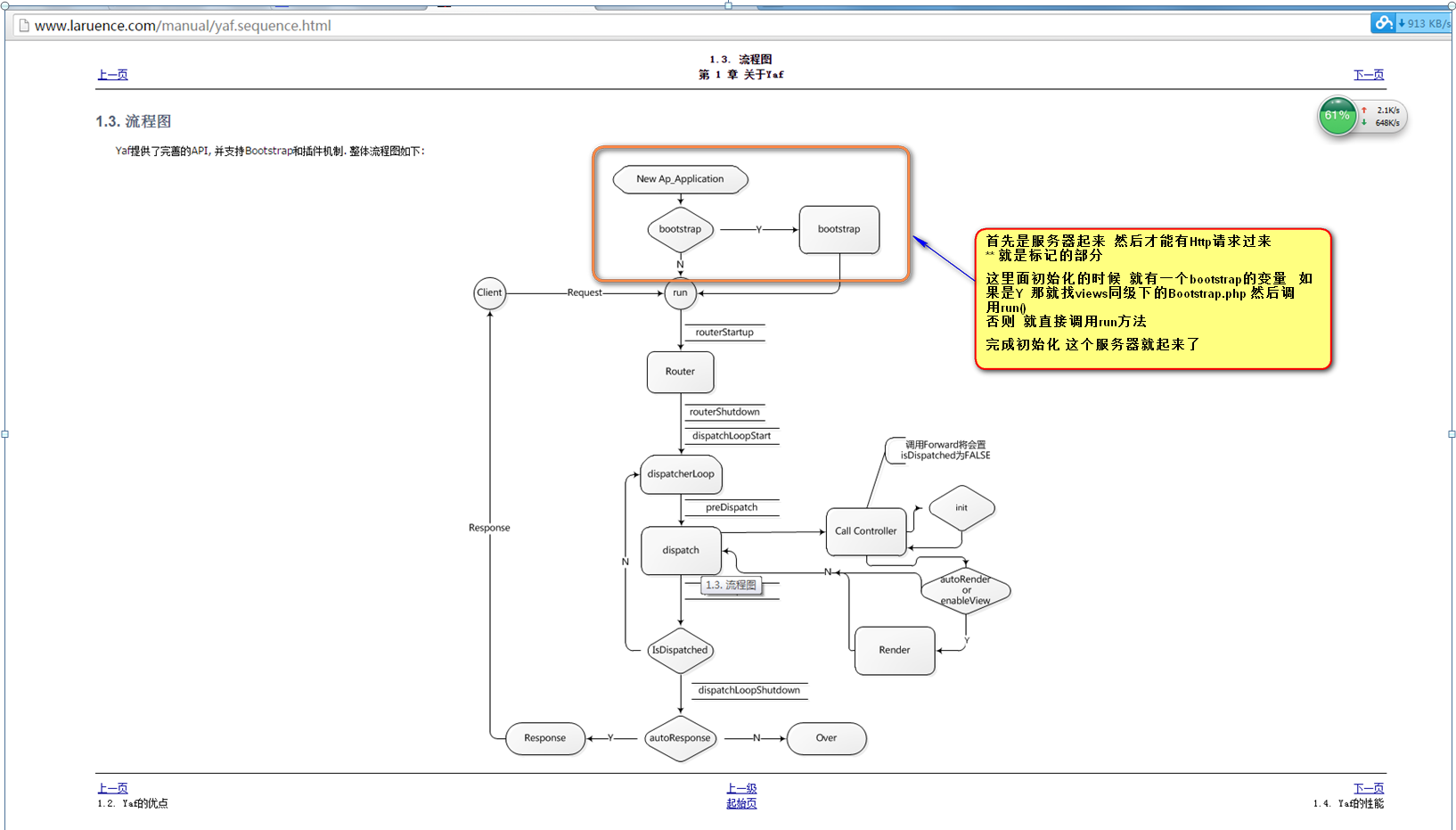
Yaf的运行流程和相关模块

了解一下Yaf的运行过程

官方书册 有一个流程图

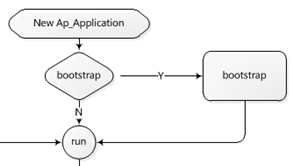




首先Yaf单入口

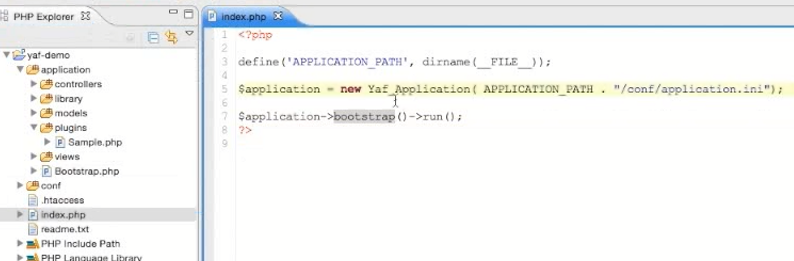
首先New 

