三满百步网设框架（Bible Web Design Frame）

cai_yankun

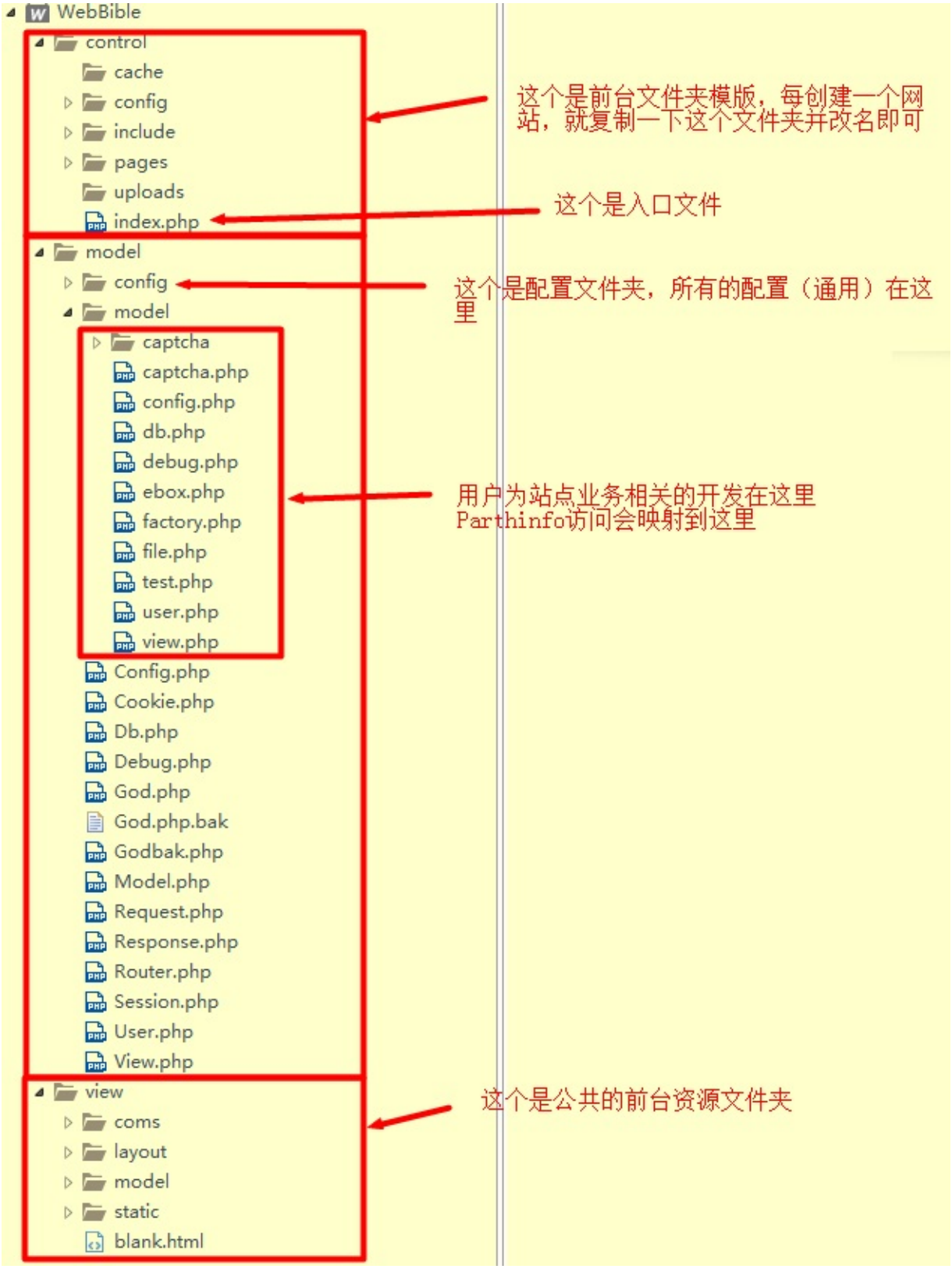
目　录

框架简介

框架的作用和特点是：

1. 轻量级，惰性加载；
2. 不仅包括后端（LAMP），也包括了前端的设计（JS，Ajax，工厂机制），以及前后端的交互机制，是一套前后端拉通的快速开发网站的思路和工具；
3. 彻底实现了前后端分离，降低了服务器的运行负荷，简化了后台PHP的复杂度，并且实现了数据访问的效率以及后期维护的便捷
4. 支持一套框架同时驱动多个独立的站点，不同的站点之间实现灵活的共享、独立；
5. 支持框架项目与普通项目的无缝结合，能将框架快速的集成到一个非框架类项目中进行快速开发；也能将一个非框架类的项目快速适配到框架项目中继续进行开发；
6. 重定义了MVC架构，将控制功能的实现前移，移至js，目的是简单，高效；
    - Model类封装了框架固有的一些类（配置，SESSON，COOKIE，DB），实现了Pathinfo到PHP函数映射，权限处理等问题；
    - Controller为逻辑业务的实现，不在PHP中实现，前移到前端JS中实现；简化后台PHP为数据流通管道功能；
    - View类封装了Pathinfo到前台展示页面的映射，以及前台代码相关的的功能；

