

# 梦之都 Yaf 教程

\*\*\*\*\*梦之都\*\*\*\*\*

- 1.梦之都致力于优秀绿色的教程供应，希望能够给您前进的道路上一点帮助。
- 2.官网地址:[www.mengzhidu.com](http://www.mengzhidu.com)，即"梦之都"的全拼。

\*\*\*\*\*与君共勉\*\*\*\*\*

- 1.学无止境，追求技术的道路上，我们一同前进。
- 2.当学习成为一种习惯，进步就是一种必然。
- 3.传播知识，传递温情。我心永恒，始终如一。

\*\*\*\*\*提示信息\*\*\*\*\*

- 1.本教程发布作者:辛星。
- 2.本教程发布时间:2016 年 12 月。
- 3.更新版本可以去官网查找。

\*\*\*\*\*更多交流\*\*\*\*\*

- 1.PHP 技术交流群:[459233896](https://t.me/459233896)。
- 2.官方网站:[www.mengzhidu.com](http://www.mengzhidu.com)。
- 3.我是辛星，我在这里等你。

\*\*\*\*\* 目录 \*\*\*\*\*

第零节:写在前面.....	3
第一节:yaf 安装.....	4
第二节:快速入门.....	7
第三节:控制器.....	10
第四节:视图.....	15
第五节:MVC 实现.....	17
第六节:引导文件.....	18
第七节:配置.....	20
第八节:类库加载.....	22
第九节:总结.....	24
附录:致敬读者.....	25

## 第零节:写在前面

\*\*\*\*\*鸟哥\*\*\*\*\*

在写 Yaf 之前,不得不提一下 Yaf 的作者,也就是大名鼎鼎的鸟哥。鸟哥之所以被称为鸟哥,是因为他的网名—雪候鸟。鸟哥真名"惠新宸",他自己给自己的介绍是:PHP 开发组成员, Zend 兼职顾问, PHP7 核心开发者, Yaf、Yar、Yac 等项目作者。

鸟哥也是国内最有影响力的 PHP 技术大牛, PHP 开发组的唯一一名国人,也是 PECL 的开发者,曾供职于雅虎、百度、新浪微博,现在应该在链家地产。

而 Yaf 框架也是出自鸟哥之手,该框架先后应用于百度、新浪的多款产品,是一个相当成熟的框架了。

Yaf 作为一个 C 语言编写的框架,在执行效率上确实要快很多,这一点在性能测试上确实会比较明显。

需要指出的是, Yaf 是一个轻量级的框架,它自带的功能并不多,很多东西还是需要我们去实现的。

\*\*\*\*\*Yaf\*\*\*\*\*

Yaf 是"Yet Another Framework"的简写,翻译成汉语即"另一个框架"。由于它是使用 C 语言开发的 PHP 框架,相比那些使用 PHP 编写的 PHP 框架来说,几乎不会带来额外的性能开销。

而且所有的框架类,无需编译,在 PHP 启动的时候加载,并且常驻内存。即它有着更快的执行速度,更少的内存占用。

它的内存周转周期更短,即可以提高内存利用率,降低内存占用率。而且它有一个高性能的视图引擎,还支持灵巧的自动加载,支持全局和局部加载规则,方便类库共享。

而且它是一个高度灵活可扩展的框架,支持自定义视图引擎、支持插件、支持自定义路由等我们大多数情况下的需要。而且对配置文件提供了良好的支持,并且支持缓存配置文件,有效避免复杂的配置结构带来的性能损失。

Yaf 框架的主要的亮点在于其卓越的性能,这也是我们选择 Yaf 的最重要的原因。

\*\*\*\*\*资源列表\*\*\*\*\*

yaf 手册地址:[点此打开](#)

鸟哥网站地址:[点此打开](#)

## 第一节:Yaf 安装

\*\*\*\*\*开始安装\*\*\*\*\*

那我们接下来就开始安装 Yaf 框架吧，它是通过一个扩展的方式来实现的，我们可以去 [pecl.php.net](http://pecl.php.net) 去搜索 Yaf 这个扩展，我们也可以直接到 "<http://pecl.php.net/package/yaf>" 这里进行访问，界面如下：