创建一个这个Ap\_Application之后 下面就有可能 就是 执行**BootStrap还是不执行**



对应到代码中

在index.php中 就会



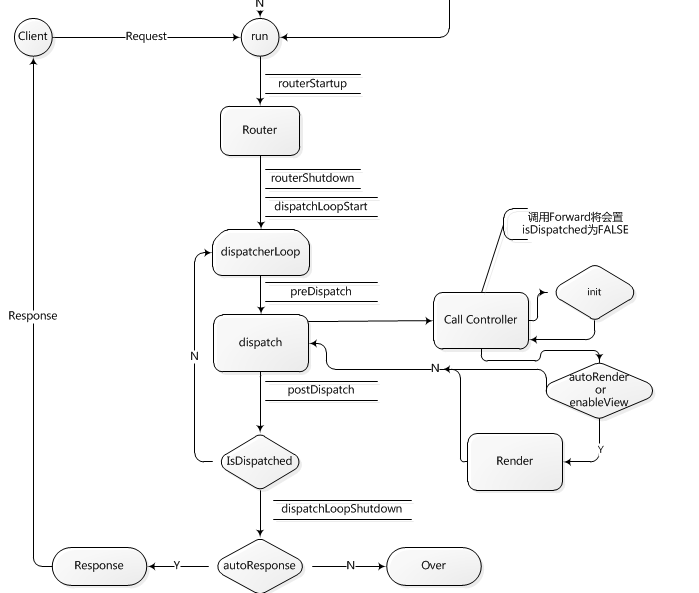
这里面就会有一个 **new Yaf\_Application**这样的代码**【我们项目中叫Yaf\Application】**

