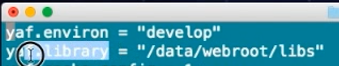
Yaf类库加载规则

Yaf类库加载规则

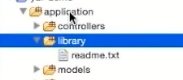
Yaf有全局类库和本地类库

Php.ini的yaf.library指定了全局类库



那么本地类库是可选配置项中的application.library这个选项配置的

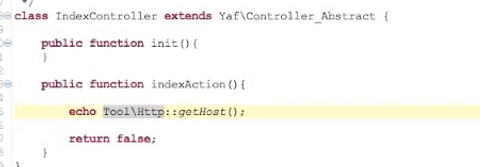


 --- 这个是本地类库

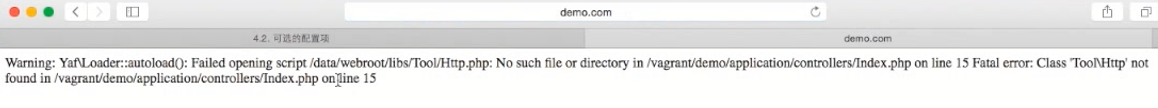
现在使用类库的时候

使用发发

现在使用一个工具类



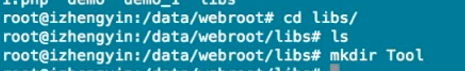
这样去输出主机地址



但是 没有加载到 说明去了全局类库加载文件

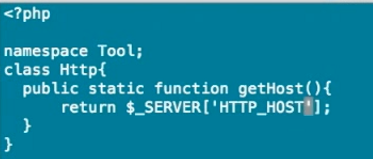
去全局把这个文件建立一下





这样就完成了

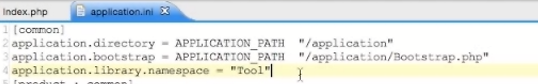
在这里面简历一个http.php





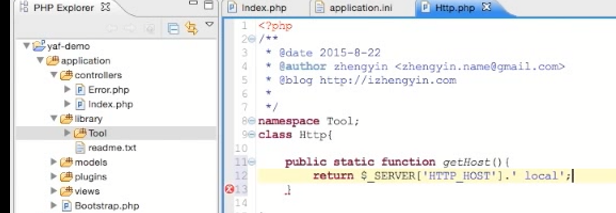
现在这个工具不想放到全局 是本地的

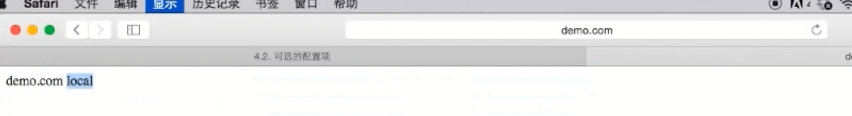
在application.ini中指定命名空间





这次到找

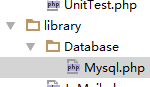




指定多个本地的类库

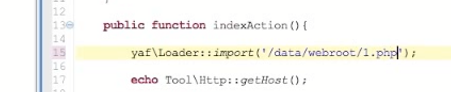


Yaf是使用命名空间加载类库 --- 按照命名空间区分目录

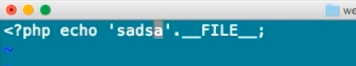
【我们的项目中】

如果本地找不到 就去全局

除此之外 还提供一个类



这个就加在一个指定一个绝对路径的路径



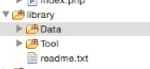
以上是绝对路径去加载

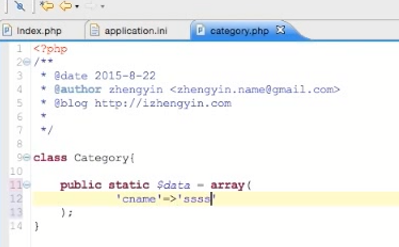
现在 想办法相对路径去加载



Yaf会去本地类库下面的Data目录找coategory.php

这个Data/就相当于命名空间





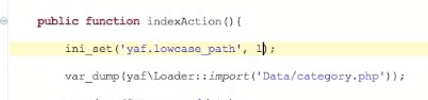


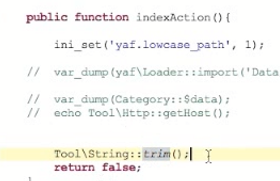


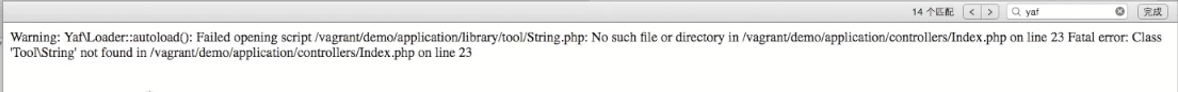
这是手动的加载模式

Yaf还有一个配置









加载额时候



但是 前面的tool是小写的 不是大写的

这个影响了目录文件的大小写问题

通常不开启这个

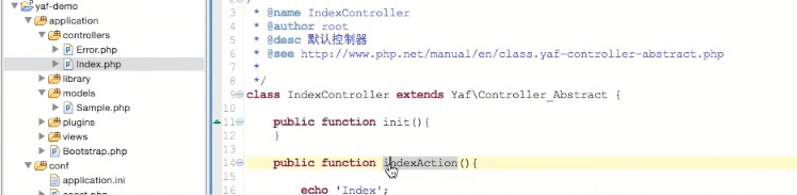
