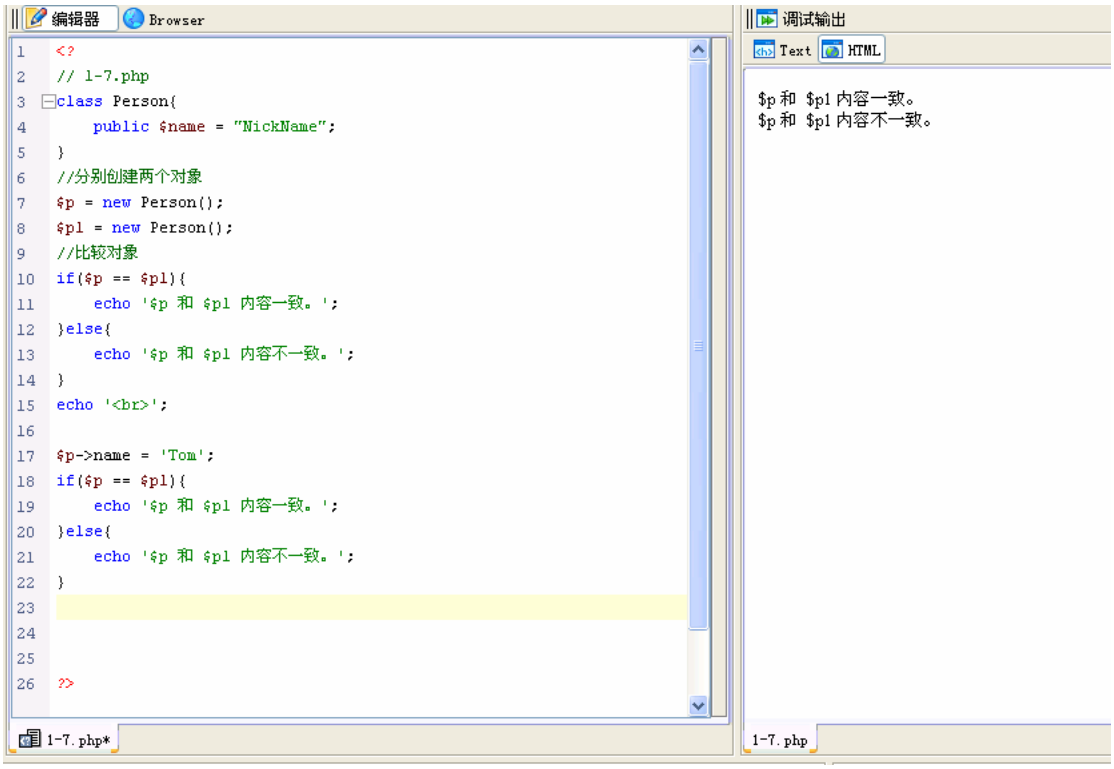
1.5 对象的比较

在 PHP 中有 = 赋值符号、== 等于符号 和 === 全等于符号，这些符号代表什么意思？

- 当使用比较操作符（==）时，对象以一种很简单的规则比较：当两个对象有相同的属性和值，属于同一个类且被定义在相同的命名空间中，则两个对象相等。等于符号比较对象时，比较对象是否有相同的属性和值。

