一、yaf 介绍

a) 官网手册（http://www.laruence.com/manual/index.html）

b) Yaf 的特点 （http://www.laruence.com/manual/yaf.infos.html）

i. Yaf 是一个 C 语言编写的 PHP 框架

ii. 以 PHP 扩展方式编写的 PHP 框架

c) Yaf 的优点（http://www.laruence.com/manual/yaf.advantages.html）

二、 统一开发环境

a) 使用 Vagrant 打造跨平台开发环境 （通过网络获取安装步骤）

b) 编译安装 LNMP 环境（略）

三、yaf 的安装与 ide 配置

a) 安装

i. 获取资源

1. https://github.com/laruence/yaf

2. http://pecl.php.net/package/yaf （建议使用稳定版本）

ii. 安装了 gcc、gcc-c++、make、automake、autoconf 等依赖库

1. unbuntu

a) sudo apt-get install gcc gcc-c++ make automake autoconf

2. centos

a) yum install gcc gcc-c++ make automake autoconf

iii. 将压缩包上传服务器（建议使用 github）

iv. 解压文件包并进入文件夹目标

v. 预处理并安装 （使用 phpize 脚本预处理生产 configure 配置文件）

{$path}/php/bin/phpize

./configure --with-php-config={$path}/php/bin/php-config

make && make install

vi. 查看确认编译后的文件 （yaf.so）

1. {$path}/php/lib/php/extensions/no-debug-zts-20121212/

vii. 配置 php.ini

vi {$path}/php/etc/php.ini

extension=yaf.so //关键步骤:载入 yaf.so

viii. 重启服务（nginx　php-fpm）

ix. 检测是否加载成功 （phpinfo()）

b) 创建演示项目（https://github.com/laruence/yaf/tree/master/tools/cg）

i. 执行 {$path}/php/bin/php yaf_cg demo

ii. 在此 yaf-master/tools/cg/output 目录生成演示项目

c) 配置 IDE 环境（https://github.com/elad-yosifon/php-yaf-doc）

i. 编写代码时提示补全代码，实现代码跟踪

ii. 下载代码 php-yaf-doc，添加到 IDE 环境配置中即可（External Libraries）