Yaf加载的时候 有一个loader这个类  


【Tool\String.php这个文件中有一个方法叫trim() 🡪 那么就应该调用 Tool\String::trim】

MVC --- model和controller

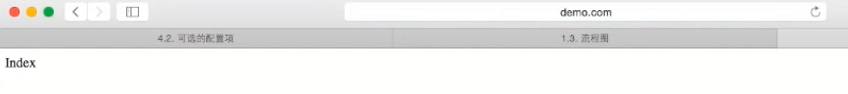
Controller负责接收用户的请求 ---- 请求完成之后

Yaf中如何使用控制器

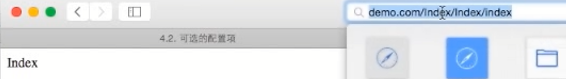


当我们请求对应的控制器的时候

Yaf就会调用IndexController去处理 默认交给indexAction()处理



访问站点的时候 等同于



第一个index代表模块

第二个index代表controller

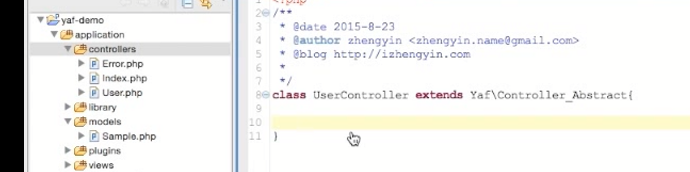
第三个index代表action

所以 假如想访问不同的控制器

demo.com/index/User/index



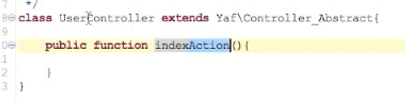
回去找这个User.php



这样就可以找到了

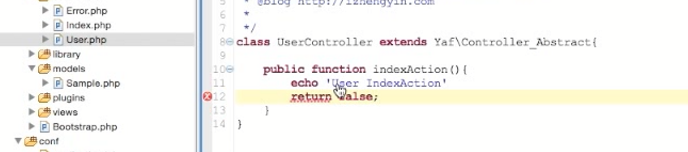


没有找到对应的action

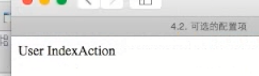


---- 约定 就是控制器以Controller结尾 动作以Action结尾

---- 文件名不需要以controller结尾



Return false不渲染视图



这是控制器的概念

===== 模块的概念

在我们开发app的时候 有很多的功能

有些功能具有相似点 就会放到一个模块下面

模块就是负责处理相关用户逻辑的

对业务分模块

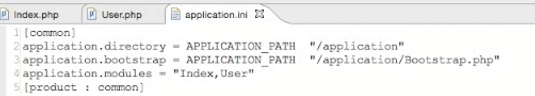
Yaf如何分模块

在application.ini中定义

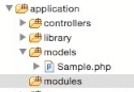


这个里面定义使用了哪些模块

一定要加上index这个模块



在application下面建立一个目录叫modules



在modules下面 建立一个User



建立存放控制器的目录

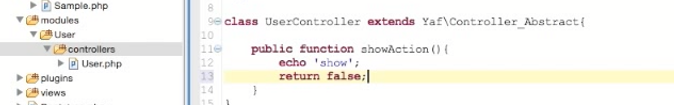


再往下建立控制器文件



这个时候 建立一个User.php

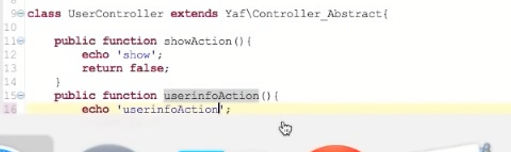
建立对应的action







建立一个方法



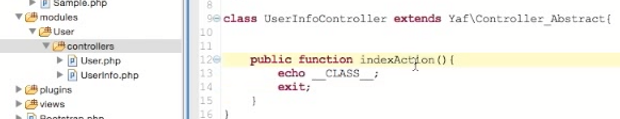
这就是Yaf分模块

【index不需要建立模块 就是默认的模块 不需要放在modules下面 所以 application下面的直接的controllers views等等都属于index这个模块的】