  注意：== 比较两个不同的对象的时候，可能相等也可能不等。

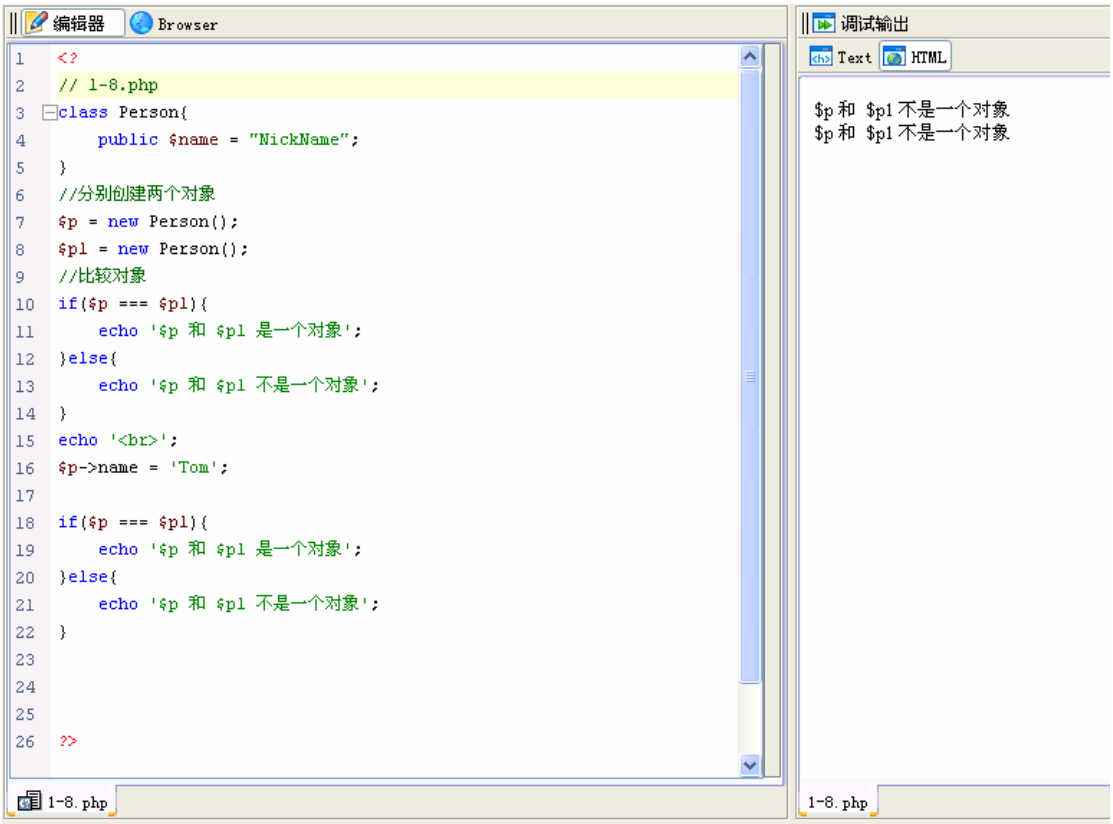
例：1-7：



使用 == 符号比较两个对象 ，比较的仅仅是两个对象的内容是否一致。

- 当使用全等符（===）时，当且仅当两个对象指向相同类（在某一特定的命名空间中）的同一个对象时才相等。
  是否在是同一个对象，两边指向的对象是否有同样的内存地址。