带beta的是测试版  
带stable的是稳定版

Available Releases				
Version	State	Release Date	Downloads	
3.0.3	stable	2016-07-02	<a href="#">yaf-3.0.3.tgz</a> (93.6kB) <a href="#">DLL</a>	<a href="#">[Changelog]</a>
3.0.2	beta	2015-12-28	<a href="#">yaf-3.0.2.tgz</a> (93.4kB) <a href="#">DLL</a>	<a href="#">[Changelog]</a>
3.0.1	beta	2015-12-12	<a href="#">yaf-3.0.1.tgz</a> (93.0kB) <a href="#">DLL</a>	<a href="#">[Changelog]</a>
3.0.0	beta	2015-10-27	<a href="#">yaf-3.0.0.tgz</a> (92.5kB) <a href="#">DLL</a>	<a href="#">[Changelog]</a>
2.3.5	stable	2015-09-06	<a href="#">yaf-2.3.5.tgz</a> (102.3kB) <a href="#">DLL</a>	<a href="#">[Changelog]</a>
2.3.4	stable	2015-08-13	<a href="#">yaf-2.3.4.tgz</a> (102.3kB) <a href="#">DLL</a>	<a href="#">[Changelog]</a>
2.3.3	beta	2014-10-22	<a href="#">yaf-2.3.3.tgz</a> (102.8kB) <a href="#">DLL</a>	<a href="#">[Changelog]</a>
2.3.2	beta	2014-01-09	<a href="#">yaf-2.3.2.tgz</a> (102.7kB) <a href="#">DLL</a>	<a href="#">[Changelog]</a>
2.3.1	beta	2014-01-08	<a href="#">yaf-2.3.1.tgz</a> (102.5kB) <a href="#">DLL</a>	<a href="#">[Changelog]</a>
2.2.9	stable	2013-01-04	<a href="#">yaf-2.2.9.tgz</a> (92.8kB) <a href="#">DLL</a>	<a href="#">[Changelog]</a>

对于 state 来说，其中 beta 是测试版，stable 是稳定版。一般来说，在学习和实验阶段，选择哪个都无所谓，但是在正式的项目中，我们一般都应该选择稳定版。而且这里的 3.x 系列的是为 PHP7 准备的，如果朋友们还在使用 PHP5 的话，可以考虑使用 2.x 系列的版本。由于考虑到目前使用 PHP5 的还是占多数，因此本教程以 Yaf2.3 为蓝本进行介绍。

这里假设朋友们使用的是 Windows 操作系统，我们点击 Downloads 中的 "DLL" 既可下载对应的 dll 文件，这里我们以 2.3.5 为例来进行说明，然后我们来到如下页面：

yaf 2.3.5

Package Information	
Summary	PHP Framework in PHP extension
Maintainers	Xinchen Hui < <a href="mailto:laruence@php.net">laruence@php.net</a> > (lead) <a href="#">[details]</a>
License	PHP
Description	Yaf is a PHP framework similar to zend framework, which is written in c and built as PHP extension
Homepage	<a href="http://www.yafdev.com/">http://www.yafdev.com/</a>
Release notes Version 2.3.5 (stable)	- Fixed bug (redirect doesn't work)

DLL List	
PHP 5.6	<a href="#">5.6 Non Thread Safe (NTS) x86</a> <a href="#">5.6 Thread Safe (TS) x86</a> <a href="#">5.6 Non Thread Safe (NTS) x64</a> <a href="#">5.6 Thread Safe (TS) x64</a>
PHP 5.5	<a href="#">5.5 Non Thread Safe (NTS) x86</a> <a href="#">5.5 Thread Safe (TS) x86</a> <a href="#">5.5 Non Thread Safe (NTS) x64</a> <a href="#">5.5 Thread Safe (TS) x64</a>
PHP 5.4	<a href="#">5.4 Non Thread Safe (NTS) x86</a> <a href="#">5.4 Thread Safe (TS) x86</a>
PHP 5.3	<a href="#">5.3 Non Thread Safe (NTS) x86</a> <a href="#">5.3 Thread Safe (TS) x86</a>

根据32位或者64位操作系统以及PHP的版本来选择

需要说明的是 TS 和 NTS 的区别，其中 TS 是线程安全的，NTS 是非线程安全的。

那么我们如何来判断我们当前的版本是否是线程安全的呢，我们可以在 `phpinfo()` 中查看 Thread Safety 这一项的内容，如果是 enabled，那么就是线程安全的，否则就是非线程安全的。我这里来一个截图范例：

Debug Build	no
Thread Safety	enabled
Zend Signal Handling	disabled

这里表示是线程安全的

我们根据我们的 PHP 版本和操作系统来选择对应的文件，我们下载文件且解压后得到的是这么几个文件：

Windows8_OS (C:) > Users > VOL01 > Downloads > php_yaf-2.3.5-5.5-ts-vc11-x64			
名称	修改日期	类型	大小
composer.json	2015/9/6 18:38	JSON 文件	1 KB
CREDITS	2015/9/6 18:44	文件	1 KB
LICENSE	2015/9/6 18:44	文件	4 KB
php_yaf.dll	2015/9/6 18:44	应用程序扩展	211 KB
php_yaf.pdb	2015/9/6 18:44	PDB 文件	859 KB

然后我们把 `php_yaf.dll` 放到 PHP 的扩展目录中即可，那么 PHP 的扩展目录在哪呢？一般来说是 PHP 安装目录中的 `ext` 目录，这个目录就是 PHP 的扩展目录，我这里来个截图吧：

php_sybase_ct.dll	2014/5/1 14:48	应用程序扩展	38 KB
php_tidy.dll	2014/5/1 14:48	应用程序扩展	277 KB
php_vld.dll	2013/9/25 0:29	应用程序扩展	31 KB
php_xcache.dll	2014/9/19 11:01	应用程序扩展	143 KB
php_xlrpc.dll	2014/5/1 14:48	应用程序扩展	61 KB
php_xsl.dll		应用程序扩展	278 KB
<input checked="" type="checkbox"/> php_yaf.dll		应用程序扩展	189 KB

