# 简介

行为是 [yii\base\Behavior](https://www.yiichina.com/doc/api/2.0/yii-base-behavior) 或其子类的实例。

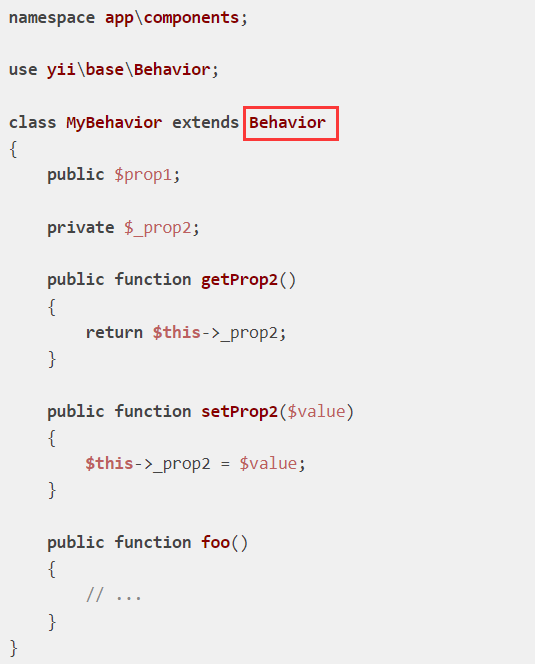
行为，也称为 [mixins](http://en.wikipedia.org/wiki/Mixin)， 可以无须改变类继承关系即可增强一个已有的 [组件](https://www.yiichina.com/doc/api/2.0/yii-base-component) 类功能。

 当行为附加到组件后，它将“注入”它的方法和属性到组件， 然后可以像访问组件内定义的方法和属性一样访问它们。

此外，行为通过组件能响应被触发的[事件](https://www.yiichina.com/doc/guide/2.0/basic-events)，从而自定义或调整组件正常执行的代码。

# 定义行为

要定义行为，通过继承 [yii\base\Behavior](https://www.yiichina.com/doc/api/2.0/yii-base-behavior) 或其子类来建立一个类。如：



以上代码定义了行为类 app\components\MyBehavior 并为要附加行为的组件提供了两个属性 prop1 、 prop2 和一个方法 foo() 。

注意属性 prop2 是通过 getter getProp2() 和 setter setProp2() 定义的。 能这样用是因为 yii\base\Object 是 [yii\base\Behavior](https://www.yiichina.com/doc/api/2.0/yii-base-behavior) 的祖先类，此祖先类支持用 getter 和 setter 方法定义[属性](https://www.yiichina.com/doc/guide/2.0/basic-properties)。

因为这是一个行为类，当它附加到一个组件时，该组件也将具有 prop1 和 prop2 属性和 foo() 方法。

****提示：****在行为内部可以通过 [yii\base\Behavior::$owner](https://www.yiichina.com/doc/api/2.0/yii-base-behavior" \l "$owner-detail) 属性访问行为已附加的组件。

# 处理事件

如果要让行为响应对应组件的事件触发， 就应覆写 [yii\base\Behavior::events()](https://www.yiichina.com/doc/api/2.0/yii-base-behavior" \l "events()-detail) 方法，如：



[events()](https://www.yiichina.com/doc/api/2.0/yii-base-behavior" \l "events()-detail) 方法返回事件列表和相应的处理器。

上例声明了 [EVENT\_BEFORE\_VALIDATE](https://www.yiichina.com/doc/api/2.0/yii-base-model" \l "EVENT_BEFORE_VALIDATE-detail) 事件和它的处理器 beforeValidate() 。 当指定一个事件处理器时，要使用以下格式之一：

* 指向行为类的方法名的字符串，如上例所示；
* 对象或类名和方法名的数组，如 [$object, 'methodName']；
* 匿名方法。

# 附加行为

可以静态或动态地附加行为到[组件](https://www.yiichina.com/doc/api/2.0/yii-base-component)。前者在实践中更常见。

要静态附加行为，覆写行为要附加的组件类的 [behaviors()](https://www.yiichina.com/doc/api/2.0/yii-base-component" \l "behaviors()-detail) 方法即可。 [behaviors()](https://www.yiichina.com/doc/api/2.0/yii-base-component" \l "behaviors()-detail) 方法应该返回行为[配置](https://www.yiichina.com/doc/guide/2.0/basic-configs)列表。 每个行为配置可以是行为类名也可以是配置数组。如：

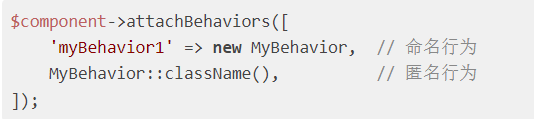


通过指定行为配置数组相应的键可以给行为关联一个名称。这种行为称为****命名行为****。 上例中，有两个命名行为：myBehavior2 和 myBehavior4 。如果行为没有指定名称就是****匿名行为****。

