# HTTP 服务器

最先开始的时候，我们说了 炸毛框架 不仅仅是聊天机器人的框架，还是一个完整的 HTTP + WebSocket 服务器，你不仅可以开发机器人功能，框架还可用于开发任何 Web 应用程序，构建高性能的 Web 系统、API、中间件、基础服务等。

::: tip 提示

炸毛框架的注解和 HTTP 服务器控制器响应方式高度类似 **Swoft** 框架，如果你熟悉 Swoft 框架的话，这里学习起来将会非常容易。

:::

## 功能特色

* 基于 @RequestMapping 和 @Controller 的路由注解
* 提供简单的参数支持
* 响应迅速，无任何臃肿的框架代码，无过多封装，直接操作原生的 Swoole 对象

在一般情况下，我们启动了炸毛框架后，HTTP 服务器就自动启动了，访问框架监听的地址和端口，就可以访问 HTTP 服务。

## 控制器和路由

控制器主要通过 @Controller 注解实现。此注解仅适用于对类进行修饰。

* 名称：@Controller
* 命名空间：ZM\Annotation\Http\Controller
* 参数：\*prefix
* 作用：声明此类下所有路由的根路由地址。
* 名称：@RequestMapping
* 命名空间：ZM\Annotation\Http\RequestMapping
* 参数：\*route，name，method
* 作用：路由地址

### 路由规则

通常情况，一个完整的路由 path 等于 @Controller 的 prefix + @RequestMapping 的 route。例如：

<?php  
namespace Module\Example;  
  
use ZM\Annotation\Http\Controller;  
use ZM\Annotation\Http\RequestMapping;  
/\*\*  
 \* @Controller("/api")  
 \*/  
class Test {  
 /\*\*  
 \* @RequestMapping("/index")  
 \*/  
 public function index(){  
 //context()->getResponse()->end("This is API index page");  
 ctx()->getResponse()->end("This is API index page");  
 }  
 /\*\*  
 \* @RequestMapping("/ping")  
 \*/  
 public function ping(){  
 ctx()->getResponse()->end("pong");  
 }  
}

以上编写后，从浏览器访问 http://127.0.0.1:20001/api/index 时就会访问到 This is API index page，访问 /api/ping 时就会返回 pong。

::: tip 提示

当 @Controller 为 / 的时候，效果和不写是一样的，@RequestMapping 为 / 或 /index/inside 等多级路由也是可以的。

:::

## 绑定参数

在 @RequestMapping 中，不仅可以写静态的路由地址，也可以写绑定的参数。例如：@RequestMapping(route="/index/{name}")，则访问 /index/xxx 的时候，你在函数方法内可以这样获取此参数：

public function index($arg) {  
 ctx()->getResponse()->end("Your param 'name' is ".$arg["name"]);  
}

## 设置路由请求方式

如果想要设置允许请求控制器的 HTTP 请求方式，可以使用方法在控制器中的 @RequestMapping 注解配置 method 参数，可以是 GET，POST，PUT, PATCH，DELETE，OPTIONS，HEAD 中的一个或多个。

* 限定 HTTP 方法：@RequestMapping(method="GET")，@RequestMapping(method={"GET","POST"})

## 事件可用的基类变量

* $this->request：swoole\_http\_request 对象，可查阅 [Swoole 文档](https://wiki.swoole.com/#/http_server?id=httprequest)
* $this->response：swoole\_http\_response 对象，可查阅 [Swoole 文档](https://wiki.swoole.com/#/http_server?id=httpresponse)

::: tip 提示  
对于 Request HTTP 请求事件，在函数内还可以使用 context()->getRequest() 和 context()->getResponse() 来获取上述对象。  
:::

## 中间件

对于 @RequestMapping 方法，还支持中间件，完成 Session 会话、认证等功能。中间件是用于控制 请求到达 和 响应请求 的整个流程的。从一定意义上来说相当于切面编程（AOP）。目前该中间件仅支持 @RequestMapping 绑定的路由函数。

::: tip 提示

从 1.3.0 版本开始，框架的任何事件类注解绑定均支持中间件的绑定了，而且还支持多个中间件的绑定，此部分原先内容已迁移到 [中间件注解](/guide/event/middleware.html) 部分。

:::

## HTTP 客户端

框架内依赖了 Swoole 官方推荐的 Saber，可直接在任意位置使用。这里是 [Saber 地址](https://github.com/swlib/saber)。

对于 GET 请求 和 POST 请求，框架内也封装了轻量的函数调用，如果使用 file\_get\_contents 类和 curl\_exec 之类的网络请求函数，在 1.1 版本之后开启了 [一键协程化](https://wiki.swoole.com/#/runtime?id=常见的hook列表)，也可以正常使用。Saber 和框架内的轻量化函数都是协程优化的，也都可以正常使用。

框架内的轻量函数见组件中 [ZMRequest 组件](/guide/component/zmrequest.html)

## 静态文件服务器

从 1.1.2 版本开始，框架支持了静态文件的访问。如需使用，则需要先到配置文件中配置相应的 static\_file\_server 参数，这里是 [配置文件](/guide/configuration.html)。

设置完成后，访问框架的地址，加上文件名，例如在默认路径下自带的 resources/html/static.html 文件，通过访问 http://127.0.0.1:20001/static.html 就可以访问了。

你可以将前后端分离的前端部分放入这里进行分发和部署。

## 局部静态文件服务

此功能 1.4 版本开始可用。

上面的静态文件服务器为全局的，如果你只需要在一个路由下使用静态文件服务，则按照如下例子即可。

注意要先 use ZM\Http\StaticFileHandler;

/\*\*  
 \* @RequestMapping("/images/{filename}")  
 \* @param $param  
 \* @return StaticFileHandler  
 \*/  
public function staticImage($param) {  
 Console::info("[lexicon] " . $param["filename"]);  
 return new StaticFileHandler($param["filename"], "/path/to/your/images/");  
}

上面假设你在 /path/to/your/images/ 这个目录下有大量的图片，比如你有 a.jpg，浏览器访问：http://框架地址:端口/images/a.jpg 就可以获取图片了。