就是这个文件

然后我们还需要在 `php.ini` 中在扩展那里追加 "`extension=php_yaf.dll`"，这里来个截图范例吧：

```
extension=php_pdo_pgsql.dll
extension=php_pdo_sqlite.dll
extension=php_pgsql.dll
extension=php_shmop.dll
extension=php_yaf.dll
extension=php_redis.dll
extension=php_igbinary.dll
extension=php_xcache.dll
extension=php_phalcon.dll
```


注意前面不能有分号噢

至此，这个扩展就安装完毕了。

\*\*\*\*\*检测安装\*\*\*\*\*

那么如何来检测我们的 Yaf 是否安装成功了呢?我们可以在 phpinfo() 中找一下，如果朋友们看到如下类似的内容，那么说明安装成功了：

### yaf

yaf support	enabled	
Version	看到这一段，说明Yaf安装成功	
Supports		
a/yaf		

Directive	Local Value	Master Value
yaf.action_prefer	Off	Off
yaf.cache_config	Off	Off
yaf.envIRON	product	product
yaf.forward_limit	5	5
yaf.library	no value	no value
yaf.lowercase_path	Off	Off
yaf.name_separator	no value	no value
yaf.name_suffix	On	On
yaf.st_compatible	Off	Off
yaf.use_namespace	Off	Off
yaf.use_spl_autoload	Off	Off

在安装完扩展之后，我们接下来的操作就会容易很多，然后让我们一起来开始接下来的学习吧。如果扩展安装失败，可以仔细对照一下上述步骤，看看哪里有问题，如果忘记了哪个步骤可能会有问题，重来一次也是不错的选择奥。

## 第二节:快速入门

### \*\*\*\*\* 目录结构 \*\*\*\*\*

在安装完扩展之后，我们就可以考虑进行项目的开发了，这里我是在 WWW 目录下新建了一个 yaf 目录，我们的所有代码都在此目录中进行开发。

那么这个项目的目录结构该怎么编写呢?通常建议是这样的:

```
/----public    公共目录
|
|-----index.php  入口文件
|
|-----css      样式目录
|
|-----img      图片目录
|
|-----js       脚本目录
|
|
|----conf      配置目录
|
|-----application.ini  应用程序配置
|
|
|----application  应用程序目录
|
|-----controllers  控制器目录
|
|-----views      视图目录
|
|-----models     模型目录
|
|-----modules    模块目录
|
|-----library    类库目录
|
|-----plugins    插件目录
|
|-----Bootstrap.php  引导文件
|
|
|----.htaccess  重写文件
```

上述目录是我们在通常情况下会建立的目录，不过这里作为入门，本着简化的目的，我们通常并不需要建立那么多的目录和文件。

但是我们也至少需要一个能够让 Yaf 框架运行起来的最小目录，我们可以这么建立:

```
/----public    公共目录
|
|-----index.php  入口文件
|
|
|
|----conf      配置目录
|
|-----application.ini  应用程序配置
|
|
|----application  应用程序目录
|
|-----controllers  控制器目录
|
|-----views      视图目录
```

```
|  
|----.htaccess 重写文件
```

这也就是我们所构建的最小的程序所需要的目录了，首先我们创建出来上面的文件和目录结构，然后后面我们就需要去对它们编写具体的代码了。

\*\*\*\*\*编写文件\*\*\*\*\*

首先我们编写.htaccess文件，它这里在主要是实现了对url的重写(我们这里以Apache服务器为例)，它可以让我们url中不出现public/index.php，但是却可以访问到它，这个会随着具体的名称的变动会有所变化，我们这里可以这么写：

```
RewriteEngine On  
RewriteCond %{REQUEST_FILENAME} !-f  
RewriteRule ^(.*)$ public/index.php/$1 [L]
```

然后就是public目录里面的index.php，它是全局的入口文件，它首先定义一个常量，然后获取到配置文件，然后根据配置文件来实例化整个应用，然后调用\$app->run()来启动应用，其代码如下：

```
<?php  
//定义应用目录  
define("APP_PATH",realpath(dirname(__FILE__).'../'));  
//导入配置文件  
$app = new Yaf_Application(APP_PATH."/conf/application.ini");  
//运行程序  
echo $app->run();
```

然后我们编写conf目录里面的application.ini，它是全局的配置文件，它就是我们index.php里面引用的配置文件，这里我们可以只需要定义一下应用程序的目录即可，该文件代码如下：

```
[product]  
application.directory = APP_PATH"/application"
```

至此，我们的程序的骨架就已经搭建起来了，然后我们就可以开始写具体的功能代码了。

我们在application目录下的controllers目录下新建一个Index.php，然后我们写入如下代码：

```
<?php  
/**  
 * 所有的控制器都必须继承自 Yaf_Controller_Abstract  
 * 所有的控制器都必须用 Controller 后缀  
 *  
 */  
class IndexController extends Yaf_Controller_Abstract  
{
```



```
//所有的动作都需要带有后缀 Action
public function indexAction()
{
    //这里我们调用了视图
    //然后我们给视图中的内容进行赋值
    $this->getView()->assign("content","追梦之心，永不停息");
}
}
```