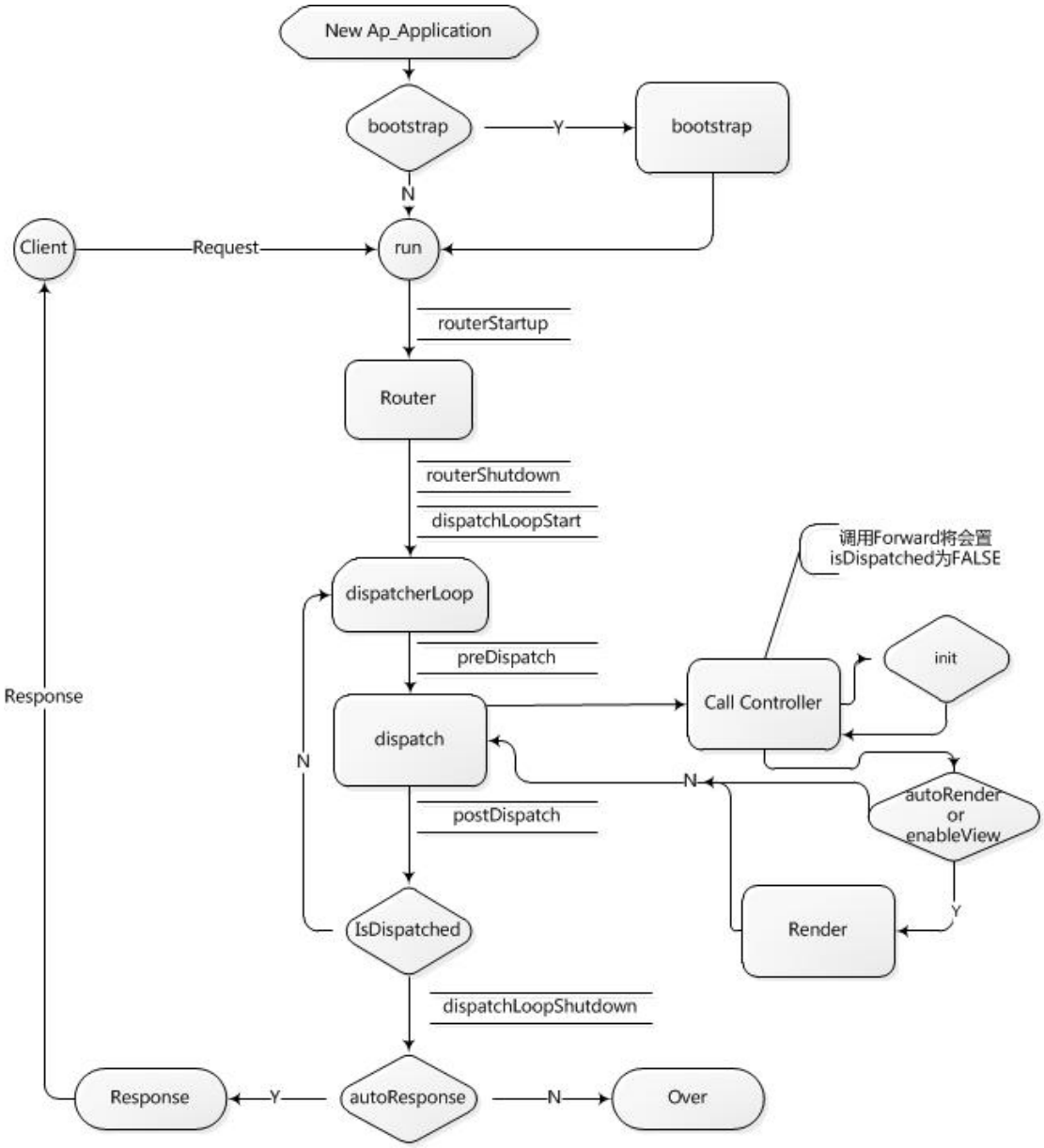
四、yaf 的 nginx 配置

a) 站点配置 （https://github.com/laruence/yaf）

```
server {
   listen ****；
   server_name    domain.com；
   root     document_root；
   index    index.php index.html index.htm；
```

```
    if (!-e $request_filename) {
        rewrite ^/(.*)  /index.php  last；  //与官方有区别
    }
}
```

五、Yaf 运行流程



a) 执行单一入口文件 index.php 会实例化应用

b) 加载配置文件$application->bootstrap()->run();运行应用

c) 当选择加载 bootstrap 时会执行应用 bootstrap.php 脚本，反之

d) Client ----Request--->Run 客户端请求开始之前 YAF 会进行路由处理

e) 先执行插件目录下 6 个 Hook 回调函数，插件之间的执行顺序是先进先 Call（路由开始，路由结束，循环调度开始，调度开始，调度结束，循环调度结束。）

注意：在路由开始之前是无法获取模块，控制器，方法的，路由完成才能得到，所以路由开始时可以作一些登录验证，路由完成时可以作一些 RBAC 的控制。

http://www.laruence.com/manual/yaf.plugin.times.html

f) 路由结束后执行单个循环调度 preDispatch 会根据路由规则得到路由结果获取模块，控制器，方法从而执行对应的脚本。

g) 当调用控制器时会首先执行 init 方法，接着执行控制器中方法

h) 执行控制器的方法后是否自动渲染视图

i) 返回 postDispatch 单次循环调度结束，接着再根据 Forward 判断是否再次调度

indexAction(){ $this->forward('index','index','show')}

//执行 index 模块下的 index 控制器 show 方法

j) 循环调度结束后执行 dispatchLoopShutdown，再判断是否自动响应

k) 开始自动响应则将结果返回给客户端（Yaf_Response_Abstract）处理相应的类

六、Yaf-Bootstarp 介绍

a) 在一个 Yaf_Application 被实例化之后，运行(Yaf_Application::run)之前，可选的我们可以运行 Yaf_Application::bootstrap

b) 运行 Bootstrap 是用来在 Application 运行(run)之前做一些初始化工作的机制.