然后就选择是否运行bootstrap

如果选择了bootstrap文件 就会到views目录下面找bootstrap文件（同级）



初始化完成就可以处理客户端的请求

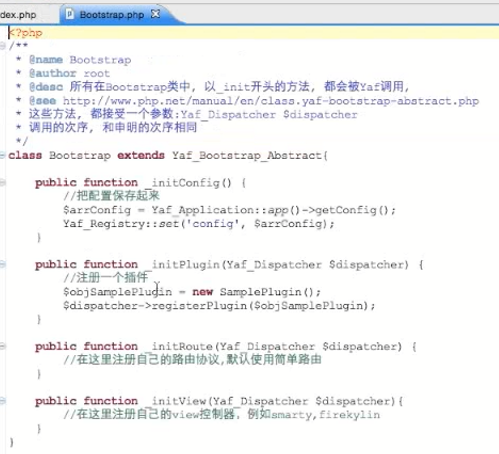


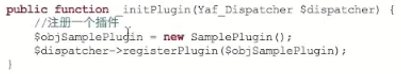
看到箭头的方向 就可以知道是client客户端 向这个发起了请求

然后交给router 会进行routerStartup的hook进行回调 hook是yaf的插件

查看一下这个插件

这个插件通常会在bootstrap里面定义

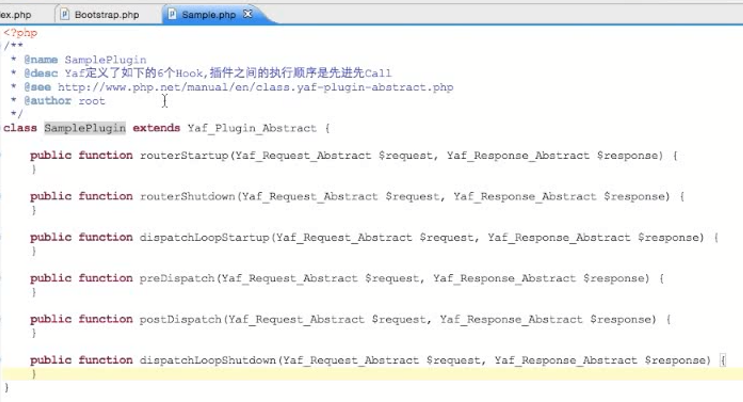




这个**SamplePlugin**的就是插件

所以 可以在这里面定义插件

这个插件中有6个方法



【可以看到 这个SamplePlugin里面 定义的六个方法 都是刚才request过来之后 的所有流程

】

路由开始 路由结束 循环调度开始 循环调度结束 调度前 调度后

正常情况下 就是routerStartup和routeShutdown调度完成 就直接结束了

之后就可以

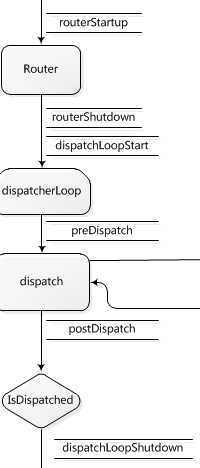
这样 就可以在routerStartup里面就可以验证登录了

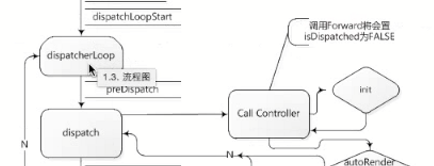
可以在这里面验证session

接着看

循环调度开始 就执行一个



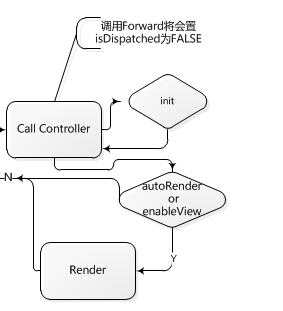
 可以看到 这个图 这样 这六个hook就连续这样被调用 可以在每一个hook中添加自己的想要嵌入的内容