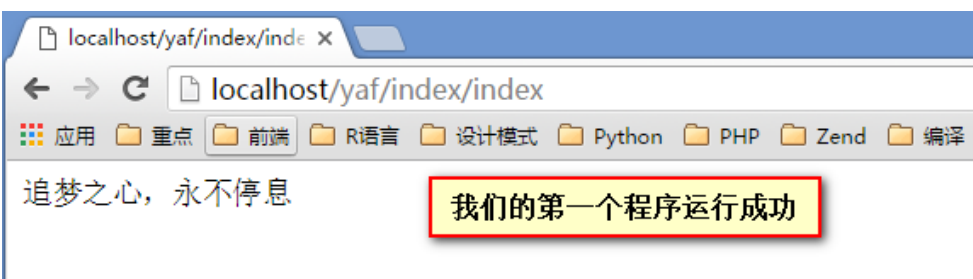
因为我们的文件名为 Index.php，它的首字母是需要大写的，而且该文件中的控制器的类名必须有 Controller 后缀，而且去掉后缀后的部分要和文件名保持一致，而且这个类必须继承自 Yaf\_Controller\_Abstract，而且它里面的所有的动作都必须使用 Action 后缀。

这里的 \$this->getView() 是获取默认的视图，然后 assign() 则是执行了赋值，我们的视图文件中有一个 \$content 变量，这里我们把这个值赋给了它。

然后我们在 views 目录下新建一个 index 目录，这个目录中对应的是我们的 IndexController，该目录下的所有视图文件都是和这个控制器有关系的。然后在 index 目录中新建一个 index.phtml 文件，它对应的是 indexAction。需要说明的是，这个后缀默认应该是 phtml。我们在该文件中输入如下代码：

```
<html>
    <body>
        <?php echo $content; ?>
    </body>
</html>
```

然后通过 url 来访问这个控制器下的这个动作了，这里我们通过 "localhost/yaf/index/index" 来访问它，需要说明的是，这里的 "localhost/yaf" 是我们的项目的 url 地址，而 "index/index" 中的第一个 index 则是对应的 IndexController 这个控制器，第二个 index 则是对应的 indexAction 这个动作，这里之所以没有写 "public/index.php" 是因为我们用 .htaccess 进行了重写。如果朋友们使用的是 nginx 或者其他非 Apache 服务器的话，这里也要进行相应的路由配置，否则可能会访问错误。然后我们来看一下具体的网页效果吧：



至此，我们的第一个通过 yaf 建立的应用终于看到了运行结果，不过由于这一步做的工作比较多，如果朋友们在这一步没有成功的话，可以多尝试几次奥。

### 第三节:控制器

\*\*\*\*\*书写规则\*\*\*\*\*

通常我们的控制器都定义在 application 目录下的 controllers 目录下，而且文件名的首字母要大写，而且文件中的类名的后缀是 Controller，去掉后缀之后要和文件名相同。

比如我们要定义一个关于"学生"的控制器，它的文件名我们通常定义为 Student.php，然后它的类名我们通常使用 StudentController。

每个控制器都需要继承自 Yaf\_Controller\_Abstract 这个类，它是所有控制器的父类。控制器中的动作都需要加 Action 后缀，比如 indexAction，比如 readAction 等等。

需要特殊说明的是，控制器是默认渲染模板的，我们可以通过如下语句来暂时屏蔽这一功能：

```
Yaf_Dispatcher::getInstance()->disableView();
```

比如我们这里写一个 StarController，我们在其中编写如下代码：

```
<?php
//定义 Star 控制器
class StarController extends Yaf_Controller_Abstract
{
    //定义 xin 动作
    public function xinAction()
    {
        echo "当学习成为一种习惯，进步就是一种必然";
    }
}
```

然后当我们访问上面这个动作的时候，我们却发现会报错说无法找到当前控制器对应的模板文件：

当学习成为一种习惯，进步就是一种必然

**(!)** Fatal error: Uncaught exception 'Yaf\_Exception\_LoadFailed\_View' with message 'Failed opening template C:\wamp\www\yaf\application\views\star\xin.phtml: No such file or directory' in C:\wamp\www\yaf\public\index.php on line 7

**(!)** Yaf\_Exception\_LoadFailed\_View: Failed opening template C:\wamp\www\yaf\application\views\star\xin.phtml: No such file or directory in C:\wamp\www\yaf\public\index.php on line 7

Call Stack

#	Time	Memory	Function	Location
1	0.0012	243200	{main} ()	..\index.php:0
2	0.0016	247888	run ()	..\index.php:7
3	0.0021	259384	render ()	..\index.php:7
4	0.0021	259576	render ()	..\index.php:0