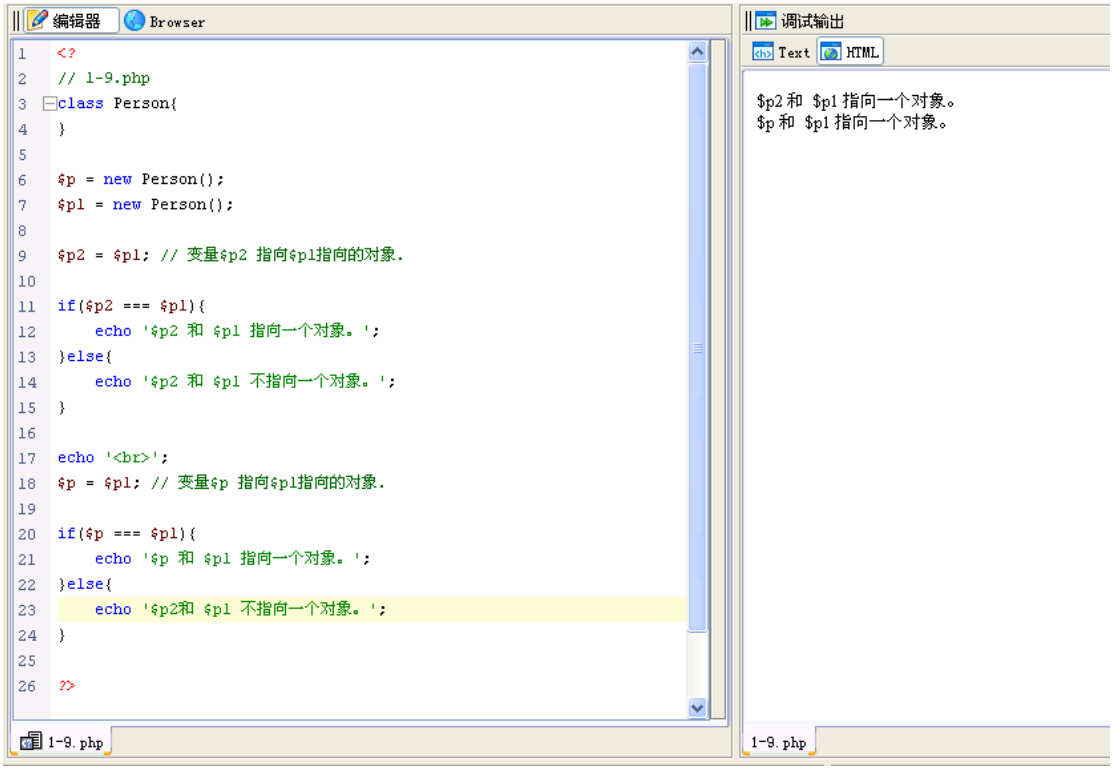
例 1-8.PHP



结果=== 比较的是两个变量是否一个对象。

- 一个等于符号（=）表示赋值，是赋值计算。
  如果将对象赋予变量，是指变量将指向这个对象。

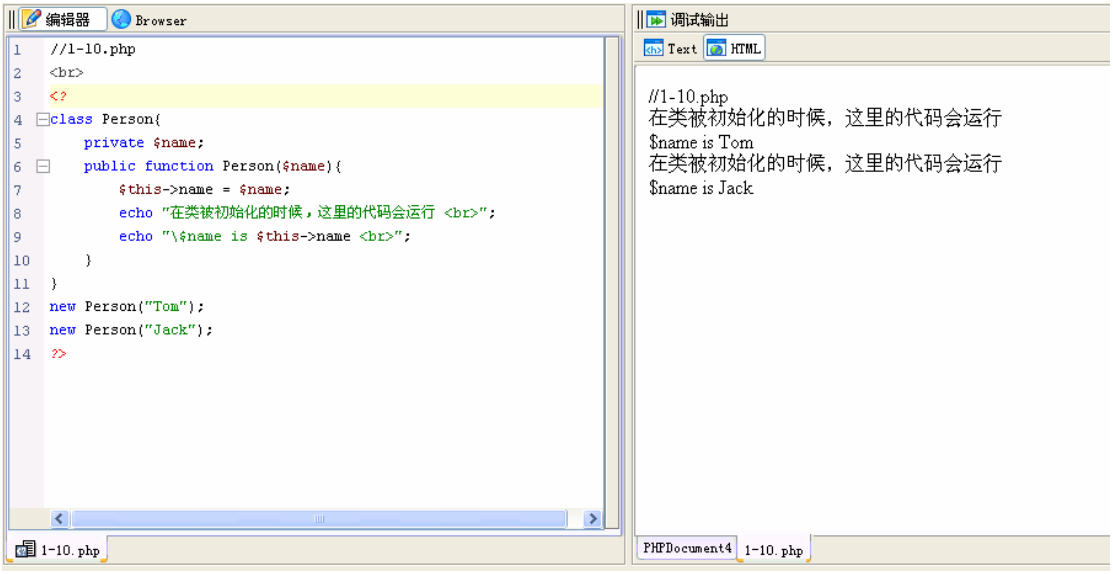