然后就开始了dispatcherLoop 这是一个单个的循环调度

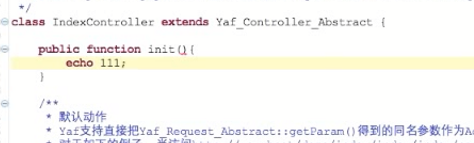
然后就给出路由过后的结果

然后根据这个路由过后的结果 去调度（dispatcher）模块控制器 然后去调用具体的模块中的Controller

然后调用这里面的init方法

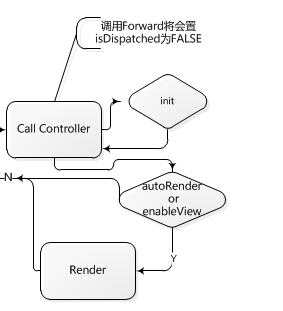
默认会路由到controller下面的index.php里面

如果 你有init方法 会先执行你的init方法





发现这个先执行了

Init执行完  可以看到 这个箭头返回到Call Controller 也就是执行具体的controller中的action方法

这里面就是执行我的indexAction



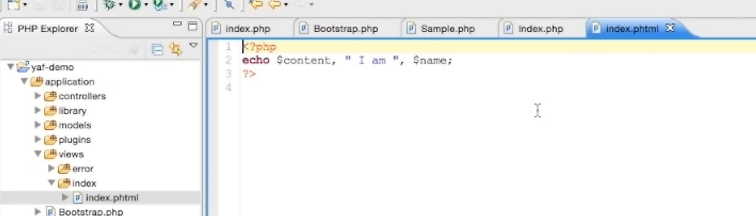
就是这个过程

然后 有一个AutoRender

开启 或者 进入视图

如果了开启了自动渲染 就会调用Reader

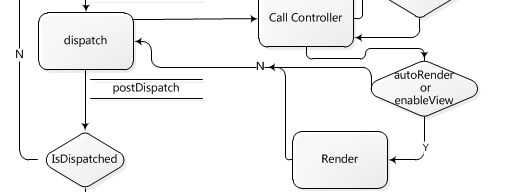
默认情况下就会调用views/${php主文件名}/${php主文件名}.phtml这个文件



那么这里面因为调用的是indexAction() 那么 就调用views/index/index.phtml这个视图

这个是自动加载Reader的过程

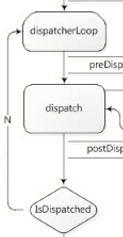
然后就返回Dispatcher



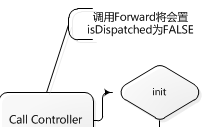
单次调度结束了

因为外面是dispatcherLoop ---- 循环调度

所以 要看你真的是否调度完成 所以 会检查IsDispatched这个标志位

如果没有 就继续调度

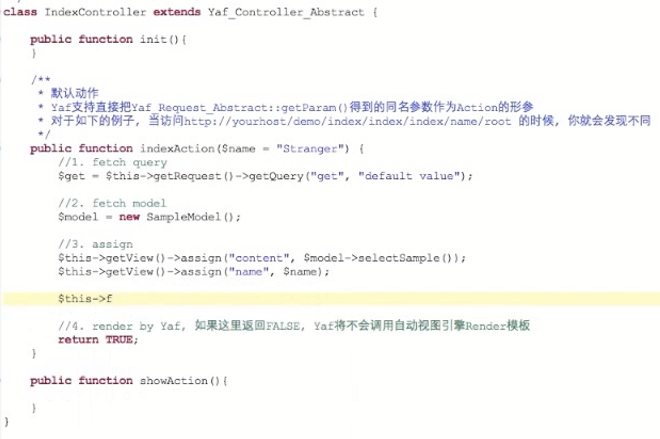
什么时候dispatcher没有完成呢？就是forward的时候【转发的时候】

 ---- 置为false

这样就会继续调度

比如有一个方法

在IndexController中添加一个showAction()





这样 就会输出

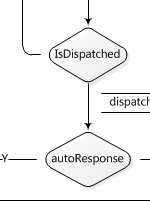


这样 showAction中的内容 就会被转发到showAction中 这样 打印出show 打印完成 因为返回时false 根据刚才的逻辑图 会继续进行dispatcherLoop

