# 上下文

上下文作为整个框架中最重要的内容之一，请务必理解和完整地阅读此部分！

一个上下文描述了一个事件和所关联的对象的环境。例如：你在处理 HTTP 请求的 @RequestMapping 绑定的事件中，你需要获取请求的 HTTP 头和 Cookie，再比如你在处理 QQ 机器人发来的命令 @CQCommand("随机数") 的时候，在这个方法内，你需要获取发来的人的 QQ 号码。以上我们将处理以上运行环境的对象叫做上下文。

由于 Swoole 的协程加持，我们利用了协程 ID 绑定对象来进行构造上下文。

以默认的机器人收发消息为例，通过对默认模块的了解，我们可以知道，在绑定 @CQCommand 等类似事件后，你可以用上下文获取发来这条消息的人的 QQ 号码：

/\*\*  
 \* @CQCommand("你好")  
 \*/  
public function hello() {  
 $user\_id = context()->getUserId();  
 context()->reply("你好啊，".$user\_id."，很高兴认识你！");  
}

context() 就是获取上下文对象的全局函数，它还有简写：ctx()。

当然，上下文中的方法不是每个都能在任何时候使用的。例如 getUserId() 你不能在 @RequestMapping 注解的函数中使用，因为它不是机器人消息的上下文。下面说明上下文对象的方法中，每个都会说明每个方法可以在哪些事件中使用：

## getServer()

获取 Swoole WebSocker Server 对象。此对象是 Swoole 的对象，详情见 [Swoole 文档](https://wiki.swoole.com/#/websocket_server)。

可以使用的事件：@SwooleEventAt("message")，@SwooleEventAt("open")，@SwooleEventAt("close")，@OnStart() 以及所有 HTTP API 发来的事件：@CQCommand()，@CQMessage() 等。

## getFrame()

获取 \Swoole\Websocket\Frame 对象，此对象是 Swoole 的对象，详情见 [Swoole 文档](https://wiki.swoole.com/#/websocket_server?id=swoolewebsocketframe)。

可以使用的事件：@SwooleEventAt("message") 以及所有 HTTP API 发来的事件：@CQCommand()，@CQMessage() 等，

## getFd()

获取当前连入 Swoole 服务器的连接 fd 号。返回 int。

可以使用的事件：所有 **getFrame()** 可以使用的，@SwooleEventAt("open")，@SwooleEventAt("close")

## getData()

返回 CQHTTP 事件上报的原始数据包，已经被解析成数组，可以直接操作。

可以使用的事件：所有 HTTP API 发来的事件：@CQCommand()，@CQMessage() 等。

/\*\*  
 \* @CQMessage()  
 \*/  
public function onMessage() {  
 $data = ctx()->getData();  
 Console::info("消息类型是：" . $data["message\_type"]);  
}

## getRequest()

返回 \Swoole\Http\Request 对象，可在 @RequestMapping 中使用，获取 Cookie，请求头，GET 参数什么的。[Swoole 文档](https://wiki.swoole.com/#/http_server?id=httprequest)。

可以使用的事件：@RequestMapping()，@SwooleEventAt("request")，@SwooleEventAt("open")。

## getResponse()

返回 \Swoole\Http\Response 对象的增强版，可在 HTTP 请求相关的事件中使用，返回内容和设置 Cookie 什么的。[Swoole 文档](https://wiki.swoole.com/#/http_server?id=httpresponse)。

可以使用的事件：@RequestMapping()，@SwooleEventAt("request")。

下面是使用以上两个功能的组合示例：

/\*\*  
 \* @RequestMapping("/ping")  
 \*/  
public function ping() {  
 $name = ctx()->getRequest()->get["name"] ?? "unknown";  
 ctx()->getResponse()->end("Hello ".$name."!");  
}

## getConnection()

返回此上下文相关联的 WSConnection 对象。此对象详情在 [这里](/guide/websocket-server.html#连接对象)。

可以使用的事件：所有 **getFrame()** 可以使用的。

## getCid()

返回当前上下文所绑定的协程 ID，不出意外的话此 ID 和 \Co::getCid() 返回值一样。

## getRobot()

返回当前上下文关联的机器人 API 调用对象 [ZMRobot](/guide/component/zmrobot.html)。

可以使用的事件：所有 HTTP API 发来的事件：@CQCommand()，@CQMessage() 等。

ctx()->getRobot()->sendPrivateMsg(123456, "发送私聊消息");

## getMessage()

获取 data 数据中的 message 消息。

可以使用的事件：@CQCommand()，@CQMessage，@CQBefore("message")，@CQAfter("message")

## getUserId()

获取发消息的用户的 QQ 号码。

可以使用的事件：所有 **含有** user\_id 上报参数的 CQHTTP 事件。[CQHTTP - 事件上报](https://cqhttp.cc/docs/#/Post)。

## getGroupId()

获取发消息来自的 QQ 群号。

可以使用的事件：所有含有 group\_id 上报参数的 CQHTTP 事件。[CQHTTP - 事件上报](https://cqhttp.cc/docs/#/Post)。