指示 Yaf_Application 去寻找 Bootstrap，并按照声明的顺序，执行所有在 Bootstrap 类中定义的以_init 开头的方法。 如果没有提供变量 bootstrap，Yaf 默认会去 application.directory 中寻找 Bootstrap。

http://www.laruence.com/manual/ch06s02.html

c) 可选修改文件 Bootstrap.php 目录

http://www.laruence.com/manual/yaf.config.optional.html

d) 传递对象 public function _initRoute(Yaf_Dispatcher $dispatcher) { }

传递 Yaf_Dispatcher 内建对象类（类型约束）

http://php.net/manual/zh/language.oop5.typehinting.php

七、Yaf 配置

a) php.ini [yaf]

i. yaf.library 全局类库的目录路径（公共类库）

ii. yaf.use_namespace 开启的情况下, Yaf 将会使用命名空间方式注册自己的类, 比如 Yaf_Application 将会变成 Yaf\Application

vi /usr/local/php/etc/php.ini

```
[yaf]
yaf.environ = product
yaf.library = NULL
yaf.cache_config = 0
yaf.name_suffix = 1
yaf.name_separator = ""
yaf.forward_limit = 5
yaf.use_namespace = 1    //建议开启命名空间
yaf.use_spl_autoload = 0
extension=yaf.so //关键步骤:载入 yaf.so ,上面也可忽略
```

注意：配置 yaf.environ 时需要 application.ini 添加对应配置段

[product: common]

b) Application.ini 应用配置文件

i. http://www.laruence.com/manual/yaf.config.html

ii. Yaf 通过在不同的环境中，选取不同的配置节，再结合配置可继承，来实现一套配置适应多种环境(线上,测试,开发).

iii. 在配置文件中添加配置段时一定要加在对应环境配置上

```
[common]
application.directory = APPLICATION_PATH  "/application"
application.dispatcher.catchException = TRUE

[redis]
reids.cache.host = localhost
reids.cache.port = 6379
reids.cache.db = 1

[product : common :redis]
```

c) 可选的配置项

i. http://www.laruence.com/manual/yaf.config.optional.html

八、YAF 类库的加载规则

框架的一个重要功能就是类的自动加载了 ,框架就基于这个目录结构来自动加载需要的类文件。 Yaf 为了方便在一台服务器上部署的不同产品之间共享公司级别的共享库，支持全局类和本地类两种加载方式.

全局类是指，所有产品之间共享的类，这些类库的路径是通过 ap.library 在 php.ini(当然,如果 PHP 在编译的时候，支持了 with-config-file-scan-dir,那么也可以写在单独的 ap.ini 中)而本地类是指，产品自身的类库，这些类库的路径是通过在产品的配置文件中，通过 ap.library 配置的.

在 Yaf 中，通过调用 Yaf_Loader 的 registerLocalNamespace 方法，来申明那些类前缀是本地类,即可.

http://www.laruence.com/manual/yaf.autoloader.html#yaf.autoloader.library

a) 全局类库

i. php.ini 配置 yaf.library （全局类库的目录路径）

b) 本地类库

i. application.library (本地(自身)类库的绝对目录地址)

c) 加载顺序

i. 先到全局类库中加载，接着到本地类库寻找

ii. Tool\Http::gethost(); //使用了命名空间

{类库路径(php.ini 中指定的 yaf.library)}/Tool/Http.php

d) 注册本地类

i. 如果你想使用 Yaf_Loader 搜索本地类（库）（定义在 application.ini， 默认情况下，它是 application.directory . "/libraray" ）， 你需要使用 Yaf_Loader::registerLocalNameSpace() 注册本地类前缀。

```
class   Bootstrap   extends   Yaf_Bootstrap_Abstract {
  public function _initLoader(Yaf\Dispatcher $dispatcher){
      $loader = Yaf\Loader::getInstance();
      $loader -> registerLocalNamespace("Tool");
   }
}
```