例：1-9



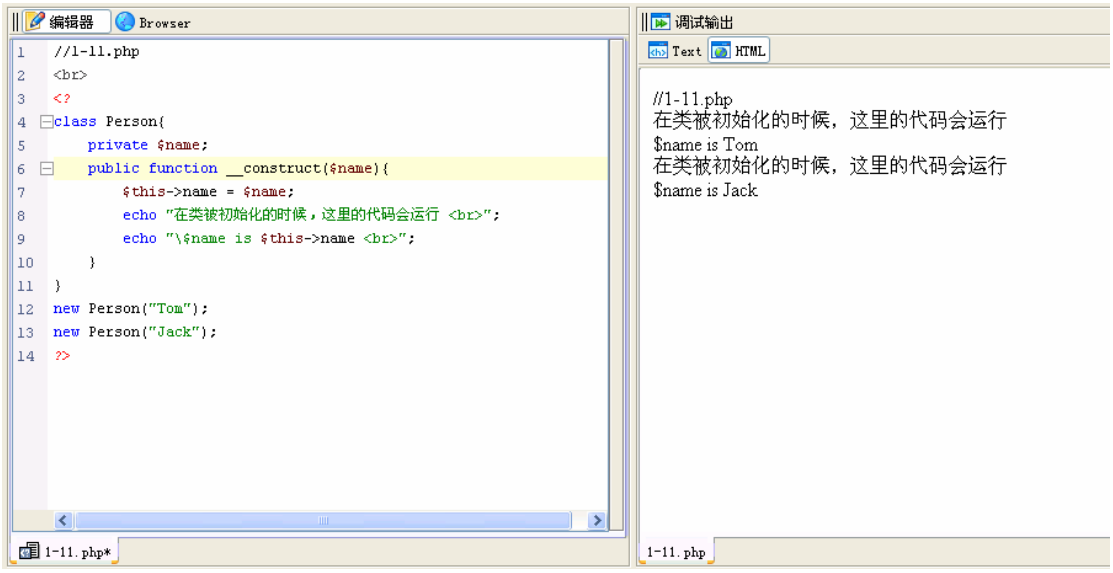
1.6 构造函数

构造方法又称为构造函数，是对象被创建时自动调用的方法，用来完成类初始化的工作。
构造函数和其它函数一样，可以传递参数，可以设定参数默认值。
构造函数可以调用属性，可以调用方法。
构造函数可以被其它方法显式调用。