这里面进行了两次调度

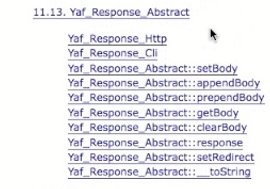
这就是整个循环调度

isDispatchered为true的时候

最后 ---- autoResponse 如果这个是yes

就会返回给客户端

Yaf有一个响应的类 叫YafResponse类



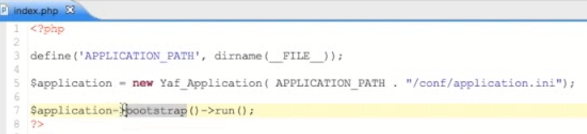
以上是Yaf的自动执行过程

这里面没有看到model model是自己抽象的东西

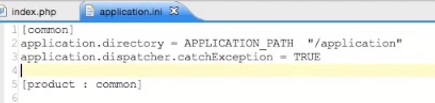
Yaf配置类model的加载规则 但是 运行的时候 没有model的体现

Yaf的Bootstrap

可以选择运行还是不运行

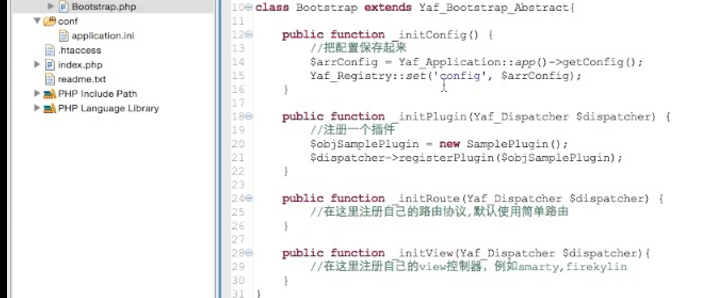


如果选择运行boostrap 就会执行Bootstrap.php 这个文件路径的定义是在/conf/application.ini这个文件中





可以配置application.bootstrap来覆盖默认位置



这Bootstrap.php实际上就是一个类 会从上到下依次调用\_init开头的这四个方法

所以 这个bootstrap是在路由开始之前 就运行了【前面的图也说明了】

所以 在这个bootstrap中可以做很多初始化的东西

分别是 初始化配置 初始化插件【注册一个插件】 初始化自己的路由协议 初始化视图控制器【定义自己的视图 注册session 定义自己的常亮】

这里面举个例子

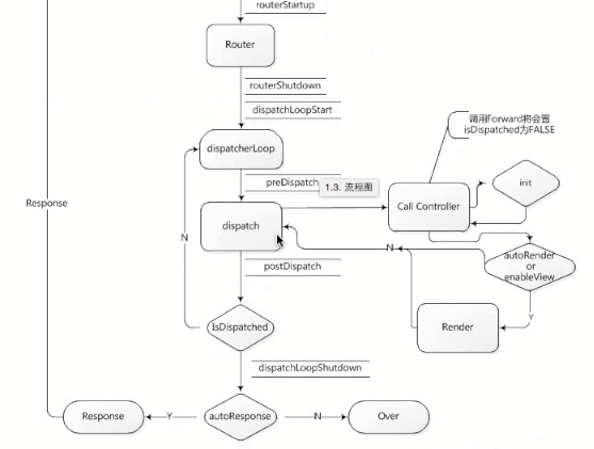




这里面接受一个对象

因为bootstrap是在路由开始之前

可能用到很多的对象

 这个图中的对象都可能被用到

看下Yaf\_Dispatcher

 ---- 这里面有很多方法 可以被调用

Bootstrap就是这样一个过程

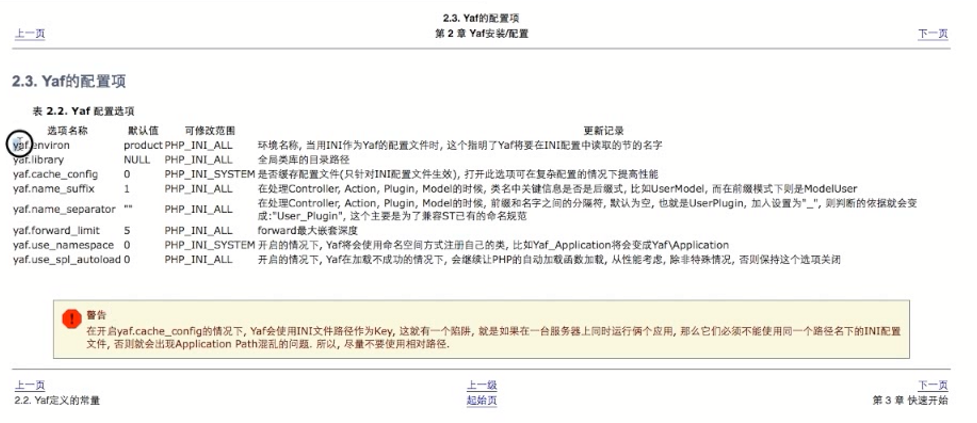