或者在 php.ini 定义了全局类库 yaf.library = "/home/webroot"时如果优先加载本地类库需要在 application.ini 配置文件中指定加载的本地类前缀目录

application.library.namespace=" Tool，Cache"

{$path}/application/library/Tool

```
namespace   Tool;
class Http{
    static   function gethost(){
        return $_SERVER['HTTP_HOST'];
    }
}
```

{$path}/application/controllers/Index.php

```
public function indexAction($name = "Stranger") {
    echo Tool\Http::gethost();
}
```

e) 手动加载类

i. yaf_Loader::import （ '绝对路径' ）

ii. yaf_Loader::import （ ‘相对路径’ ）//会在 library 目录下寻找 s

九、YAF 模块与控制器

a) 控制器：接收用户的请求，根据业务类型交给模型去处理，处理之后进行视图渲染

http://yafmy.com/index/index/index -》 http://yafmy.com/模型/控制器/动作

b) 默认路由协议 （http://www.laruence.com/manual/yaf.routes.static.html）

i. 默认的路由协议 Yaf_Route_Static，就是分析请求中的 request_uri，在去除掉 base_uri 以后（如果我们的 application 是通过文件夹 base 访问的，baseUri 就是/base），获取到真正的负载路由信息的 request_uri 片段，具体的策略是，根据"/"对 request_uri 分段，依次得到 Module,Controller,Action，在得到 Module 以后，还需要根据 Yaf_Application::$modules 来判断 Module 是否是合法的 Module，如果不是，则认为 Module 并没有体现在 request_uri 中，而把原 Module 当做 Controller，原 Controller 当做 Action

c) 控制器（不支持驼峰命名控制名）

http://www.laruence.com/manual/tutorial.firstpage.html#tutorial.controller

d) 当用户在地址栏 http://yafmy.com/user/index/show 输入时，先会寻找模块 user 下 index 控制器中的 show 方法，当输出不合法的模块时就使用默认的模块 index，系统会报 application/controllers/User.php: No such file or directory，不会报模块找不到，use 模块不合法就直接寻找默认模块 index 下 user 控制器中 index 方法）

e) 当方法名不合法（不存在）时会报错 There is no method showAction in IndexController，注意

f) 配置 Yaf 多模块开发

i. 在目录 application\下新建目录 modules。除了默认模块，其他模块都放在 application\modules\下。新建一个模块，模块名自定义。假设我的新模块叫 Api，创建目录 application\modules\Api。

ii. 修改项目配置文件 conf\application.ini

; 多个模块，使用逗号分隔

application.modules = "Index,Api"

iii. 在新模块下创建控制器，在目录 application\modules\Api\下创建控制器目录 controllers，用于存放模块 Api 下的控制器文件。

iv. 新建文件 application\modules\Api\controllers\Passport.php

```
class PassportController extends Yaf_Controller_Abstract {
    public function loginAction() {
        echo '我是登录接口';
        return false;
    }
}
```

v. 浏览器访问：http://127.0.0.1/api/passport/login

输出 ： 我是登录接口

g) 模块：把一些业务独立功能相关（比如用户模块，接口模块）用户模块就负责针对用户相关的操作的，接口模块就负责请求接口的业务逻辑的

十、Yaf 路由 http://www.laruence.com/manual/yaf.routes.html

a) YAF 本质是通过 REQUEST_URI 进行路由的$_SERVER[‘REQUEST_URI’]，不是 PATHINFO，所以在配置 NIGINX 时不会配置 pathonfo 只配置重写规则。