在 PHP4 中使用与类名同名的方法为构造函数。在 PHP5 中依然支持了这种方式，但不建议再使用这种方式。
例 1-10.php



在 PHP5 中规定构造函数使用_ _construct() 函数表示构造函数，注意是两个 _ 。
例 1-11.php



1.7 析构函数与 PHP 的垃圾回收机制

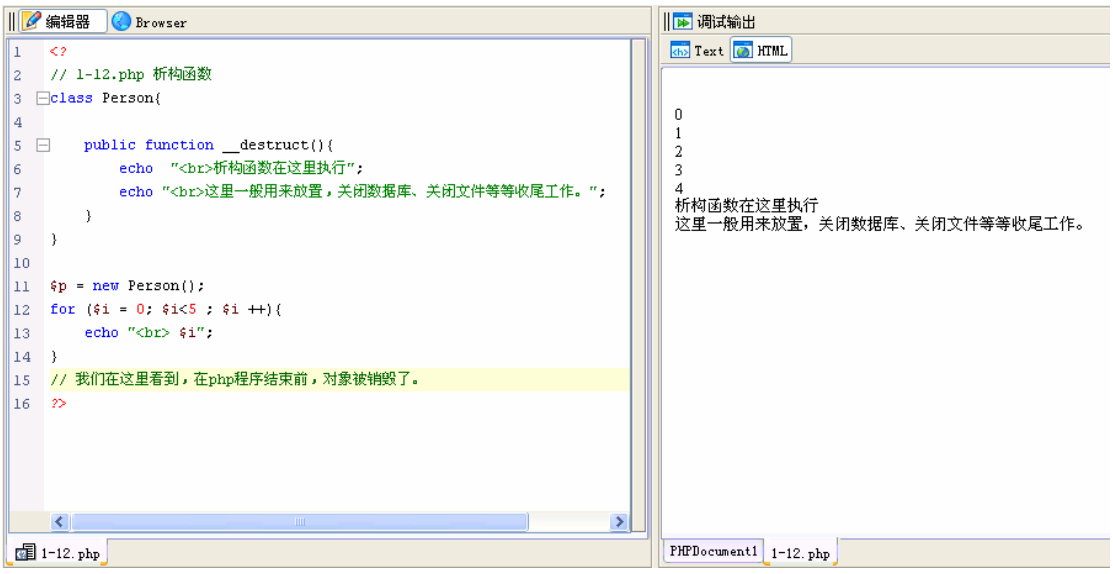
析构函数：当某个对象成为垃圾或者当对象被显式销毁时执行。

GC(Garbage Collector)
在 PHP 中，没有任何变量指向这个对象时，这个对象就成为垃圾。PHP 会将其在内存中销毁。
这是 PHP 的 GC(Garbage Collector)垃圾处理机制,防止内存溢出。
当一个 PHP 线程结束时，当前占用的所有内存空间都会被销毁，当前程序中的所有对象同样被销毁。