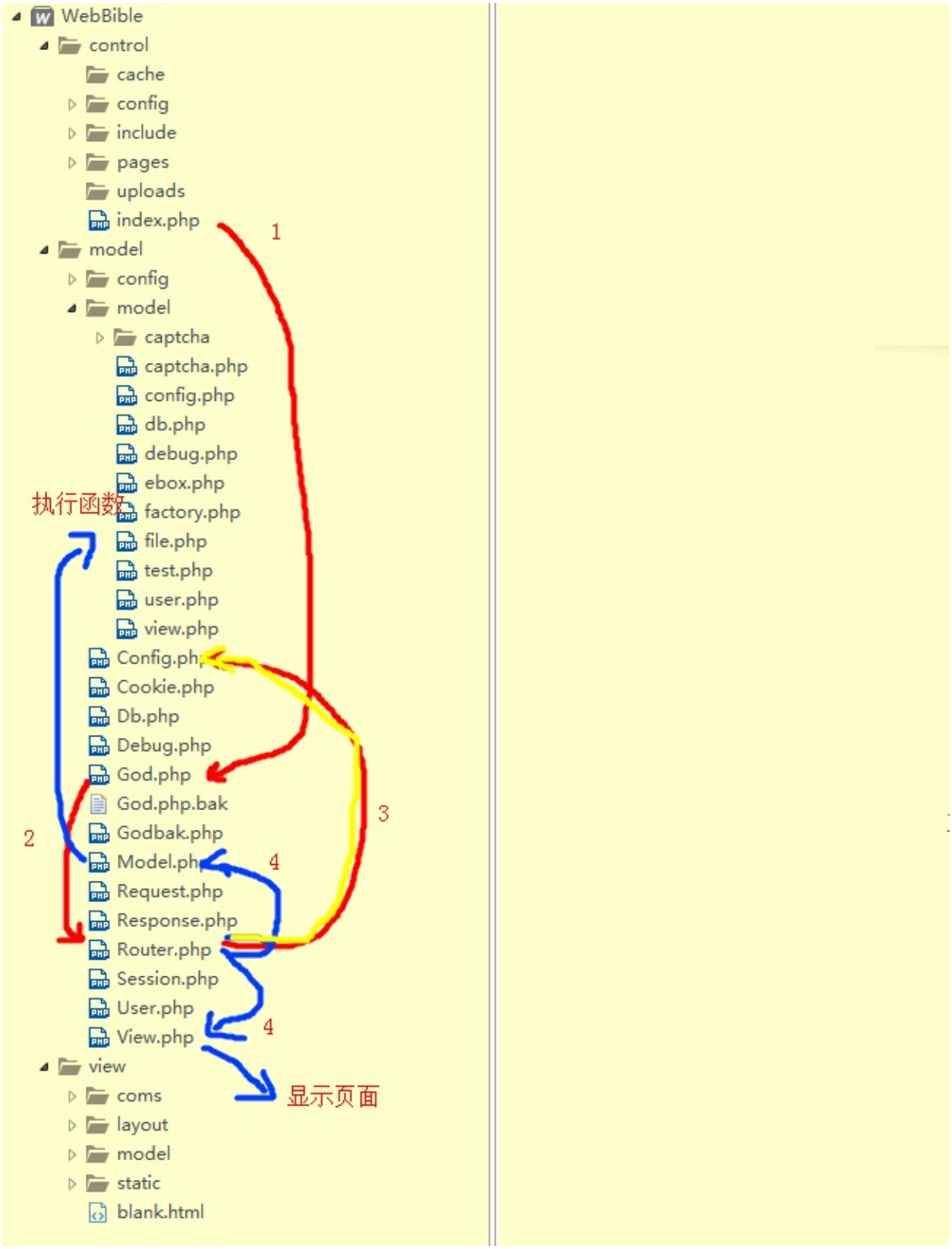
框架原理



- Model类封装了框架固有的一些类（配置，SESSON，COOKIE，DB），实现了

Pathinfo到PHP函数映射，权限处理等问题；

- Controller为逻辑业务的实现，不在PHP中实现，前移到前端JS中实现；简化后台PHP为数据流通管道功能；
- View类封装了Pathinfo到前台展示页面的映射，以及前台代码相关的的功能；



运行流程：地址栏输入成功进入到入口函数以后，首先加载

`God.php  /*启动惰性加载功能，然后直接调用router进行分发*/`

Router.php

框架的安装

安装要求

PHP版本>5

安装步骤

解压到服务1器文件夹

https://caiyankun@github.com/caiyankun/webbible.git

配置单站点apache:

`/etc/httpd/conf/httpd.conf`

```
DocumentRoot "/var/webbible/control"
```

配置多站点apache：

`/etc/httpd/conf/httpd.conf`

```
<VirtualHost *:80>
    ServerAdmin cai_yankun@qq.com
    DocumentRoot /var/webbible/site1.com
    ServerName site1.com
    ErrorLog logs/more.sanmantech.com-error_log
    CustomLog logs/more.sanmantech.com-access_log common
</VirtualHost>
<VirtualHost *:80>
    ServerAdmin cai_yankun@qq.com
    DocumentRoot /var/webbible/site2.com
    ServerName site.com
    ErrorLog logs/more.sanmantech.com-error_log
    CustomLog logs/more.sanmantech.com-access_log common
</VirtualHost>
```

开通地址栏重写：

`/etc/httpd/conf/httpd.conf`

```
LoadModule rewrite_module modules/mod_rewrite.so
```