b) 获取请求的信息 Yaf_Request_Abstract

c) 路由和路由协议（http://www.laruence.com/manual/yaf.routes.html）

i. 概述 ，设计[重要]，默认情况，使用路由，路由协议

ii. 路由协议详解

1. Yaf_Route_Simple

//通过 Bootstrap.php

```
public function _initRoute(Yaf_Dispatcher $dispatcher) {
        //在这里注册自己的路由协议,默认使用简单路由
        $router = $dispatcher->getInstance()->getRouter();
        //  指定 3 个变量名  在 URL 参数名
        $route = new Yaf_Route_Simple("m", "c", "a");
        $router->addRoute("Simple", $route);
        //对于如下请求: "http://domain.com/index.php?c=index&a=test
    }
```

//通过 application.ini

```
;routes sample
[routes]
routes.index.type = "simple"
routes.index.module = "m"
routes.index.controller = "c"
routes.index.action = "a"
[product : common:routes]

public function _initRoute(Yaf_Dispatcher $dispatcher) {
        //在这里注册自己的路由协议,默认使用简单路由
        $router = $dispatcher->getInstance()->getRouter();
        $router ->addConfig(Yaf_Registry::get('config')->routes);
}
```

2. Yaf_Route_Regex

```
;routes regex
routes.index.type = "regex"
routes.index.match = "#^/([0-9])[\/]?$#";匹配域名斜杠后的参数  如:http://localhost/12
routes.index.route.module = "User"
routes.index.route.controller = "Index"
```

routes.index.route.action = "show";路由给 User 模块下 Index 控制器中 show 方法处理
routes.index.map.1 = name ;获取正则第一个模式单元匹配的数值赋给 name 参数
//添加

```
public function _initRoute(Yaf_Dispatcher $dispatcher) {
        //在这里注册自己的路由协议,默认使用简单路由
        $router = $dispatcher->getInstance()->getRouter();
        $router ->addConfig(Yaf_Registry::get('config')->routes);
}
```

备注：可以进行伪静态处理

```
routes.index.type = "regex"

routes.index.match = "#i^/list/([a-z]+).html?$#"    ;

routes.index.route.module = Index
routes.index.route.controller = Index
routes.index.route.action = index
routes.index.map.1 = name
```

访问：http://localhost/list/abc.html

```
"#i^/list/([a-z]+).[-]?([1-9]+)?.html?$#"
routes.index.map.1 =cat name
routes.index.map.2 = pagenum
```

访问：http://localhost/list/php-1.html http://localhost/list/mysql3.html

```
public function _initRoute(Yaf_Dispatcher $dispatcher) {
//在这里注册自己的路由协议,默认使用简单路由
//通过派遣器得到默认的路由器(默认路由器是:Yaf_Router;默认路由协议是:Yaf_Rout_Static)

    $router = $dispatcher->getInstance()->getRouter();
    $arrRoute = array(
        "content" => new Yaf_Route_Regex(
            '#^/([a-zA-Z-_0-9]+).html$#',
            array(
                'controller' => 'content',
                'action' => 'action'
            ),
            array('1' => 'ident')
        ),
        "category" => new Yaf_Route_Regex(
            '#^category/(/[a-zA-Z-_0-9]+/[a-zA-Z-_0-9]+/)$#',
            array(
                'controller' => 'category',
                'action' => 'subcat'
```

```
                ),
                array('1' => 'ident')
            )

        );//注意分号

        foreach($arrRoute as $key => $val){
            $router->addRoute($key, $val);
        }
    }
```