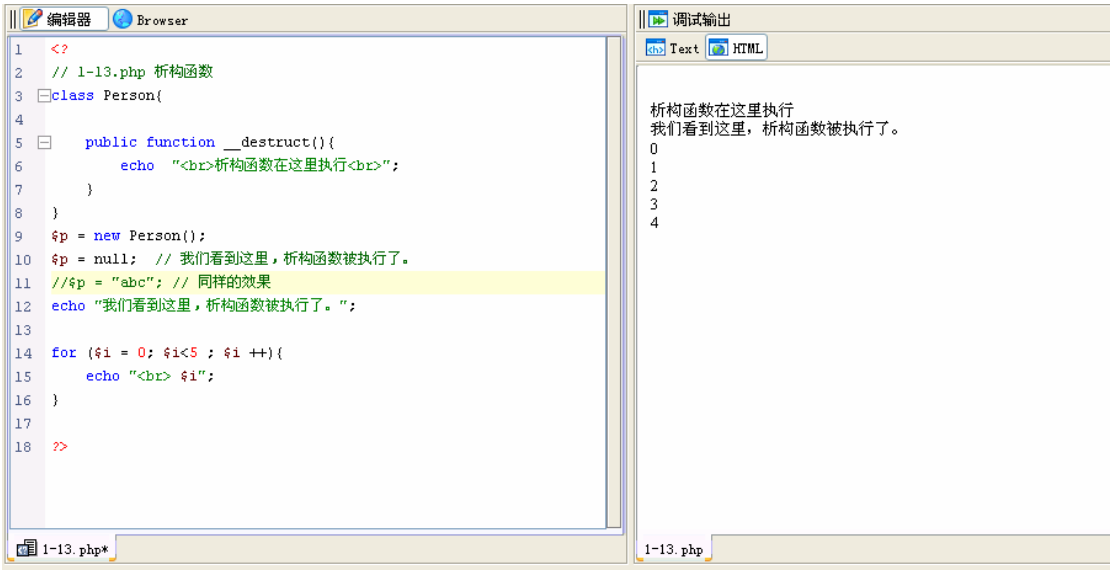
__destruct() 析构函数，是在垃圾对象被回收时执行。

析构函数也可以被显式调用,但不要这样去做。
析构函数是由系统自动调用的，不要在程序中调用一个对象的虚构函数。
析构函数不能带有参数。

例 1-12.php 。程序结束前，所有对象被销毁。
析构函数被调用了。

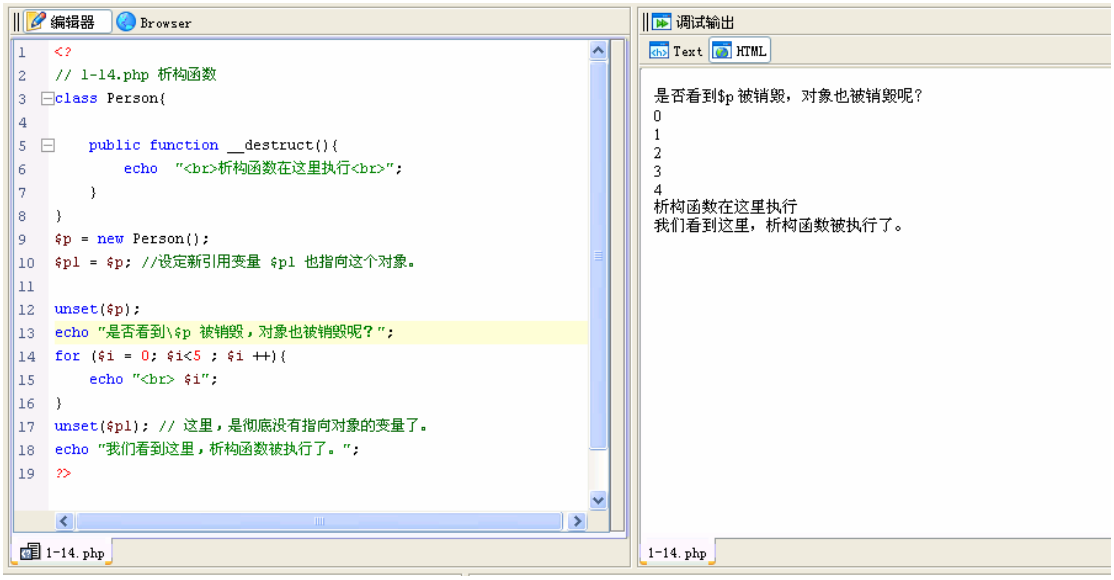


例 1-13.php 当对象没有指向时，对象被销毁。



上面的例子第 10 行，我们将$p 设置为空或者第 11 行赋予$p 一个字符串，这样$p 之前指向的对象就成为了垃圾对象。PHP 将这个对象垃圾销毁。

例 1-14.php unset 变量

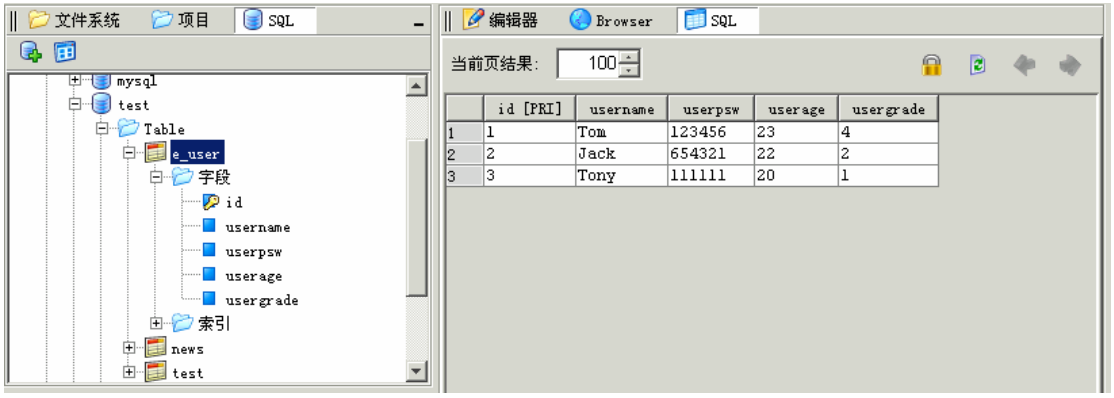