配置Apache用户/PHP脚本的可写权限：

在webbible目录下

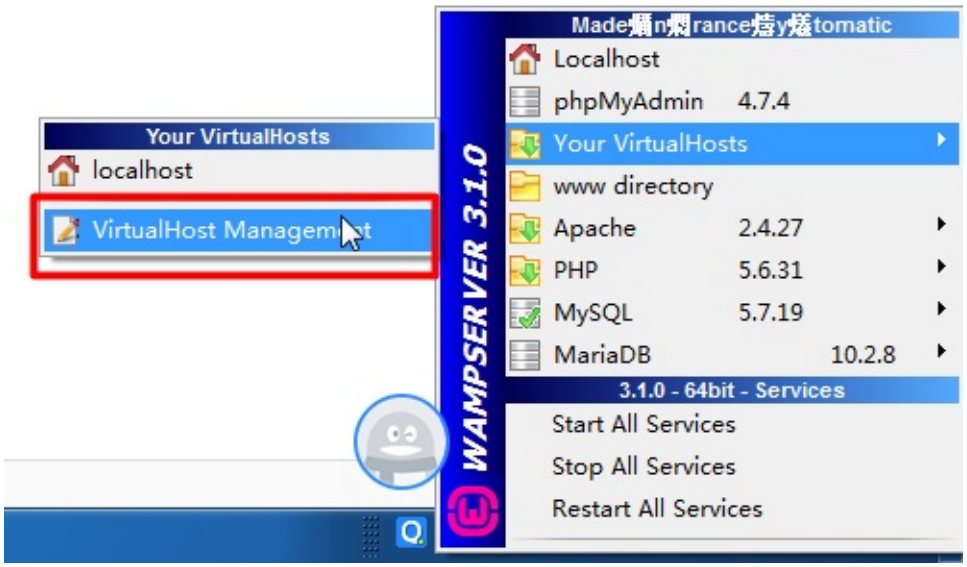
```
chown apache:apache -R
ls-l
```

安装测试：

浏览器输入`www.site1.com`

本地PHP服务器环境快速搭建

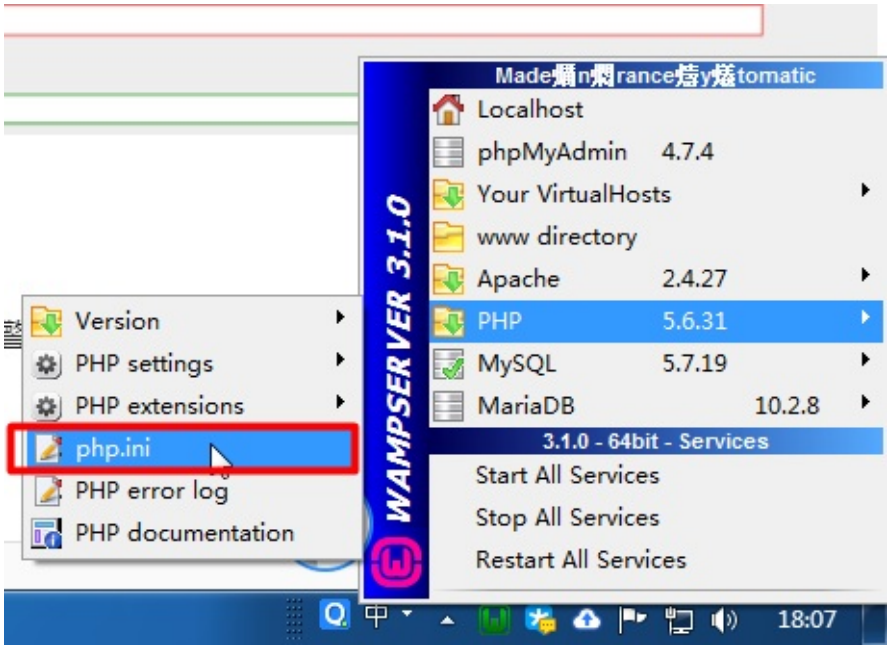
- 下载WampServer64，安装并运行；
  如果出现运行时错误（缺少.dll）就先卸载，并安装相应的VC++ redistribute
  120 对应的是2013版本
  140对应的是2015版本
  100对应的是2012版本
  每个版本都要装
- 创建虚拟主机：



填写需要使用的域名，以及项目所在的路径，路径中好像

- 修改PHP报错级别

新版本的PHP报错级别很低，可以设置忽略告警信息：



搜索 error_reporting

设置error_reporting=E_ERROR | E_PARSE

```
如果参数 level 未指定，当前报错级别将被返回。下面几项是 level 可能的值：
  值      常量                    描述
  1       E_ERROR                 致命的运行错误。错误无法恢复，暂停执行脚本。
  2       E_WARNING               运行时警告(非致命性错误)。非致命的运行错误，脚
本执行不会停止。
  4       E_PARSE                 编译时解析错误。解析错误只由分析器产生。
