# 简介

Yii提供了一个强大的日志框架，这个框架具有高度的可定制性和可扩展性。

使用这个框架， 你可以轻松地记录各种类型的消息，过滤它们， 并且将它们收集到不同的目标，诸如文件，数据库，邮件。

使用Yii日志框架涉及下面的几个步骤：

* 在你代码里的各个地方记录 [log messages](https://www.yiichina.com/doc/guide/2.0/runtime-logging" \l "log-messages)；
* 在应用配置里通过配置 [log targets](https://www.yiichina.com/doc/guide/2.0/runtime-logging" \l "log-targets) 来过滤和导出日志消息；
* 检查由不同的目标导出的已过滤的日志消息（例如：[Yii debugger](https://www.yiichina.com/doc/guide/2.0/tool-debugger)）。

在这部分，我们主要描述前两个步骤。

# 日志消息

记录日志消息就跟调用下面的日志方法一样简单：

* [Yii::trace()](https://www.yiichina.com/doc/api/2.0/yii-baseyii" \l "trace()-detail)：记录一条消息去跟踪一段代码是怎样运行的。这主要在开发的时候使用。
* [Yii::info()](https://www.yiichina.com/doc/api/2.0/yii-baseyii" \l "info()-detail)：记录一条消息来传达一些有用的信息。
* [Yii::warning()](https://www.yiichina.com/doc/api/2.0/yii-baseyii" \l "warning()-detail)：记录一个警告消息用来指示一些已经发生的意外。
* [Yii::error()](https://www.yiichina.com/doc/api/2.0/yii-baseyii" \l "error()-detail)：记录一个致命的错误，这个错误应该尽快被检查。

这些日志记录方法针对 严重程度 和 类别 来记录日志消息。

它们共享相同的函数签名 function ($message, $category = 'application')，$message代表要被 记录的日志消息，而 $category 是日志消息的类别。

在下面的示例代码中，在默认的类别 application 下 记录了一条跟踪消息：

Yii::trace('start calculating average revenue');

为了更好地组织和过滤日志消息，我们建议您为每个日志消息指定一个适当的类别。

您可以为类别选择一个分层命名方案， 这将使得 [log targets](https://www.yiichina.com/doc/guide/2.0/runtime-logging" \l "log-targets) 在基于它们的分类来过滤消息变得更加容易。

一个简单而高效的命名方案是使用PHP魔术常量 \_\_METHOD\_\_ 作为分类名称。 这种方式也在Yii框架的核心代码中得到应用， 例如，

Yii::trace('start calculating average revenue', **\_\_METHOD\_\_**);

# 日志目标

一个日志目标是一个 [yii\log\Target](https://www.yiichina.com/doc/api/2.0/yii-log-target) 类或者它的子类的实例。

它将通过他们的严重层级和类别来过滤日志消息，然后将它们导出到一些媒介中。

例如，一个 [database target](https://www.yiichina.com/doc/api/2.0/yii-log-dbtarget) 目标导出已经过滤的日志消息到一个数据的表里面， 而一个 [email target](https://www.yiichina.com/doc/api/2.0/yii-log-emailtarget)目标将日志消息导出到指定的邮箱地址里。

在一个应用里，通过配置在应用配置里的 log [application component](https://www.yiichina.com/doc/guide/2.0/structure-application-components)，你可以注册多个日志目标。 就像下面这样：



在上面的代码中，在 [yii\log\Dispatcher::$targets](https://www.yiichina.com/doc/api/2.0/yii-log-dispatcher" \l "$targets-detail) 属性里有两个日志目标被注册：

* 第一个目标选择的是错误和警告层级的消息，并且在数据库表里保存他们；
* 第二个目标选择的是错误层级的消息并且是在以 yii\db\ 开头的分类下，并且在一个邮件里将它们发送到 admin@example.com 和 developer@example.com。

Yii配备了以下的内建日志目标。请参考关于这些类的API文档， 并且学习怎样配置和使用他们。

* [yii\log\DbTarget](https://www.yiichina.com/doc/api/2.0/yii-log-dbtarget)：在数据库表里存储日志消息。
* [yii\log\EmailTarget](https://www.yiichina.com/doc/api/2.0/yii-log-emailtarget)：发送日志消息到预先指定的邮箱地址。
* [yii\log\FileTarget](https://www.yiichina.com/doc/api/2.0/yii-log-filetarget)：保存日志消息到文件中.
* [yii\log\SyslogTarget](https://www.yiichina.com/doc/api/2.0/yii-log-syslogtarget)：通过调用PHP函数 syslog() 将日志消息保存到系统日志里。

## 消息过滤

对于每一个日志目标，你可以配置它的 [levels](https://www.yiichina.com/doc/api/2.0/yii-log-target" \l "$levels-detail) 和 [categories](https://www.yiichina.com/doc/api/2.0/yii-log-target" \l "$categories-detail) 属性来指定哪个消息的严重程度和分类目标应该处理。

