# 简介

在 PHP 中，类的成员变量也被称为属性。它们是类定义的一部分， 用来表现一个实例的状态（也就是区分类的不同实例）。

# 读取器和设定器

 在具体实践中，常常会想用一个稍微特殊些的方法实现属性的读写。 例如，如果有需求每次都要对 label 属性执行 trim 操作， 就可以用以下代码实现：

$object->label = trim($label);

上述代码的缺点是只要修改 label 属性就必须再次调用 trim() 函数。 若将来需要用其它方式处理 label 属性，比如首字母大写， 就不得不修改所有给 label 属性赋值的代码。这种代码的重复会导致 bug， 这种实践显然需要尽可能避免。

为解决该问题，Yii 引入了一个名为 yii\base\Object 的基类， 它支持基于类内的 getter 和 setter（读取器和设定器）方法来定义属性。 如果某类需要支持这个特性，只需要继承 yii\base\Object 或其子类即可。

****信息：****几乎每个 Yii 框架的核心类都继承自 yii\base\Object 或其子类。 这意味着只要在核心类中见到 getter 或 setter 方法，就可以像调用属性一样调用它。

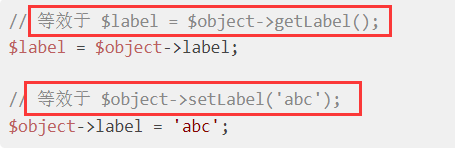
getter 方法是名称以 get 开头的方法，而 setter 方法名以 set 开头。

方法名中 get 或 set 后面的部分就定义了该属性的名字。如下面代码所示， getter 方法 getLabel() 和 setter 方法 setLabel() 操作的是 label 属性：



（详细解释：getter 和 setter 方法创建了一个名为 label 的属性， 在这个例子里，它指向一个私有的内部属性 \_label。）

getter/setter 定义的属性用法与类成员变量一样。两者主要的区别是： 当这种属性被读取时，对应的 getter 方法将被调用；而当属性被赋值时， 对应的 setter 方法就调用。如：



## 只读属性

只定义了 getter 没有 setter 的属性是只读属性。

## 只写属性

类似的，只有 setter 方法而没有 getter 方法定义的属性是只写属性， 尝试读取这种属性也会触发异常。使用只写属性的情况几乎没有。

# 特殊规则和限制

* 这类属性的名字是不区分大小写的。如，$object->label 和 $object->Label 是同一个属性。 因为 PHP 方法名是不区分大小写的。
* 如果此类属性名和类成员变量相同，以后者为准。例如， 假设以上 Foo 类有个 label 成员变量，然后给 $object->label = 'abc' 赋值， 将赋给成员变量而不是 setter setLabel() 方法。
* 这类属性不支持可见性（访问限制）。定义属性的 getter 和 setter 方法是 public、protected 还是 private 对属性的可见性没有任何影响。
* 这类属性的 getter 和 setter 方法只能定义为非静态的，若定义为静态方法（static）则不会以相同方式处理。
* 对不确定有无魔术方法（getter 或 setter）的属性正常调用 property\_exists() 将不会生效。你应该分别调用 [canGetProperty()](https://www.yiichina.com/doc/api/2.0/yii-base-baseobject" \l "canGetProperty()-detail) 或 [canSetProperty()](https://www.yiichina.com/doc/api/2.0/yii-base-baseobject" \l "canSetProperty()-detail) 。