  8       E_NOTICE                运行时提醒(这些经常是你代码中的bug引起的，也可
能是有意的行为造成的。)
  16      E_CORE_ERROR            PHP启动时初始化过程中的致命错误。
  32      E_CORE_WARNING          PHP启动时初始化过程中的警告(非致命性错)。
  64      E_COMPILE_ERROR         编译时致命性错。这就像由Zend脚本引擎生成了一个
E_ERROR。
  128     E_COMPILE_WARNING       编译时警告(非致命性错)。这就像由Zend脚本引擎
生成了一个E_WARNING警告。
  256     E_USER_ERROR            用户自定义的错误消息。这就像由使用PHP函数trig
ger_error（程序员设置E_ERROR）
  512     E_USER_WARNING          用户自定义的警告消息。这就像由使用PHP函数trig
ger_error（程序员设定的一个E_WARNING警告）
  1024    E_USER_NOTICE           用户自定义的提醒消息。这就像一个由使用PHP函数t
rigger_error（程序员一个E_NOTICE集）
  2048    E_STRICT                编码标准化警告。允许PHP建议如何修改代码以确保
最佳的互操作性向前兼容性。
  4096    E_RECOVERABLE_ERROR     开捕致命错误。这就像一个E_ERROR，但可以通过用
户定义的处理捕获（又见set_error_handler（））
  8191    E_ALL
```