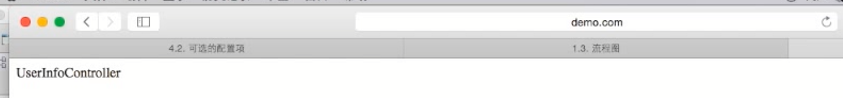
控制器这里面

Yaf对于控制器 不识别驼峰





现在去访问



解释一下

系统是Mac系统 Mac不区分文件名的大小写





但是 找的时候 没有找SignUp 找的是Signup

但是 生产环境Linux是区分大小写的

所以 我们的只是首字母大写 剩下的都是小写

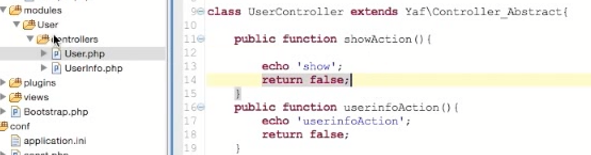
======

Yaf路由

路由 就是把url通过定义的规则发送到预先定义的模块 动作的过程

Yaf可以采取这种形式

之前写了一个模块



访问这个showAction



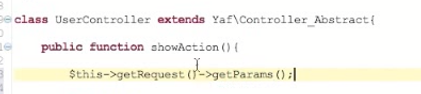
现在

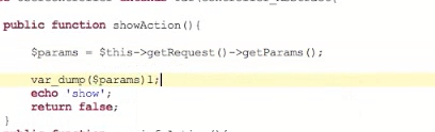


Yaf就是通过这个去路由的

这个是模块 控制器 和 action

下面是url中参数的获取







这个是默认的路由形式

Yaf支持的五种路由形式



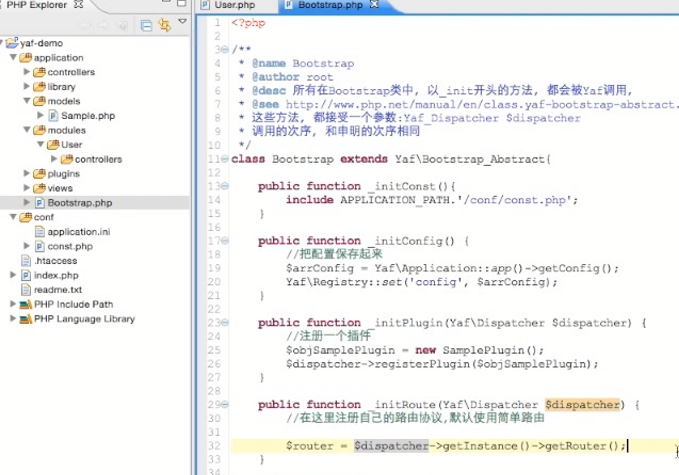
说一个simple和Regex

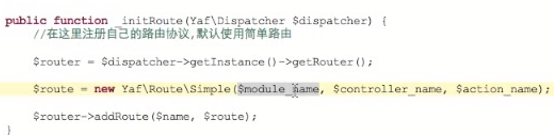
--- 如何添加一个路由？

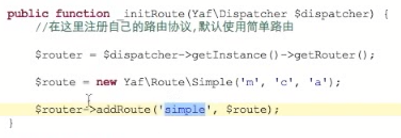
在哪里添加？



可以通过Yaf\_Dispatcher来获取路由对象





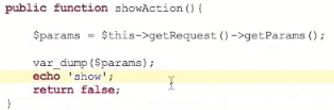


Bootstrap会自动执行上面的代码

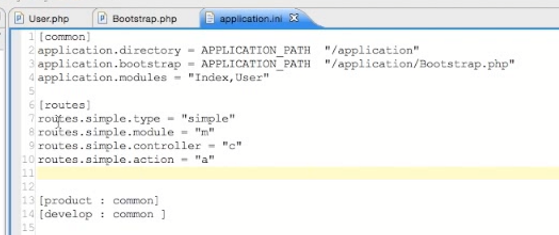
添加完成之后



这样这里面

这个就被访问到了

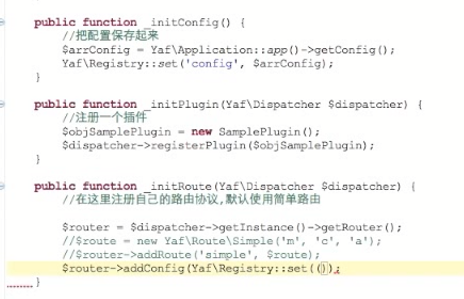