十一、YAF 视图

a) Yaf 支持简单的视图引擎，并且支持用户自定义自己的视图引擎，比如 Smarty.对于默认模块，视图文件的路径是在 application 目录下的 views 目录中以小写的 action 名的目录中.

```
class IndexController extends Yaf_Controller_Abstract {
    public function indexAction() {//默认 Action
        $this->getView()->assign("content", "Hello World");
    }
}
```

b) 手动关闭视图渲染

i. Yaf_Dispatcher::getInstance()->autoRender(FALSE);

c) Ajax 关闭视图渲染

i. 使用 application.ini 配置文件配置 view.disableView=1

d) Ajax 关闭视图渲染

```
    public function init() {
        //如果是 Ajax 请求，则关闭 HTML 输出
        if ($this->getRequest()->isXmlHttpRequest()) {
            Yaf_Dispatcher::getInstance()->disableView();
        }
    }
```

e) http://www.laruence.com/manual/yaf.class.view.html#yaf.class.view.simple

f) Yaf_View_Simple::render 应用 （载入渲染一个视图模板，得到结果）

g) 设置模板的绝对路径

i. 设置模板的基目录，默认的 Yaf_Dispatcher 会设置此目录为 APPLICATION_PATH . "/views".

```
public funciton indexAction() {
        $this->getView()->setScriptPath("/tmp/views/");
}
```

h) 设置模板后缀名

application.view.ext="html" 默认 phtml 视图模板扩展名

十二、Yaf 请求与响应

a) 请求

i. $this->getRequest()->getPost(); //类推

b) 响应

i. 设置网页重定向

```
class IndexController extends Yaf_Controller_Abstract {
        public funciton init() {
            $this->getResponse()->setRedirect("http://domain.com/");
        }
 }
```

ii. 设置 body 体

```
$this->getRequest()-> setBody ( "Hello" );
$this->getRequest()-> response ();//自动响应
echo 'World';
```

输出：HelloWorldHello

```
$this->getRequest()-> setBody ( "Hello" );
$this->getRequest()-> response ();
echo 'World';
exit;//注册插件时会影响插件部分执行，当一个调用完成才执行自动响应
```

输出：HelloWorld

c) 关闭自动响应

```
public function _initView(Yaf_Dispatcher $dispatcher){
        //在这里注册自己的 view 控制器，例如 smarty,firekylin
        $dispatcher->getInstance()->returnResponse(true);//关闭自动响应
}
```

十三、Yaf 模型的使用

a) 概念

i. 对一个业务逻辑的封装，定义一个类提供入口也是实例化对象，对该对象进行操作（增删改查）

ii. 自动加载模型类文件

```
public function indexAction($name = "Stranger") {
    //1. fetch query
    $get = $this->getRequest()->getQuery("get", "default value");

    //2. fetch model
    $model = new SampleModel(); //实例化时就 yaf 自动加载类而不是 include
    //3. assign
    $this->getView()->assign("content", $model->selectSample());
    $this->getView()->assign("name", $name);
    //4. render by Yaf, 如果这里返回 FALSE, Yaf 将不会调用自动视图引擎 Render 模板
    return TRUE;
}
```

解析 :当在控制中 new SampleModel();实例化时 ,对于 Model 会自动加载路径为{项目目录}/models/ 类文件，类似比如 UserModel 则自动加载{项目目录}/models/User.php,注意大小写

b) 多站点模型设计

i. {项目目录}/models/Local/类文件 //本地库

namespace Local;//使用命名空间，让 yaf 识别文件目录

class UserModel{ }

在控制器中　new Local\UserModel(); 使用命名空间实例化对象

ii. {项目目录}/models/Remote/类文件 //请求接口获取数据(其他系统)

namespace Remote;//使用命名空间，让 yaf 识别文件目录

class UserModel{ }

在控制器中　new Remote\UserModel(); 使用命名空间实例化对象

十四、Yaf 工程结构

a) 目录结构设计

```
+ public
  |- index.php //入口文件
  |- .htaccess //重写规则
  |- favicon.jpg
  |+ css
  |+ images
  |+ js  //按照功能区分
+ conf
  |- application.ini //配置文件
+ application
  |+ controllers
      |- Index.php //默认控制器
  |+ views
      |+ index    //控制器
      |- index.phtml //默认视图
  |- Bootstrap.php //项目的全局配置,包括路由和 memcached 的配置等
  |+ modules //其他模块
      |+ controllers
      |+ views
  |+ library //本地类库
 |+ Tool
 |+ language //语言包
      |- Object.php //模型基类（多站点）
  |+ models  //model 目录
      |-User.php
  |+ plugins //插件目录
  |+ functions //函数目录
```