- 初始化Mysql密码：

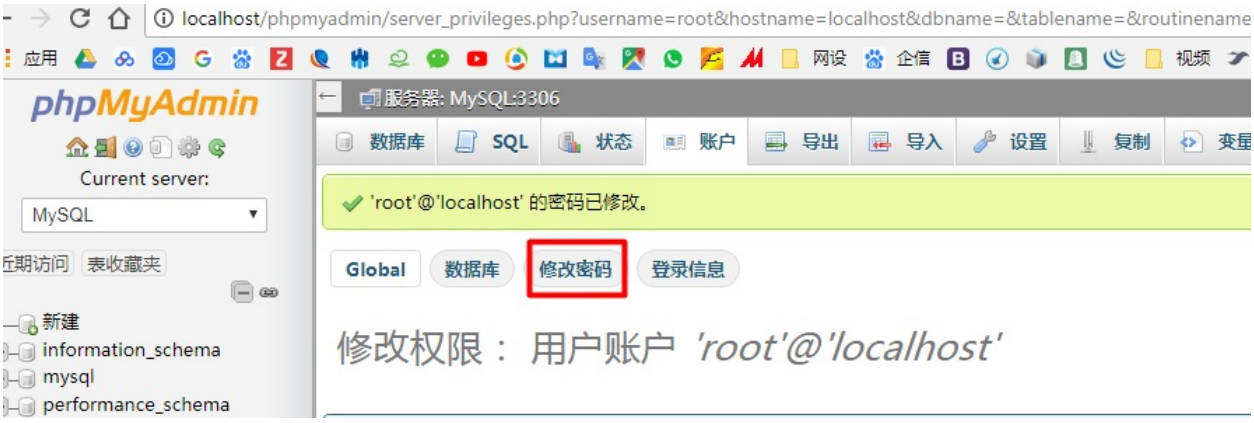
刚安装时PHPmyadmin可用root+空密码即可登录

如果要创建密码，请先登录console

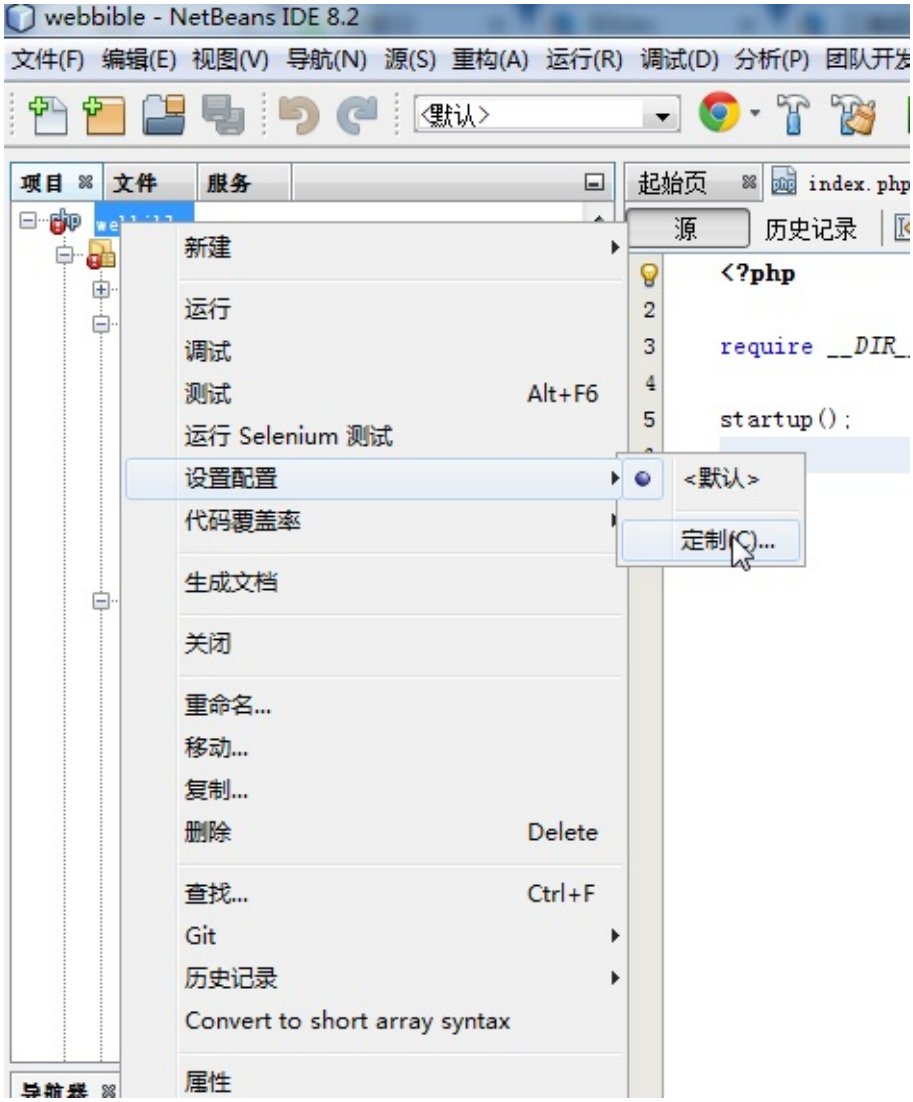
```
USE mysql;
update user set password=“1234Abcd!” where user=’root’;
flush privileges;
quit;
```