[levels](https://www.yiichina.com/doc/api/2.0/yii-log-target" \l "$levels-detail) 属性是由一个或者若干个以下值组成的数组：

* error：相应的消息通过 [Yii::error()](https://www.yiichina.com/doc/api/2.0/yii-baseyii" \l "error()-detail) 被记录。
* warning：相应的消息通过 [Yii::warning()](https://www.yiichina.com/doc/api/2.0/yii-baseyii" \l "warning()-detail) 被记录。
* info：相应的消息通过 [Yii::info()](https://www.yiichina.com/doc/api/2.0/yii-baseyii" \l "info()-detail) 被记录。
* trace：相应的消息通过 [Yii::trace()](https://www.yiichina.com/doc/api/2.0/yii-baseyii" \l "trace()-detail) 被记录。
* profile：相应的消息通过 [Yii::beginProfile()](https://www.yiichina.com/doc/api/2.0/yii-baseyii" \l "beginProfile()-detail) 和 [Yii::endProfile()](https://www.yiichina.com/doc/api/2.0/yii-baseyii" \l "endProfile()-detail) 被记录。更多细节将在 [Profiling](https://www.yiichina.com/doc/guide/2.0/runtime-logging" \l "performance-profiling) 分段解释。

如果你没有指定 [levels](https://www.yiichina.com/doc/api/2.0/yii-log-target" \l "$levels-detail) 的属性， 那就意味着目标将处理 任何 严重程度的消息。

[categories](https://www.yiichina.com/doc/api/2.0/yii-log-target" \l "$categories-detail) 属性是一个包含消息分类名称或者模式的数组。 一个目标将只处理那些在这个数组中能够找到对应的分类或者其中一个相匹配的模式的消息。 一个分类模式是一个以星号 \* 结尾的分类名前缀。假如一个分类名与分类模式具有相同的前缀， 那么该分类名将和分类模式相匹配。例如， yii\db\Command::execute 和 yii\db\Command::query 都是作为分类名称运用在 [yii\db\Command](https://www.yiichina.com/doc/api/2.0/yii-db-command) 类来记录日志消息的。 它们都是匹配模式 yii\db\\*。

假如你没有指定 [categories](https://www.yiichina.com/doc/api/2.0/yii-log-target" \l "$categories-detail) 属性， 这意味着目标将会处理 任何 分类的消息。

除了通过 [categories](https://www.yiichina.com/doc/api/2.0/yii-log-target" \l "$categories-detail) 属性设置白名单分类，你也可以通过 [except](https://www.yiichina.com/doc/api/2.0/yii-log-target" \l "$except-detail) 属性来设置某些分类作为黑名单。假如一条消息的分类在这个属性中被发现或者是匹配其中一个， 那么它将不会在目标中被处理。

## 消息格式化

日志目标以某种格式导出过滤过的日志消息。

例如， 假如你安装一个 [yii\log\FileTarget](https://www.yiichina.com/doc/api/2.0/yii-log-filetarget) 类的日志目标， 你应该能找出一个日志消息类似下面的 runtime/log/app.log 文件：

2014-10-04 18:10:15 [::1][][-][trace][yii\base\Module::getModule] Loading module: debug

默认情况下，日志消息将被格式化，格式化的方式遵循 [yii\log\Target::formatMessage()](https://www.yiichina.com/doc/api/2.0/yii-log-target" \l "formatMessage()-detail)：

Timestamp [IP address][User ID][Session ID][Severity Level][Category] Message Text

你可以通过配置 [yii\log\Target::$prefix](https://www.yiichina.com/doc/api/2.0/yii-log-target" \l "$prefix-detail) 的属性来自定义格式，这个属性是一个 PHP 可调用体返回的自定义消息前缀。

例如，下面的代码配置了一个日志目标的前缀是每个日志消息中当前用户的 ID（IP 地址和 Session ID 被删除是由于隐私的原因）。



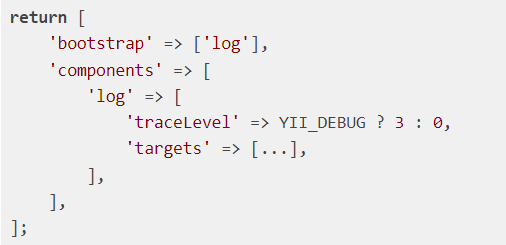
除了消息前缀以外，日志目标也可以追加一些上下文信息到每组日志消息中。 默认情况下，这些全局的PHP变量的值被包含在：$\_GET，$\_POST，$\_FILES，$\_COOKIE，$\_SESSION 和 $\_SERVER 中。 你可以通过配置 [yii\log\Target::$logVars](https://www.yiichina.com/doc/api/2.0/yii-log-target" \l "$logVars-detail) 属性适应这个行为， 这个属性是你想要通过日志目标包含的全局变量名称。 举个例子，下面的日志目标配置指明了只有 $\_SERVER 变量的值将被追加到日志消息中。