通过Yaf\_Dispatcher可以完成一些初始化的工作

Yaf的配置

Yaf的官方手册

\*\*\*下面是Yaf.ini的配置







这个可以做一个环境的区分

比如本地 修改为dev 线上就是prod



共用的代码库就放在yaf.library



一般打开

常常使用后缀





前面的一个dispatcher有一个forward 一般来 说最大的嵌套



Php5.3支持命名空间



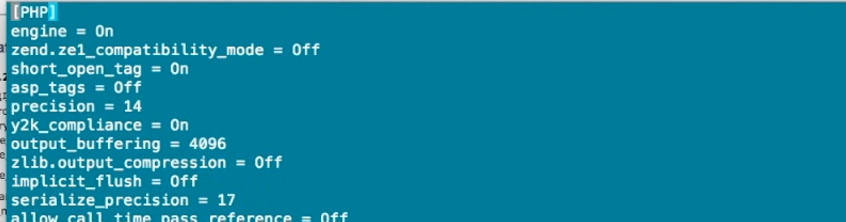
自动加载的类 一般不用 这个框架就是在yaf上面自己做的延伸

这个放到虚拟上面做一个配置

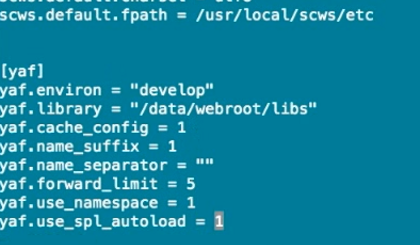


切换到root下面





做一个配置



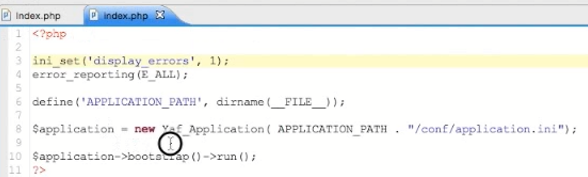


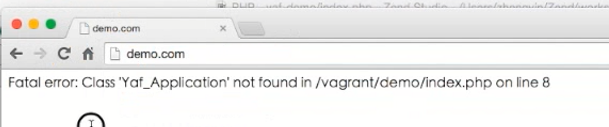
这样就生效了

因为开启命名空间之后

但是 我们的代码是没有命名空间

打开错误





这个地方说这个类不存在

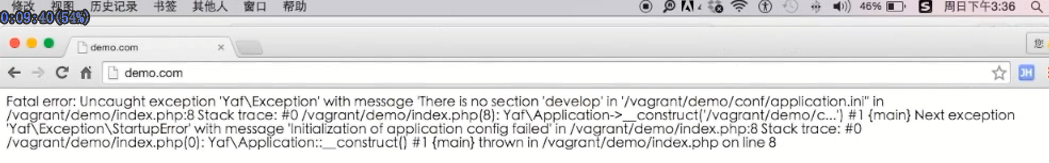
可以按照手册上面去修改



修改之后 就是Yaf\Application

基本上是这样的修改套路

这样做 还需要修改一个地方

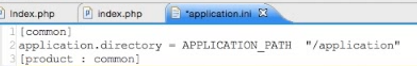


说找不到一个develop



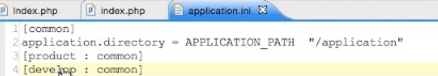
这个会直接影响到application.ini中的配置

\*\*\*下面是application.ini的配置



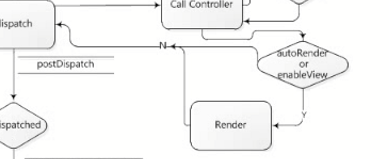
这是默认的

我们需要添加一个键值对





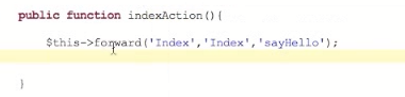
这里面抛出了一个未定义变量