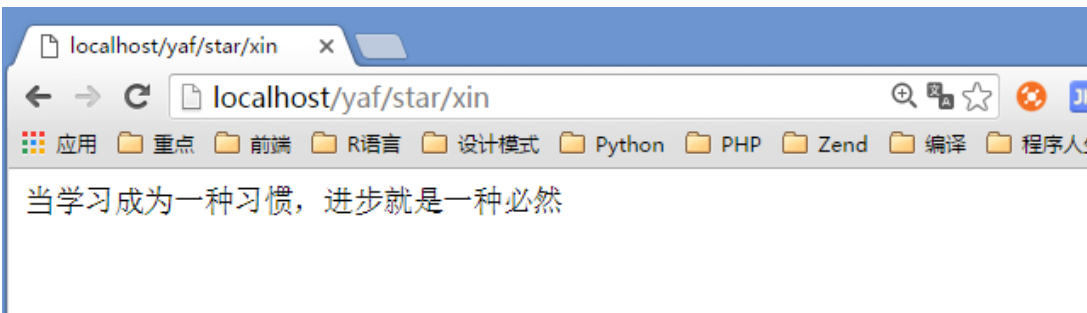
自动调用render()方法

这里它报告我们无法找到对应的模板文件

此时我们就可以关闭它，还记得我们上面介绍的那一行代码吗，我们添加之后，代码变成这样：

```
<?php
//定义 Star 控制器
class StarController extends Yaf_Controller_Abstract
{
    //定义 xin 动作
    public function xinAction()
    {
        Yaf_Dispatcher::getInstance()->disableView();
        echo "当学习成为一种习惯，进步就是一种必然";
    }
}
```

然后当我们再次访问该动作的时候，我们会发现就不会有任何问题了，截图如下：



我们可以在控制器中定义 `init()` 方法，它会在每个动作执行之前执行。通常我们可以做和本控制器下的所有的动作都相关的一些操作，比如对权限的判断等操作。

我们这里就简单的做一个内容的输出吧，我们这里新写一个控制器，代码内容如下：

```
<?php
//定义 Go 控制器
class GoController extends Yaf_Controller_Abstract
{
    //在动作执行之前执行
    public function init()
    {
        //禁止渲染视图
        Yaf_Dispatcher::getInstance()->disableView();
        //输出一条信息
        echo "来自 init()方法<br />";
    }
}
```

```

    }

    //定义 show 动作
    public function showAction()
    {
        echo "当学习成为一种习惯，进步就是一种必然";
    }
}

```

然后当我们调用它的 show 动作的时候，我们会发现它的 init() 方法会被自动执行，下面来个截图吧：



需要说明的是，通常我们不建议修改控制器的构造函数，我们可以把这些功能放到 init() 方法中，它们也会在我们的动作执行前被执行。

\*\*\*\*\*传递参数\*\*\*\*\*

很多时候，我们需要向我们的动作中传递参数，我们需要在我们的动作中写明我们的参数的变量名和变量的值。我们在 url 中使用"控制器/动作名/参数名 1/参数值 1/参数名 2/参数值 2..."的格式来请求即可。

我们可以写一个 Param 控制器，我们定义一个 msg 动作，它的方法中接受参数的，我们首先来看一下它的代码吧：

```

<?php
//定义 Param 控制器
class ParamController extends Yaf_Controller_Abstract
{
    //定义 msg 动作
    public function msgAction($name,$aim)
    {
        //禁止渲染视图
        Yaf_Dispatcher::getInstance()->disableView();
        //输出名字
        echo "您的名字是:".$name."<br />";
        //输出目标
    }
}

```

```

        echo "您的目标是:".$aim;
    }
}

```

根据我们前面的介绍，如果我们 name 取值为"辛星"，aim 取值为"PHP 大神"的话，那么"/aim/PHP 大神/name/辛星"和"/name/辛星/aim/PHP 大神"的作用是一样的，我们来看一下运行效果吧：



我们除了可以通过在定义函数的时候指定参数，我们还可以通过如下方式来获取参数：

```
$this->getRequest()->getParam(参数名);
```

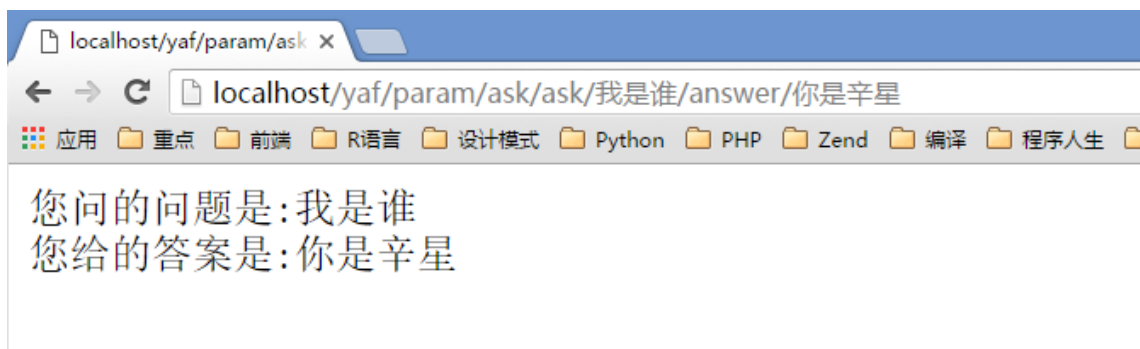
比如我们看一个范例，我们在 ParamController 中定义 askAction() 方法的代码如下：

```

//定义 ask 动作
public function askAction()
{
    //禁止渲染视图
    Yaf_Dispatcher::getInstance()->disableView();
    //获取参数
    $ask    = $this->getRequest()->getParam('ask');
    $answer = $this->getRequest()->getParam('answer');
    //进行输出
    echo "您问的问题是:".$ask."<br/>";
    echo "您给的答案是:".$answer."<br/>";
}