你可以将 logVars 配置成一个空数组来完全禁止上下文信息包含。 或者假如你想要实现你自己提供上下文信息的方式， 你可以重写 [yii\log\Target::getContextMessage()](https://www.yiichina.com/doc/api/2.0/yii-log-target" \l "getContextMessage()-detail) 方法。

## 消息跟踪级别

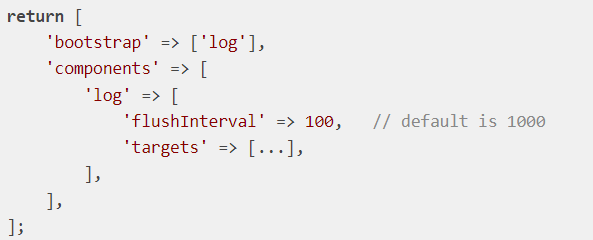
在开发的时候，通常希望看到每个日志消息来自哪里。这个是能够被实现的，通过配置 log 组件的 [traceLevel](https://www.yiichina.com/doc/api/2.0/yii-log-dispatcher" \l "$traceLevel-detail) 属性， 就像下面这样：



上面的应用配置设置了 [traceLevel](https://www.yiichina.com/doc/api/2.0/yii-log-dispatcher" \l "$traceLevel-detail) 的层级，假如 YII\_DEBUG 开启则是3，否则是0。 这意味着，假如 YII\_DEBUG 开启，每个日志消息在日志消息被记录的时候， 将被追加最多3个调用堆栈层级；假如 YII\_DEBUG 关闭， 那么将没有调用堆栈信息被包含。

## 消息刷新和导出

如上所述，通过 [logger object](https://www.yiichina.com/doc/api/2.0/yii-log-logger) 对象，日志消息被保存在一个数组里。 为了这个数组的内存消耗，当数组积累了一定数量的日志消息， 日志对象每次都将刷新被记录的消息到 [log targets](https://www.yiichina.com/doc/guide/2.0/runtime-logging" \l "log-targets) 中。 你可以通过配置 log 组件的 [flushInterval](https://www.yiichina.com/doc/api/2.0/yii-log-dispatcher" \l "$flushInterval-detail) 属性来自定义数量：



一般不用管，一次请求积累的日志不会太大。用不着中途刷新日志。

当 [logger object](https://www.yiichina.com/doc/api/2.0/yii-log-logger) 对象刷新日志消息到 [log targets](https://www.yiichina.com/doc/guide/2.0/runtime-logging" \l "log-targets) 的时候，它们并 不能立即获取导出的消息。相反，消息导出仅仅在一个日志目标累积了一定数量的过滤消息的时候才会发生。

你可以通过配置 个别的 [log targets](https://www.yiichina.com/doc/guide/2.0/runtime-logging" \l "log-targets) 的 [exportInterval](https://www.yiichina.com/doc/api/2.0/yii-log-target" \l "$exportInterval-detail) 属性来 自定义这个数量，就像下面这样：



因为刷新和导出层级的设置，默认情况下，当你调用 Yii::trace() 或者任何其他的记录方法，你将不能在日志目标中立即看到日志消息。

这对于一些长期运行的控制台应用来说可能是一个问题。为了让每个日志消息在日志目标中能够立即出现， 你应该设置 [flushInterval](https://www.yiichina.com/doc/api/2.0/yii-log-dispatcher" \l "$flushInterval-detail) 和 [exportInterval](https://www.yiichina.com/doc/api/2.0/yii-log-target" \l "$exportInterval-detail) 都为1， 就像下面这样：



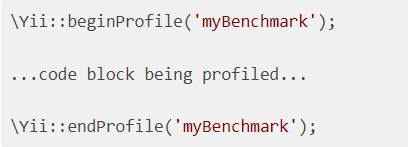
****注意：****频繁的消息刷新和导出将降低你的应用性能。

## 性能分析

性能分析是一个特殊的消息记录类型，它通常用在测量某段代码块的时间， 并且找出性能瓶颈是什么。

举个例子，[yii\db\Command](https://www.yiichina.com/doc/api/2.0/yii-db-command) 类 使用性能分析找出每个数据库查询的时间。

为了使用性能分析，首先确定需要进行分析的代码块。 然后像下面这样围住每个代码块：



这里的 myBenchmark 代表一个唯一标记来标识一个代码块。之后当你检查分析结果的时候， 你将使用这个标记来定位对应的代码块所花费的时间。

对于每个被分析的代码块，一个带有严重程度为 profile 的日志消息将被记录。 你可以配置一个 [log target](https://www.yiichina.com/doc/guide/2.0/runtime-logging" \l "log-targets) 去收集这些 消息，并且导出他们。[Yii debugger](https://www.yiichina.com/doc/guide/2.0/tool-debugger) 有一个内建的性能分析面板能够展示分析结果。