unset 一个引用变量时。
unset 销毁的是指向对象的变量，而不是这个对象。

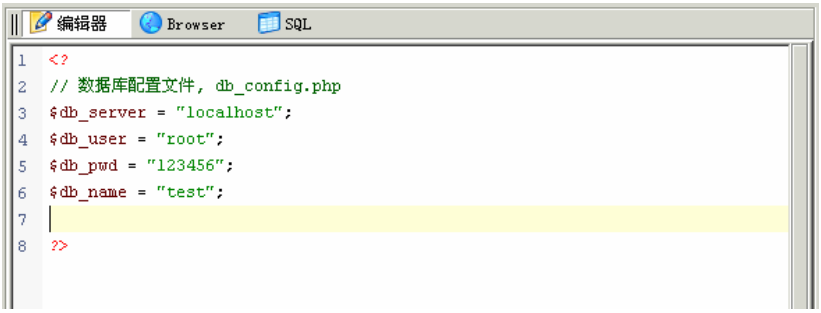
1-8 面向对象实例

我们使用面向过程的方式和面向对象的方式分别写几个程序，理解面向对象编程带来的优势。

数据库使用 mysql 数据库， 数据库结构和数据如下图所示。

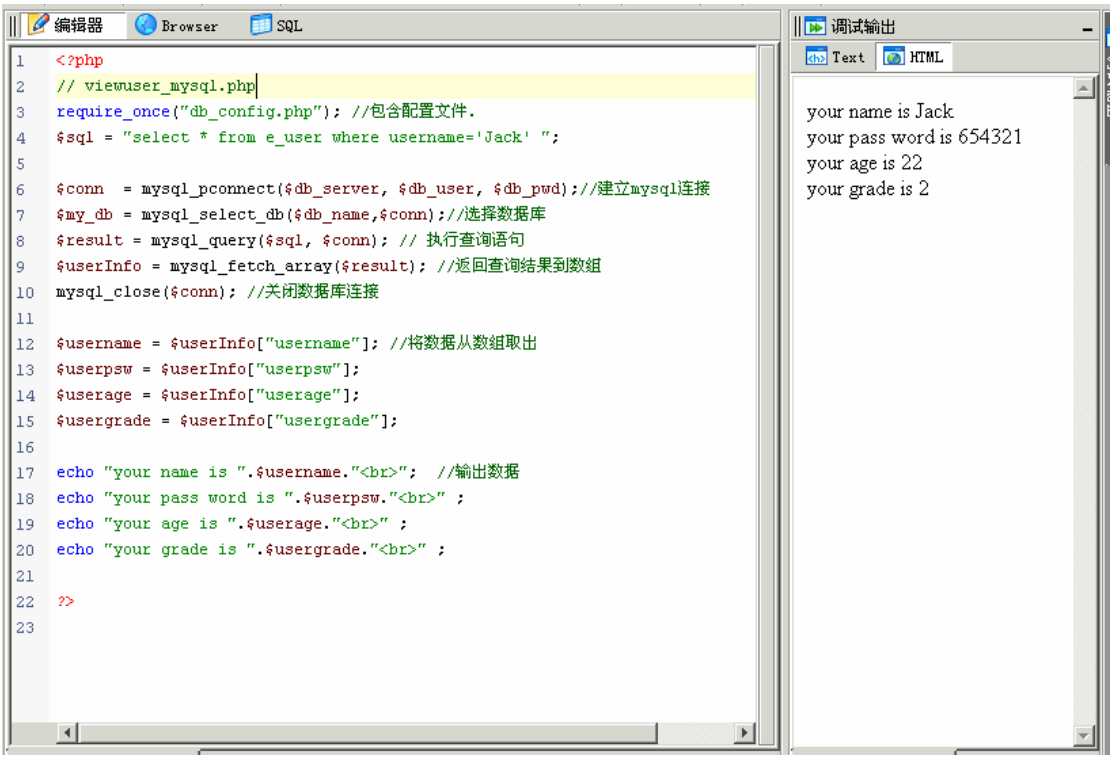


先写一个数据库配置文件如下：



我们先写一个纯粹面向过程的方式来读取数据库中的用户信息。

例： example



这个思维模式我们太熟悉不过了。

1. 读取配置文件中的数据库参数。
2. 建立数据库连接。
3. 选择数据库。
4. 执行 sql 语句。
5. 将数据返回给数组。
6. 将每个数据内容取出。
7. 将信息显示。