```

然后我们来看一下我们请求的范例吧，操作截图如下：



对于控制器的用法，我们就介绍到这里啦。

#### 第四节:视图

\*\*\*\*\*视图的使用\*\*\*\*\*

所谓视图，就是用来展示给用户的界面部分。我们可以用控制器的 getView() 方法返回一个视图的实例，然后我们调用其 assign() 方法来给视图中的每个变量进行赋值。而且需要注意的是，Yaf 会自动渲染视图，而且默认的文件后缀名为 phtml。

这里我们给一个范例吧，我们在 controllers 下面新建一个 Test.php 文件，然后我们写入如下代码：

```
<?php
class TestController extends Yaf_Controller_Abstract
{
    public function indexAction()
    {
        //定义数据
        $msg = array("love"=>"小倩","job"=>"开发工程师");
        //获取视图
        $view = $this->getView();
        //赋值一个标量
        $view->assign("name","辛星");
        //赋值一个数组
        $view->assign("msg",$msg);
    }
}
```

然后我们在 views 目录下新建一个 test 目录，然后在里面创建一个 index.phtml 文件，然后我们写入如下内容：

```
<html>
    <body>
        <p>您的姓名是:<?=$name?></p>
        <p>您喜欢的人:<?=$msg["love"]?></p>
        <p>您的职位是:<?=$msg["job"]?></p>
    </body>
</html>
```

然后我们来访问一下 <http://localhost/yaf/test/index>，我们就可以看到如下的界面了：



可能很多项目开发者会比较喜欢 Smarty 等模板，我们可以在 plugin 中来实现它，这里就不展开介绍了。



## 第五节:MVC 实现

\*\*\*\*\*简单介绍\*\*\*\*\*

对于 Web 开发来说,大多数操作都是增删改查,它们的实现机制都太相似了,这也导致 MVC 的思想成为了 Web 开发中的主流思想。所谓 MVC 即 Model-View-Controller, 即"模型-视图-控制器"。

不管读者朋友们之前是否接触过 MVC,那么在看了前面的介绍后,应该也大致了解了控制器和视图的用法,但是需要说明的是,我们的 Yaf 框架并没有提供实现。其实还没有实现的就是没有实现自己对数据库的操作,因为对于大多数框架来说,都会提供一套自己的对数据查询的方法。

其实仔细想一下,既然 PHP 已经提供了 PDO 了,因此这里无需再给出一套实现,也是合理的,而且事实上,我们也有很多的其他选择可供使用。如果想选择一个重量级的,可以考虑 Doctrine,如果想选择一个轻量级的,可以从其他框架上摘取出来,或者自己实现一个。

## 第六节:引导文件

\*\*\*\*\*具体实现\*\*\*\*\*

我们在 application 目录下，可以加一个 Bootstrap.php 文件，该文件会在执行控制器之前执行，而且通常我们在该文件中创建一个 Bootstrap 类，继承自 Yaf\_Bootstrap\_Abstract 类，而且该类中带有 "\_init" 前缀的方法都会被自动执行。

不过想让它真正的发挥作用，我们还需要在入口文件中调用 bootstrap() 方法，即我们的入口文件的代码这么写：

```
<?php
//定义应用目录
define("APP_PATH",realpath(dirname(__FILE__).'../'));
//导入配置文件
$app = new Yaf_Application(APP_PATH."/conf/application.ini");
//运行程序
echo $app->bootstrap()->run();
```

然后我们在 Bootstrap.php 中可以编写如下代码：

```
<?php
/**
 * 引导文件
 */
class Bootstrap extends Yaf_Bootstrap_Abstract
{

    public function _initXin()
    {
        //随便输出一些东西
        echo "来自 bootstrap <br/>";
    }

    public function _initConfig()
    {
        // 关闭自动加载模板
        Yaf_Dispatcher::getInstance()->disableView();
    }

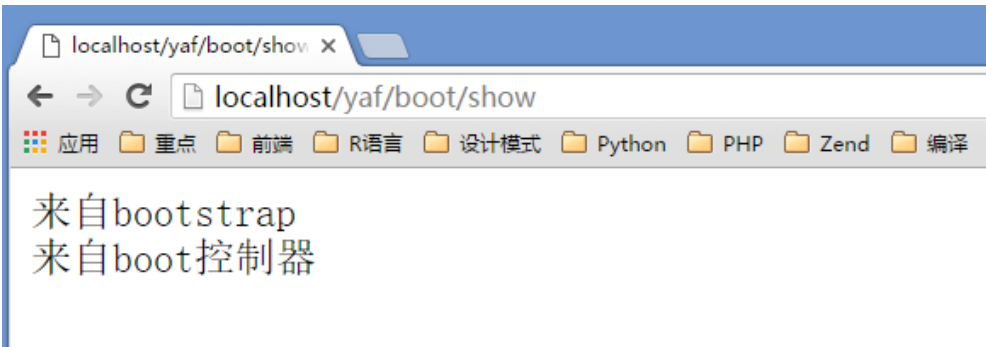
}
```