Yaf还支持通过配置文件添加路由



加到环境里面 --- 因为php.ini当前的环境里面是develop 所以 直接添加到develop中就可以了



在Bootstrap中 通过

 ---- 在\_initConfig中设置了这个config



在下面就直接get出来

同样的方式







就正常了

这就是通过配置文件增加路由

还有一个正则路由

现在显示用户资料 为用户提供一个更友好的url



这个会路由到showAction 并且传入一个参数 1001

这个借助到Yaf的正则路由

先指定一个路由的名称

##作为正则的边界符

以/开始 因为是匹配的request uri

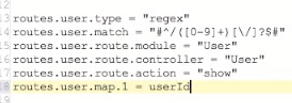
因为要把这个uri拿出来

所以 把这个对象拿出来 ()扩上

最后 可能有斜杠 也可能没有 也就是0个或者1个



是元组匹配到的内容 是第一个()中的 所以 是1 不是0 0是匹配整个的



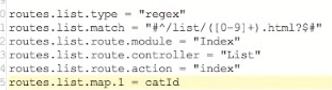


这个是Yaf的正则路由

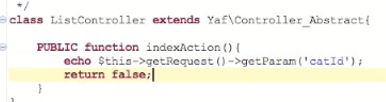
可以定义很多规则

做一个列表的展示 做成伪静态



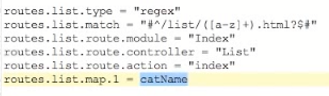








这样就可以指定规则了

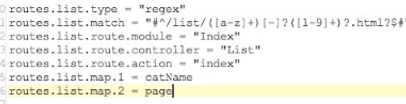


列表可能有分页

规则是 

php-1.html表示第一页

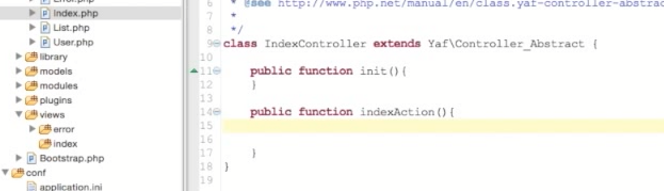






=========== 视图 =============

视图的渲染



直接访问

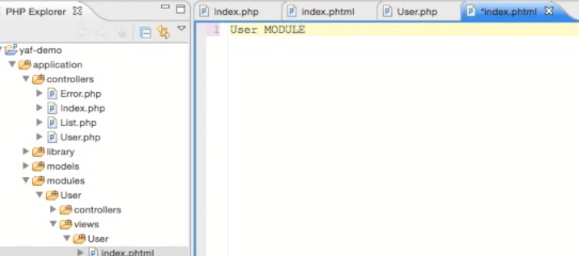


