# 简介

事件可以将自定义代码“注入”到现有代码中的特定执行点。

附加自定义代码到某个事件，当这个事件被触发时，这些代码就会自动执行。

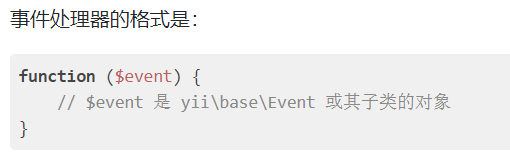
例如，邮件程序对象成功发出消息时可触发 messageSent 事件。 如想追踪成功发送的消息，可以附加相应追踪代码到 messageSent 事件。

Yii 引入了名为 [yii\base\Component](https://www.yiichina.com/doc/api/2.0/yii-base-component) 的基类以支持事件。 如果一个类需要触发事件就应该继承 [yii\base\Component](https://www.yiichina.com/doc/api/2.0/yii-base-component) 或其子类。

# 事件处理器（Event Handlers）

事件处理器是一个[PHP 回调函数](http://www.php.net/manual/en/language.types.callable.php)， 当它所附加到的事件被触发时它就会执行。可以使用以下回调函数之一：

* 字符串形式指定的 PHP 全局函数，如 'trim' ；
* 对象名和方法名数组形式指定的对象方法，如 [$object, $method] ；
* 类名和方法名数组形式指定的静态类方法，如 [$class, $method] ；
* 匿名函数，如 function ($event) { ... } 。



通过 $event 参数，事件处理器就获得了以下有关事件的信息：

* [event name](https://www.yiichina.com/doc/api/2.0/yii-base-event" \l "$name-detail)：事件名
* [event sender](https://www.yiichina.com/doc/api/2.0/yii-base-event" \l "$sender-detail)：调用 trigger() 方法的对象
* [custom data](https://www.yiichina.com/doc/api/2.0/yii-base-event" \l "$data-detail)：附加事件处理器时传入的数据，默认为空，后文详述

# 附加事件处理器（Attaching Event Handlers）

调用 [yii\base\Component::on()](https://www.yiichina.com/doc/api/2.0/yii-base-component" \l "on()-detail) 方法来附加处理器到事件上。如：



你也可以通过 [配置](https://www.yiichina.com/doc/guide/2.0/concept-configurations) 附加事件处理器。 请 参考 [配置的格式](https://www.yiichina.com/doc/guide/2.0/concept-configurations" \l "configuration-format) 小节了解更多.

附加事件处理器时可以提供额外数据作为 [yii\base\Component::on()](https://www.yiichina.com/doc/api/2.0/yii-base-component" \l "on()-detail) 方法的第三个参数。 数据在事件被触发和处理器被调用时能被处理器使用。如：

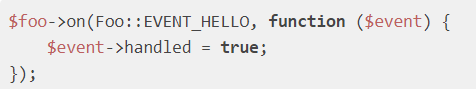


# 事件处理器顺序（Event Handler Order）

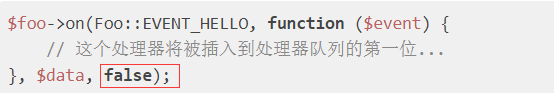
可以附加一个或多个处理器到一个事件。

当事件被触发，已附加的处理器将按附加次序依次调用。

如果某个处理器需要停止其后的处理器调用，可以设置 $event 参数的 [yii\base\Event::$handled](https://www.yiichina.com/doc/api/2.0/yii-base-event" \l "$handled-detail) 属性为真， 如下：



默认新附加的事件处理器排在已存在处理器队列的最后。 因此，这个处理器将在事件被触发时最后一个调用。 在处理器队列最前面插入新处理器将使该处理器最先调用，可以传递第四个参数 $append 为假并调用 [yii\base\Component::on()](https://www.yiichina.com/doc/api/2.0/yii-base-component" \l "on()-detail) 方法实现：