然后我们可以新建一个 BootController 类，然后我们写入如下代码：

```
<?php
//定义 Boot 控制器
class BootController extends Yaf_Controller_Abstract
```

```
{  
    //定义 show 动作  
    public function showAction()  
    {  
        echo "来自 boot 控制器";  
    }  
}
```

然后我们来访问一下这个动作，我们会看到我们的 Bootstrap 类中的方法被调用了，其运行效果截图如下：



对于 Bootstrap 的应用，我们就介绍到这里啦。

## 第七节:配置

\*\*\*\*\*配置简介\*\*\*\*\*

我们必须在 Yaf\_Application 初始化的时候给出配置文件中的配置，其中 Yaf 的必不可少的配置项只有一个，那就是 application.directory，它表示当前应用程序的绝对路径。

其实这个配置项也可以没有，至少从框架层面上做到了"零配置"，但是鸟哥认为，如果一个配置项都没有，就显得太寒酸了，于是就加了这么一个配置项。

\*\*\*\*\*常用配置\*\*\*\*\*

除了那个必选的配置项之外，我们还有若干的可选配置项，这些配置项我们都可以在 conf 目录下的 application.ini 文件中进行定义。

(1)application.ext 是 php 脚本的扩展名，它的值默认为 php，之所以会有这个配置项，是因为很早很早之前，我们也使用"php3"和"php4"作为后缀名，但是现在的文件我们一般都是使用"php"作为后缀名的，因此这个配置项可以忽略。

(2)application.bootstrap 是 bootstrap 文件的绝对路径，也就是我们上面介绍的引导文件，它通常是在 Yaf\_Application 实例化之后，我们调用它的 bootstrap() 方法来进行初始化设置。

(3)application.library 是本地类库的绝对路径地址，它的默认值为应用程序目录下的 library 目录，该目录下的类库会被自动加载。

(4)application.baseUrl 是我们在路由中，需要忽略的路径前缀，默认值为空，一般不需要设置，Yaf 会自动判断。

(5)application.dispatcher.defaultModule 是默认模块名，默认为 index。

(6)application.dispatcher.throwException 是在出错的时候是否抛出异常，默认值为 True，表示抛出异常。

(7)application.dispatcher.catchException 是捕获异常用的，我们一般使用 ErrorController 的 errorAction() 来处理这个异常，我们可以通过 \$request->getException() 来获取异常对象。

(8)application.dispatcher.defaultController 是默认控制器，默认为 index。

(9)application.dispatcher.defaultAction 是默认的动作，默认为 index。

(10)application.view.ext 是视图模板扩展名，默认为 phtml。

(11)application.modules 是存在的模块名，如果我们要定义这个值的话，一定要定义 Index 模块，默认为 Index。

\*\*\*\*\*读取范例\*\*\*\*\*

我们在平时的项目开发中，也会写大量的配置文件，这种情况下我们就可以动态的读取配置文件，这里我们通常也设置为 ini 格式文件，我们不妨在 conf 目录下新建一个 xin.ini 文件，内容如下：

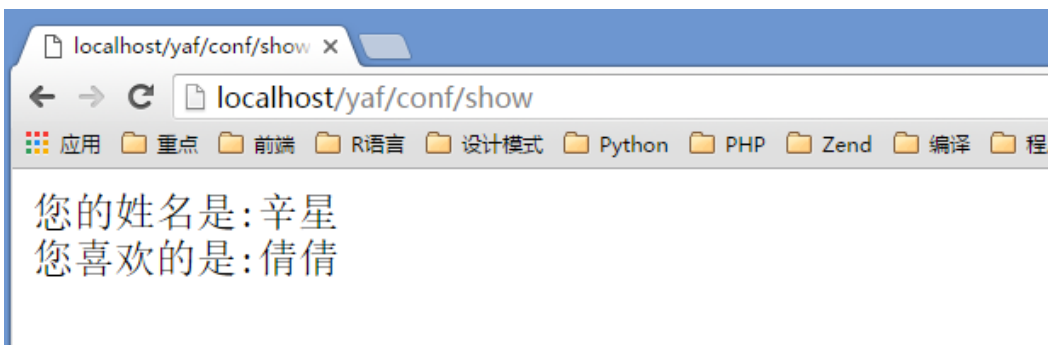
```
[star]
name=辛星
```

love=倩倩

然后我们可以通过Yaf\_Config\_ini类来读取这个配置，它需要传递这个配置文件的地址信息，在取出之后，我们就可以通过面向对象的方式来读取它了，下面是我们给出的一个操作范例：

```
<?php
//定义 Conf 控制器
class ConfController extends Yaf_Controller_Abstract
{
    //定义 show 动作
    public function showAction()
    {
        //不主动渲染视图
        Yaf_Dispatcher::getInstance()->disableView();
        //配置文件地址
        $path = APP_PATH."/conf/xin.ini";
        //配置类对象
        $config = new Yaf_Config_ini($path);
        //输出信息
        echo "您的姓名是:".$config->star->name."<br/>";
        echo "您喜欢的是:".$config->star->love;
    }
}
```