会去渲染对应的视图之后 回到dispatcher

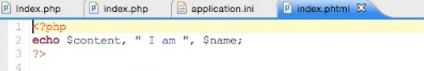
然后继续分发

这是因为



这个indexAction中 转发到index模块从index转发到sayHello中

但是 我们的index.phtml中



没有$content和$name这两个错误

我们可以return false不渲染模板【后面再说】



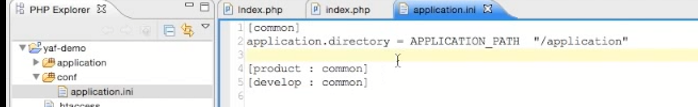
所以



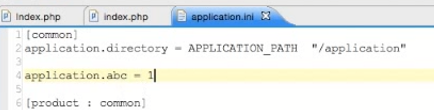
修改了对应的命名空间

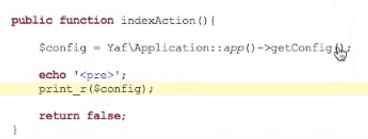
Yaf\

\*\*再看application.ini的配置 这个配置遵循了php.ini的语法



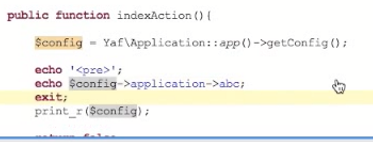
这个里面 可以添加东西



在我们的

拿到这个对象 打印一下





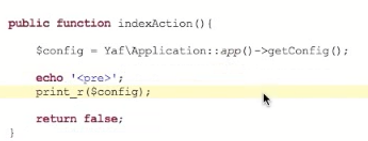
打印出1

还支持配置对

比如

配置一些redis的信息



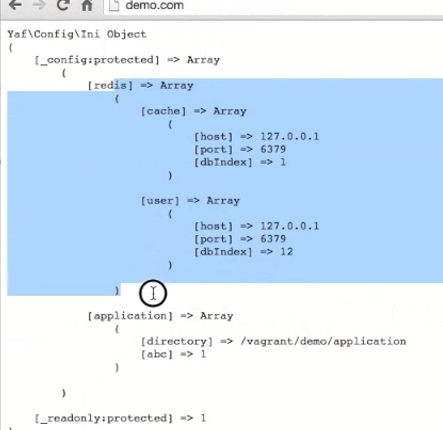




但是 没有看到这个redis的配置对

要添加一下





这样就有了

【这就相当于 原来是common加入到develop中 就写成 [develop:common] 然后再为develop加入redis 携程[develop:common:redis]】

但是我们的product仅仅加载了common 没有加载redis

这样不同环境运行不同的配置

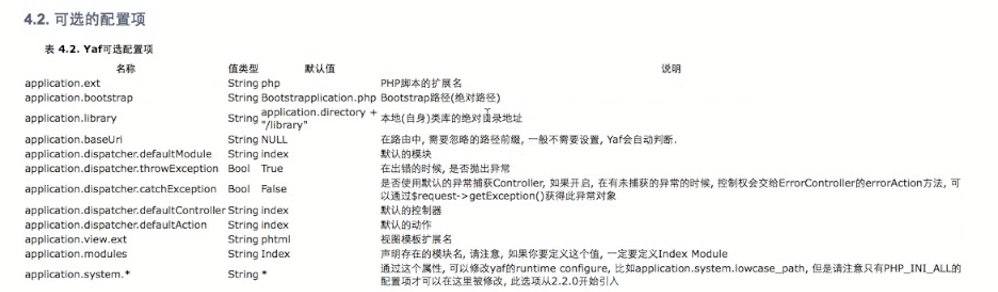
也可以写多个ini文件对应多个配置环境

Yaf也提供了别的可选配置

看下手册







一个是必选的配置项 也就是application.directory 下面的是可选的配置项