# 触发事件（Triggering Events）

事件通过调用 [yii\base\Component::trigger()](https://www.yiichina.com/doc/api/2.0/yii-base-component" \l "trigger()-detail) 方法触发，此方法须传递事件名， 还可以传递一个事件对象，用来传递参数到事件处理器。如：



有时想要在触发事件时同时传递一些额外信息到事件处理器。

例如，邮件程序要传递消息信息到 messageSent 事件的处理器以便处理器了解哪些消息被发送了。

 为此，可以提供一个事件对象作为 [yii\base\Component::trigger()](https://www.yiichina.com/doc/api/2.0/yii-base-component" \l "trigger()-detail) 方法的第二个参数。 这个事件对象必须是 [yii\base\Event](https://www.yiichina.com/doc/api/2.0/yii-base-event) 类或其子类的实例。 如：



# 移除事件处理器（Detaching Event Handlers）

从事件移除处理器，调用 [yii\base\Component::off()](https://www.yiichina.com/doc/api/2.0/yii-base-component" \l "off()-detail) 方法。如：



注意当匿名函数附加到事件后一般不要尝试移除匿名函数， 除非你在某处存储了它。以上示例中， 假设匿名函数存储为变量 $anonymousFunction 。

移除事件的全部处理器，简单调用 [yii\base\Component::off()](https://www.yiichina.com/doc/api/2.0/yii-base-component" \l "off()-detail) 即可，不需要第二个参数：

$foo->off(Foo::EVENT\_HELLO);

# 类级别的事件处理器（Class-Level Event Handlers）

以上部分，我们叙述了在实例级别如何附加处理器到事件。 有时想要一个类的所有实例而不是一个指定的实例都响应一个被触发的事件， 并不是一个个附加事件处理器到每个实例， 而是通过调用静态方法 [yii\base\Event::on()](https://www.yiichina.com/doc/api/2.0/yii-base-event" \l "on()-detail) 在类级别附加处理器。



每当 [ActiveRecord](https://www.yiichina.com/doc/api/2.0/yii-db-baseactiverecord) 或其子类的实例触发 [EVENT\_AFTER\_INSERT](https://www.yiichina.com/doc/api/2.0/yii-db-baseactiverecord" \l "EVENT_AFTER_INSERT-detail) 事件时， 这个事件处理器都会执行。在这个处理器中，可以通过 $event->sender 获取触发事件的对象。

当对象触发事件时，它首先调用实例级别的处理器，然后才会调用类级别处理器。

可调用静态方法[yii\base\Event::trigger()](https://www.yiichina.com/doc/api/2.0/yii-base-event" \l "trigger()-detail)来触发一个类级别事件。 类级别事件不与特定对象相关联。因此，它只会引起类级别事件处理器的调用。 如：



****注意：****因为类级别的处理器响应类和其子类的所有实例触发的事件， 必须谨慎使用，尤其是底层的基类，如 yii\base\Object。

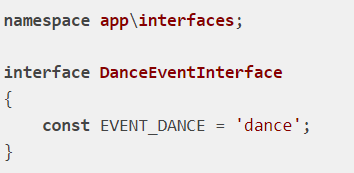
移除类级别的事件处理器只需调用[yii\base\Event::off()](https://www.yiichina.com/doc/api/2.0/yii-base-event" \l "off()-detail)，如：



