# 模块类修饰注解

模块类修饰注解是指对你编写的模块的 class 进行“打标签”，也就是添加注解，从而实现一些特殊的功能。

## Closed()

这个注解类是用于停止解析此类中的所有方法的注解用的，对模块的类添加了此注解后，此类下的所有函数的所有注解和事件均不解析。当有模块或功能需要暂时关闭时会用到此注解。

* 名称：@Closed
* 命名空间：ZM\Annotation\Module\Closed

use ZM\Annotation\Module\Closed;  
/\*\*  
 \* @Closed()  
 \*/  
class ModuleA {  
 /\*\*  
 \* @CQCommand("你好")  
 \*/  
 public function hello() { return "你好啊！我是可爱的机器人"; }  
}

上面的例子中，在以上对类 ModuleA 进行修饰后，向机器人发送 你好，机器人不会回复。

## SaveBuffer()

* 名称：@SaveBuffer
* 命名空间：ZM\Annotation\Module\SaveBuffer
* 参数：buf\_name，sub\_folder
* 作用：添加一个自动保存的缓存变量，供模块内各个功能使用，在框架启动后触发。关于缓存类，见 [缓存类](/guide/component/zmbuf.html)

此注解是用于简化对内存缓存的持久化储存，本质就是调用组件中的 ZMBuf 缓存类进行存取，每隔 15 分钟将内存中的数据保存成文本文件到硬盘。

声明：

<?php  
namespace Module\Example;  
  
use ZM\Annotation\Module\SaveBuffer;  
/\*\*  
 \* @SaveBuffer(buf\_name="test\_list",sub\_folder="Test")  
 \*/  
class Test { }

使用：发送 **你好**，并记录调用此功能的人的时间并统计次数

/\*\*  
 \* @CQCommand("你好")  
 \*/  
public function hello() {  
 $list = ZMBuf::get("test\_list");  
 $list[]=["user\_id" => ctx()->getUserId(), "time" => time()];  
 ZMBuf::set("test\_list", $list);  
 return "成功调用！";  
}  
/\*\*  
 \* @CQCommand("调用次数")  
 \*/  
public function times() {  
 $cnt = 0;  
 foreach(ZMBuf::get("test\_list") as $k => $v) {  
 if($v["user\_id"] == ctx()->getUserId()) {  
 ++$cnt;  
 }  
 }  
 ctx()->reply("调用次数：" . $cnt);  
}

使用了 SaveBuffer 后，与直接使用 ZMBuf 类不同的是，框架会自动把此变量保存到硬盘，即使框架退出，计算机关闭或内存炸掉了，只要重启，就会自动往 ZMBuf 中加载之前保存的缓存变量。而直接使用 ZMBuf 没有声明 SaveBuffer 时，框架停止运行后数据即丢失。

## LoadBuffer()

在 1.4 版本起可用。

与上方 SaveBuffer 类似，但 LoadBuffer 不会将更改保存回文件，仅仅是加载到内存中，通过 ZMBuf 调用使用。如果原本文件不存在，则初始化为空数组。

<?php  
namespace Module\Example;  
  
use ZM\Annotation\Module\SaveBuffer;  
/\*\*  
 \* @LoadBuffer(buf\_name="test\_list",sub\_foler="Test")  
 \*/  
class Test{}