总的来说，上面的代码还是比较简单的，这里就不进一步介绍了，我们还是对运行结果给个截图吧：



对于很多框架来说，我们通常都是写到xxx.php文件中，但是在Yaf中通常还是比较建议写到这种xxx.ini文件中，通常来说我们并不太需要去关注性能产生的问题，还记得Yaf是一个用C语言写的框架吗。

## 第八节:类库加载

### \*\*\*\*\*类库加载实战\*\*\*\*\*

如果我们使用命名空间的话,那么对于类库的加载基本不会有什么疑问,但是对于那些已存在的没有使用命名空间的项目,那么我们在Yaf中是怎么进行加载的呢?

默认来说,类库的目录是 application 目录下的 library 目录,比如我们要写一个 Star 类,那么我们要创建一个 Star.php 文件,然后在类中对 Star 这个类进行定义。如果我们要进行目录的嵌套,那么类名应该用下划线对路径进行连接,并且每个路径的首字母大写。

比如我们要编写的是 Db 目录下的 Mysql 目录下的 Query.php 文件,那么我们的类名就应该是 Db\_Mysql\_Query,当然需要这里的 Db 目录是在 application 目录下的 library 目录下,这样Yaf框架就可以根据我们的类名来获取对应的路径,然后就可以实现对应的自动加载。

需要说明的是,我们的模型类(即 XxxModel)和控制器类(即 XxxController)都会被自动加载的,当然这个具有标志性的后缀是不可缺少的。

比如我们在 application 目录下新建一个 library 目录,然后我们在该目录中新建一个 Tool.php 文件,然后我们在里面书写如下代码:

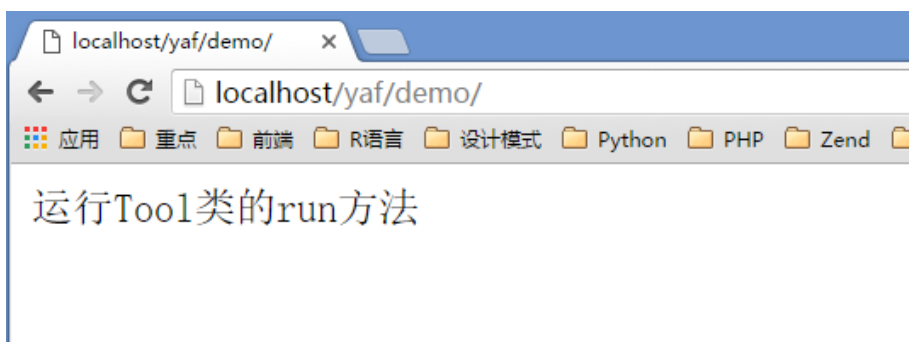
```
<?php
/**
 * 工具类 Tool
 */
class Tool
{
    //功能方法 run
    public function run()
    {
        echo "运行Tool类的 run 方法";
    }
}
```

那么我们在我们的控制器中就可以这么写:

```
<?php
/**
 * 这是 Demo 控制器
 */
class DemoController extends Yaf_Controller_Abstract
{
    //这是 index 动作
    public function indexAction()
```

```
{  
    //取消加载视图  
    Yaf_Dispatcher::getInstance()->disableView();  
    //实例化 Tool 类  
    $tool = new Tool();  
    //调用其方法  
    $tool->run();  
  
}  
}
```

在上面的代码中，我们的 Tool 类会被自动加载，然后我们可以成功的进行实例化，然后我们可以调用它的 run() 方法，这里我们来看一下效果吧：



对于类库的加载，我们就介绍到这里啦，这也是最常用的情形。

## 第九节:总结

\*\*\*\*\*小结\*\*\*\*\*

对于 Yaf 框架来说, 是一个特别典型的轻量级的性能极强的框架, 即使和其他 C 语言编写的框架相比, Yaf 的性能也是其最大的亮点, 也是我们通常选择 Yaf 最重要的原因。

但是需要说明的是, Yaf 并不是一个面面俱到的框架, 它只是帮助我们解决对性能影响较大的那块, 这样的好处是, 既不会增加很多我们根本不需要的功能, 也帮助我们完成了最需要的那部分功能。



## 附录:致敬读者

\*\*\*\*\*梦之都\*\*\*\*\*

- 1.梦之都致力于发布优秀的IT原创教程,而且所有教程都提供免费下载(通常是PDF格式)和在线阅读(通过网页的方式)。
- 2.在前进的道路上,梦之都希望做您最优秀的得力助手。学海无涯,与君共勉。
- 3.梦之都的官方网站为:[www.mengzhidu.com](http://www.mengzhidu.com)。我是辛星,我在这里等你。

\*\*\*\*\*加入我们\*\*\*\*\*

- 1.现在梦之都已经开放注册啦,朋友们可以到梦之都的官网去下载、收藏自己喜欢的教程奥。
- 2.想在前进的道路上找到更多的伙伴,可加PHP技术交流群:459233896。