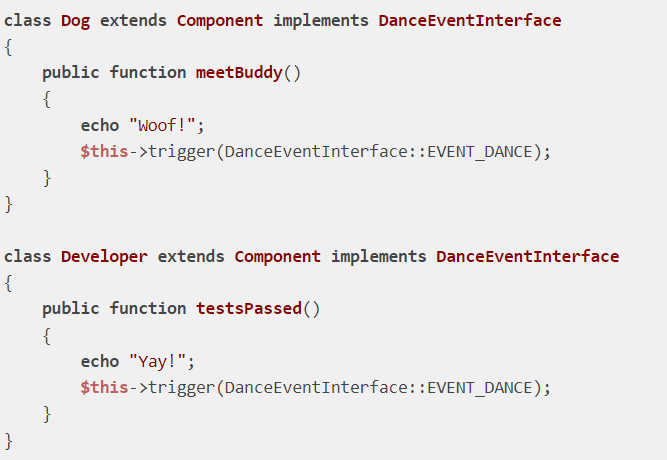
# 使用接口事件（Events using interfaces）

有更多的抽象方式来处理事件。你可以为特殊的事件创建一个独立的接口， 然后在你需要的类中实现它。

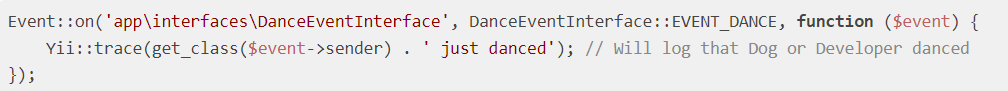
例如，我们可以先创建下面这个接口:



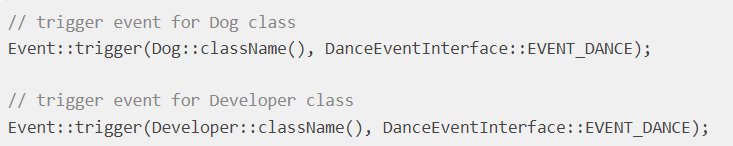
然后在两个类中实现:



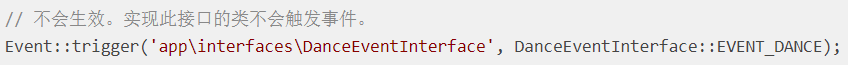
要处理由这些类触发的 EVENT\_DANCE ，调用 [Event::on()](https://www.yiichina.com/doc/api/2.0/yii-base-event" \l "on()-detail) 并将接口类名作为第一个参数:



你可以在这些类中触发这个事件：



但是请注意, 你不能让所有实现这个接口的类都触发事件：



调用 [Event::off()](https://www.yiichina.com/doc/api/2.0/yii-base-event" \l "off()-detail) 移除事件处理器。例如：



# 全局事件（Global Events）

所谓全局事件实际上是一个基于以上叙述的事件机制的戏法。

它需要一个全局可访问的单例， 如[应用](https://www.yiichina.com/doc/guide/2.0/structure-applications)实例。

事件触发者不调用其自身的 trigger() 方法，而是调用单例的 trigger() 方法来触发全局事件。 类似地，事件处理器被附加到单例的事件。如：



全局事件的一个好处是当附加处理器到一个对象要触发的事件时， 不需要产生该对象。相反，处理器附加和事件触发都通过单例 （如应用实例）完成。

然而，因为全局事件的命名空间由各方共享，应合理命名全局事件， 如引入一些命名空间（例："frontend.mail.sent", "backend.mail.sent"）

# 通配符事件（Wildcard Events）

自 2.0.14 以来，您可以为多个匹配通配符模式的事件设置事件处理程序。 例如：



通配符模式也可以用于类级别的事件。 例如：



这允许您使用以下代码通过单个处理程序捕获所有应用程序事件：



****注意：****事件处理程序设置的使用通配符可能会降低应用程序的性能。 如果可能，最好避免。

为了移除由通配符模式指定的事件处理程序，您应该在 [yii\base\Component::off()](https://www.yiichina.com/doc/api/2.0/yii-base-component" \l "off()-detail) 或 [yii\base\Event::off()](https://www.yiichina.com/doc/api/2.0/yii-base-event" \l "off()-detail) 调用中重复相同的模式。 请记住，在移除事件处理程序期间传递通配符将移除为此通配符指定的处理程序， 而为常规事件名称附加的处理程序将保留，即使它们与模式匹配。 例如：