## getDiscussId()

获取发消息来自的 QQ 讨论组 ID 号。

可以使用的事件：所有含有 discuss\_id 上报参数的 CQHTTP 事件。[CQHTTP - 事件上报](https://cqhttp.cc/docs/#/Post)。

## getMessageType()

获取消息类型，同参数 message\_type。

可以使用的事件：所有 post\_type 为 message 的响应事件，如 @CQMessage，@CQCommand。

## getRobotId()

获取事件上报的机器人自己的 QQ 号码。

可以使用的事件：所有 HTTP API 发来的事件：@CQCommand()，@CQNotice() 等。

## setMessage()

与 getMessage() 对应，用于更改上下文中保存的事件信息，可以用于消息变更和过滤。

## setUserId()

与上同理，更改 user\_id。

## setGroupId()

与上同理。

## setDiscussId()

与上同理。

## setMessageType()

与上同理，修改消息类型。

## setData()

与上同理，与 getData() 对应，用于更改上下文中的 data。

## getCache()

获取保存在上下文中的临时缓存变量。当相关联的事件结束后，数据会从内存中被释放。用于同一事件的多个函数中的信息传递。

* 参数：$key，缓存变量的键名
* 返回：mixed，存入缓存的变量值。

$a = ctx()->getCache("block\_continue");  
// 如果变量不存在，则返回 null

## setCache()

与 getCache() 对应，是设置内容的。

ctx()->setCache("abc", "asdasd");  
$result = ctx()->getCache("abc"); // asdasd

## reply()

快速回复当前用户消息内容。

* 参数1：$msg，字符串，你要回复的消息内容
* 参数2：$yield = false，可选，当为 true 时，会协程等待后返回 **消息回复** 的结果，包括 API 状态码、消息 message\_id 等。

$r = ctx()->reply("我又好了。", true);  
if($r["retcode"] == 0) Console::success("消息发送成功！");

## finalReply()

快速回复用户消息，并阻止其他模块接下来继续处理此事件。

参数同 reply()。

## waitMessage()

* 参数：waitMessage($prompt = "", $timeout = 600, $timeout\_prompt = "")
* 用途：等待用户输入消息

$prompt 参数为回复用户的文本内容，$timeout 是等待用户回复的超时时间(秒)，$timeout\_prompt 是超时后回复用户的文本。

用法示例：

/\*\*  
 \* @CQCommand("自我介绍")  
 \*/  
function yourName(){  
 $r = context()->waitMessage("你叫啥名字呀？", 600, "你都10分钟不理我了，嘤嘤嘤");  
 context()->finalReply("好的，可爱的机器人记住你叫 ".$r." 啦！以后多聊天哦！");  
}

效果就是，给机器人发送 自我介绍 后，它问你 你叫啥名字呀？，如果你在 10 分钟内回复了机器人，则会回你下面一条句子，如果 10 分钟还没有回复，则机器人会回复你不理你的句子。

## getArgs()

* 参数：getArgs(&$arg, $mode, $prompt\_msg)
* 用途：是 waitMessage() 的封装，常用于配合 @CQCommand 使用，在 @CQCommand 介绍中就用了此函数，可以看其中的例子。

$arg 为命令传递进入的参数引用。

$mode 支持：

* ZM\_MATCH\_ALL：获取除命令本身外的所有后续内容，例如 疫情 iafwebfuaw 乏味，match 后就会匹配到 iafwebfuaw 乏味。如果触发命令只有参数本身的时候，则调用 waitMessage 等待用户输入参数。
* ZM\_MATCH\_NUMBER：获取后续参数中第一个数字参数，例如 疫情 北京 123，match 后就会获取到 123。如果触发命令没有数字的时候，则调用 waitMessage 等待用户输入参数。
* ZM\_MATCH\_FIRST：获取命令后的第一个参数，例如 疫情 北京 123，match 后就会获取到 北京。如果触发命令只有参数本身的时候，则调用 waitMessage 等待用户输入参数。

$prompt\_msg：当遇到等待的时候，向用户发送输入的提醒消息内容，同 waitMesssage 的 $prompt。

## copy()

返回此上下文的所有对象和变量，以数组的形式。

* 返回值：可自行 var\_dump 查看。

## 模块阻断

模块阻断指的是，一个事件可能会依次触发多个绑定的方法，如果在执行到前面的方法，不想让后面的方法继续执行，你可以使用这个万金油阻断接下来的其他模块中的函数运行：

context()->setCache("block\_continue", true);

比如你在设置了多个 @CQMessage()，如果第一个代码里面，执行了模块阻断，则接下来的其他 CQMessage 都不会执行到。

/\*\*  
 \* @CQMessage(level=25)  
 \*/  
public function msg1() {  
 context()->setCache("block\_continue", true); //设置阻断  
}  
/\*\*  
 \* @CQMessage(level=20)  
 \*/  
public function msg2() {  
 Console::info(ctx()->getMessage()); // 这个永远不会被执行到  
}