Yaf默认开启自动渲染

所以 会按照控制器 会找views/index/index.phtml

如果是模块





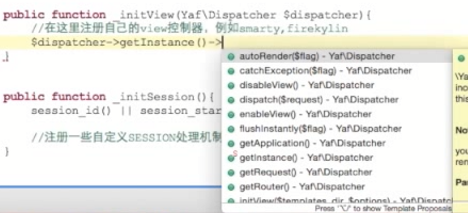
这个是yaf自动渲染的规则

如果不想渲染 就在controller中return false

或者在全局中进行禁用

需要通过Dispatcher中进行禁用

在Bootstrap中进行禁用

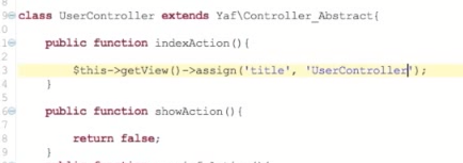
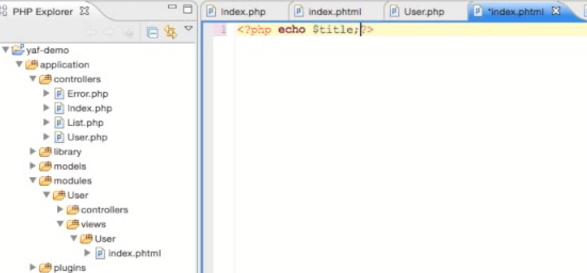




这样就不会载入视图

除了这个意外 还有试图方法

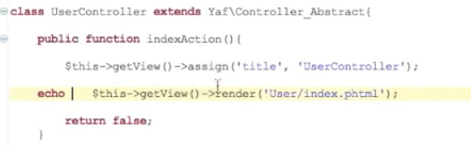


这就是分配变量 分配之后 就可以在页面中直接使用



render是拿到解析后的内容



最后关闭了自动渲染

这样可以通过规则做一些静态

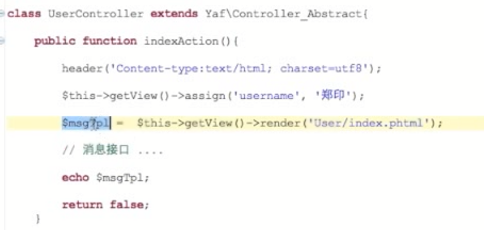
静态这个东西 有点长 这里面不牵扯这么多

可以通过render可以载入html的模板文件或者静态



这个模板有个特点 有个变量

这样 可以单独放到一个地方 通过render载入进来

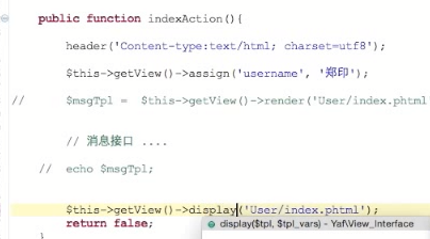




还有一个display

没有return false

实际上controller相当于执行的代码是



显示是没有变化

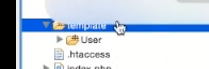
Return false 就不会显示模板了

Yaf可以更改载入的路径





出错了

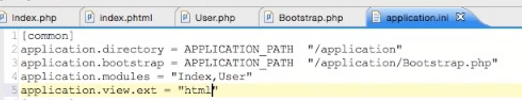


就可以了

Yaf的后缀



可以修改





这样就报错了



这里面找的index.html