现在我们写一个面向对象的取数据库信息的内容。

例：example1

编辑器 Browser SQL

```php
<?
// class_user.php
class UserInfo{
    private $userName;  //属性,用户名
    private $userPSW ;  //属性,用户密码
    private $userAge ;  //属性,用户年龄
    private $userGrade ; //属性,用户级别
    private $userInfo; //存储数据库返回信息的数组变量.

    public function __construct($name){
        require_once("db_config.php"); //包含配置信息.
        $sql = "select * from e_user where username='$name' "; //书写sql
        $conn  = mysql_pconnect($db_server, $db_user, $db_pwd);//建立mysql连接
        $my_db = mysql_select_db($db_name,$conn);//选择数据库
        $result = mysql_query($sql, $conn); // 执行查询语句
        $this->userInfo = mysql_fetch_array($result); //返回查询结果到数组
        mysql_close($conn); //关闭数据库连接
        $this->getInfo(); //调用传递信息的方法.
    }
    // 获取信息传递给属性的方法
    private function getInfo(){
        $this->userName = $this->userInfo["username"];
        $this->userPSW = $this->userInfo["userpsw"];
        $this->userAge = $this->userInfo["userage"];
        $this->userGrade = $this->userInfo["usergrade"];
    }

    //返回每个属性的public 方法.
    public function getUserName(){
        return $this->userName;
    }

    public function getUserPSW(){
        return $this->userPSW;
    }

    public function getUserAge(){
        return $this->userAge;
    }

    public function getUserGrade(){
        return $this->userGrade;
    }
}
?>
```

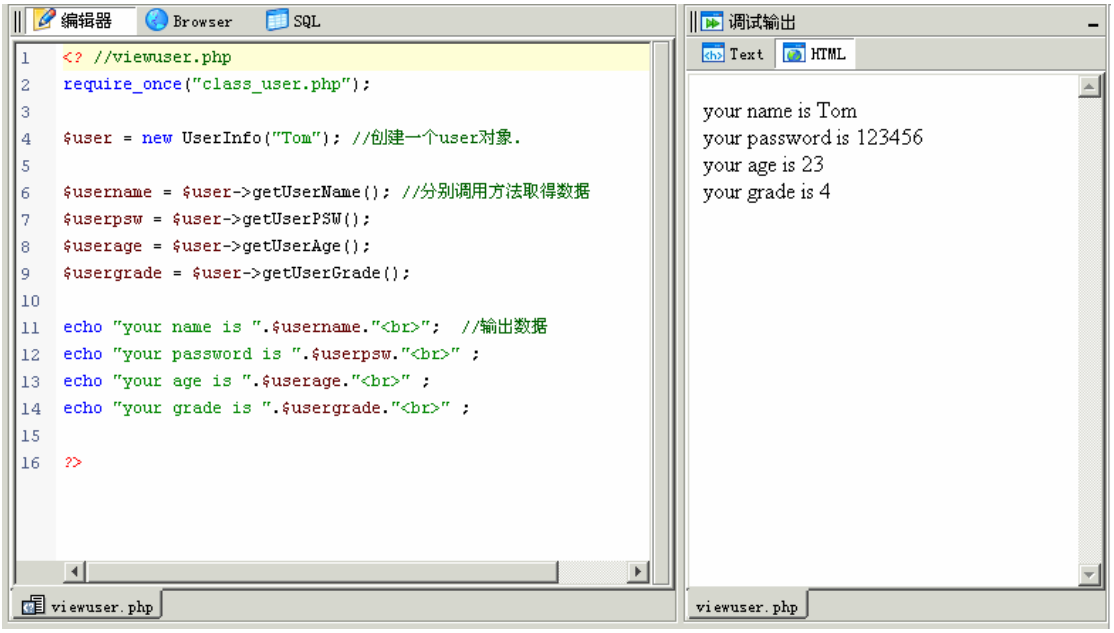
写 class 好像麻烦了些，但优点是结构清晰、扩展、重用和维护方便。

使用这个类非常复合我们看世界的眼光。
现在我们再做一次上帝，follow me。

一个典型的面向对象的案例。
大象放进冰箱里需要几步？
1. 打开冰箱门。
2. 大象放进去。
3. 关上冰箱门。

显示用户 Tom 的信息需要几步？
1. 创建 Tom 出来。
2. 让这个 Tom 告诉我们关于他的信息内容。
3. 显示这些信息。

例：



Tom 这个对象是如何创建的？创建时候做了什么？从那个服务器读取的？
从那个数据库读取的？从那个表单读取的？Tom 的信息是如何读取的？
这些问题，在这里我们不需要再考虑。写刚才那个 user 类的时候，已经考虑过了。
使用这个对象，就像使用家里的冰箱和微波炉一样方便、自然。
把 Tom 换成换成 jack 试试？

那现在，你愿意使用流水账一样的方式写代码，还是更自然的面向对象呢？