要动态附加行为，在对应组件里调用 [yii\base\Component::attachBehavior()](https://www.yiichina.com/doc/api/2.0/yii-base-component" \l "attachBehavior()-detail) 方法即可，如：



可以通过 [yii\base\Component::attachBehaviors()](https://www.yiichina.com/doc/api/2.0/yii-base-component" \l "attachBehaviors()-detail) 方法一次附加多个行为：



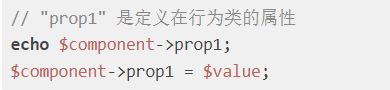
还可以通过[配置](https://www.yiichina.com/doc/guide/2.0/concept-configurations)去附加行为：



# 使用行为

使用行为，必须像前文描述的一样先把它附加到 [component](https://www.yiichina.com/doc/api/2.0/yii-base-component) 类或其子类。一旦行为附加到组件，就可以直接使用它。

行为附加到组件后，可以通过组件访问一个行为的****公共****成员变量 或 getter 和 setter 方法定义的[属性](https://www.yiichina.com/doc/guide/2.0/concept-properties)：



类似地也可以调用行为的****公共****方法：



如你所见，尽管 $component 未定义 prop1 和 foo() ， 它们用起来也像组件自己定义的一样。

如果两个行为都定义了一样的属性或方法，并且它们都附加到同一个组件， 那么****首先****附加上的行为在属性或方法被访问时有优先权。

附加行为到组件时的命名行为，可以使用这个名称来访问行为对象， 如下所示：

$behavior = $component->getBehavior('myBehavior');

也能获取附加到这个组件的所有行为：

$behaviors = $component->getBehaviors();

# 移除行为

要移除行为，可以调用 [yii\base\Component::detachBehavior()](https://www.yiichina.com/doc/api/2.0/yii-base-component" \l "detachBehavior()-detail) 方法用行为相关联的名字实现：

$component->detachBehavior('myBehavior1');

也可以移除全部行为：

$component->detachBehaviors();

# 使用 TimestampBehavior

最后以 [yii\behaviors\TimestampBehavior](https://www.yiichina.com/doc/api/2.0/yii-behaviors-timestampbehavior) 的讲解来结尾， 这个行为支持在 [Active Record](https://www.yiichina.com/doc/api/2.0/yii-db-activerecord) 存储时自动更新它的时间戳属性。

首先，附加这个行为到计划使用该行为的 [Active Record](https://www.yiichina.com/doc/api/2.0/yii-db-activerecord) 类：



# 其它行为

有几种内置和外部行为可用：

* [yii\behaviors\BlameableBehavior](https://www.yiichina.com/doc/api/2.0/yii-behaviors-blameablebehavior) - 使用当前用户 ID 自动填充指定的属性。
* [yii\behaviors\SluggableBehavior](https://www.yiichina.com/doc/api/2.0/yii-behaviors-sluggablebehavior) - 自动填充指定的属性，其值可以在 URL 中用作 slug。
* [yii\behaviors\AttributeBehavior](https://www.yiichina.com/doc/api/2.0/yii-behaviors-attributebehavior) - 在发生特定事件时自动为 ActiveRecord 对象的一个或多个属性 指定一个指定的值。
* [yii2tech\ar\softdelete\SoftDeleteBehavior](https://github.com/yii2tech/ar-softdelete) - 提供软删除和软恢复 ActiveRecord 的 方法。即将记录标记为已删除的设置标记或状态。
* [yii2tech\ar\position\PositionBehavior](https://github.com/yii2tech/ar-position) - 允许通过提供重新排序方法来 管理整数字段中的记录顺序。

# 比较行为与 Traits

虽然行为类似于 [traits](http://www.php.net/traits)，它们都将自己的属性和方法“注入”到主类中， 但它们在许多方面有所不同。如下所述，他们都有优点和缺点。 它们更像互补类而非替代类。

## 使用行为的原因

行为类像普通类支持继承。另一方面，traits 可以视为 PHP 语言支持的复制粘贴功能， 它不支持继承。

行为无须修改组件类就可动态附加到组件或移除。 要使用 traits，必须修改使用它的类。

行为是可配置的，而 traits 则不可行。

行为可以通过响应事件来定制组件的代码执行。

当附属于同一组件的不同行为之间可能存在名称冲突时， 通过优先考虑附加到该组件的行为， 自动解决冲突。由不同 traits 引起的名称冲突需要通过 重命名受影响的属性或方法进行手动解决。

## 使用 Traits 的原因

Traits 比行为更有效，因为行为是既需要时间又需要内存的对象。

因为 IDE 是一种本地语言结构，所以它们对 Traits 更友好。