b) 全局类库 https://github.com/joshua317/izhengyin-yaf-libs

i. 数据库操作类

ii. 日志，请求，redis 类

十五、使用 Yaf 过程中，统一开发模式

a) 介绍使用 YAF 进行开发

i. 配置文件定义（站点，图片系统，数据主从，缓存，NOSQL，路由）

ii. MVC 分层

iii. 调用外部接口

b) Session 处理

c) 具体业务分层

i. 父类模型定义错误信息（错误码，错误信息，错误扩展信息）

d) 略（详细见视频）

十六、在 Yaf 中定义内网接口，Yar 的使用

a) 子系统之间数据的传输，共享

b) YAR 是 RPC 框架

i. 资源地址 https://github.com/laruence/yar

ii. 安装（略）

c) YAR 使用

i. Yar-demo
|-application
|-index.php
public function indexAction(){
try{
$client = new Yar_Client("http://yaf-demo/api/yar.php");
//调用服务端接口

```
            echo $client->add();
        }catch(Exception $e){
            echo $e-> getMessage();
        }
    }
```

ii. 在 Yaf 中使用 Yar

http://izhengyin.com/post/php/yaf-yar/

十七、在 Yaf 中使用命令行执行 PHP

a) 使用 CLI （命令行）

$path/php/bin/php cli.php request_uri = “cli/sample/run/name/zhangsan”

b) 案例

学会 yaf 命令行的调用方法:

php cli.php request_uri="/daemon/start"

request_uri=” /daemon/start” 中的路径便是 Controller 的路由路径。是指向 /Controller/Daemon.php 中的 startAction()。

第一种方式：

```
/*
* 执行方式：
$path/php/bin/php/webroot/yaf-demo/cli/cli.phprequest_uri="/cli/test/run/name/zhangsan"
 */
//创建 cli.php 入口文件
ini_set('display_errors',1);
error_reporting(E_ALL);
```

```
date_default_timezone_set('Asia/Shanghai');
define('APPLICATION_PATH', dirname(__DIR__));
$application = new Yaf\Application( APPLICATION_PATH .
"/conf/application.ini");
$application->bootstrap();
$application->getDispatcher()->dispatch(new Yaf\Request\Simple());
$application->run();
//使用命令行执行
php /webroot/yaf-demo/cli/cli.php request_uri="/cli/test/run"
//根据路由指定到 cli 模块下 test 控制下的 run 方法
\yaf-demo\application\modules\Cli\controllers
|--Test.php
    |--runAction()
```

第二种方式：（同理）

```
/**
 * 执行方式：$path/php/bin/php /data/webroot/yaf-demo/cli/cli2.php cli test
run '{"name":"zhangsan"}'
 */
ini_set('display_errors',1);
error_reporting(E_ALL);
date_default_timezone_set('Asia/Shanghai');
```

```
define('APPLICATION_PATH', dirname(__DIR__));
$application = new Yaf\Application( APPLICATION_PATH .
"/conf/application.ini");
$application->bootstrap();
$module =  isset($argv[1])?$argv[1]:exit("Lack params module!\n");
$controller = isset($argv[2])?$argv[2]:exit("Lack params controller!\n");
$action = isset($argv[3])?$argv[3]:exit("Lack params controller!\n");
$args = json_decode($argv[4],true);
$params = is_array($args)?$args:array();
$request = new
Yaf\Request\Simple("CLI",$module,$controller,$action,$params);
$application->getDispatcher()->dispatch($request);
$application->run();
```

即可结合 linux 定时任务使用