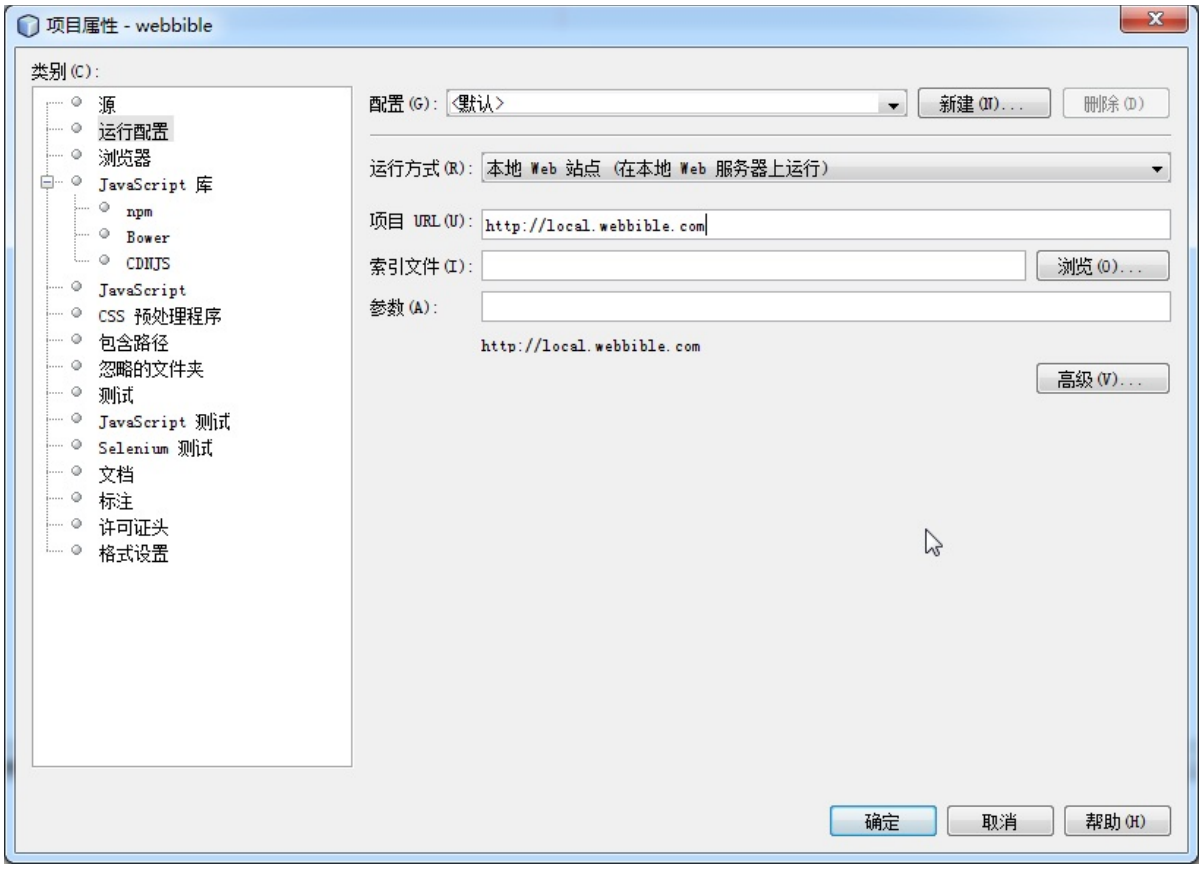
重启mysql服务

或者在phpmyadmin中直接更改



- 重启所有服务；
- 设置Netbeans 调试服务器网址；

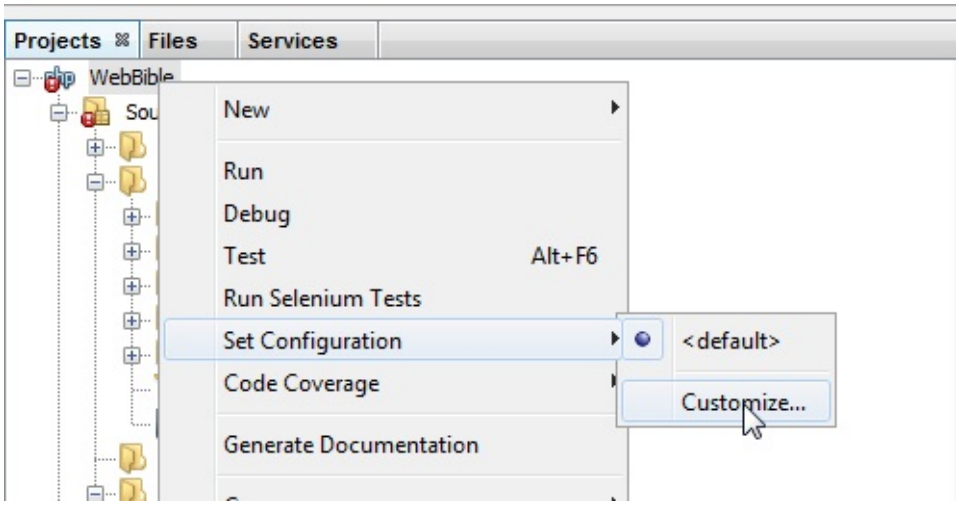

webbible - NetBeans IDE 8.2
文件(F) 编辑(E) 视图(V) 导航(N) 源(S) 重构(A) 运行(R) 调试(D) 分析(P) 团队开发
<默认>
项目
文件
服务
起始页
index.php
源
历史记录
<?php
require __DIR_
startup();
新建
运行
调试
测试 Alt+F6
运行 Selenium 测试
设置配置
代码覆盖率
生成文档
关闭
重命名...
移动...
复制...
删除 Delete
查找... Ctrl+F
Git
历史记录
Convert to short array syntax
属性
<默认>
定制(C)...

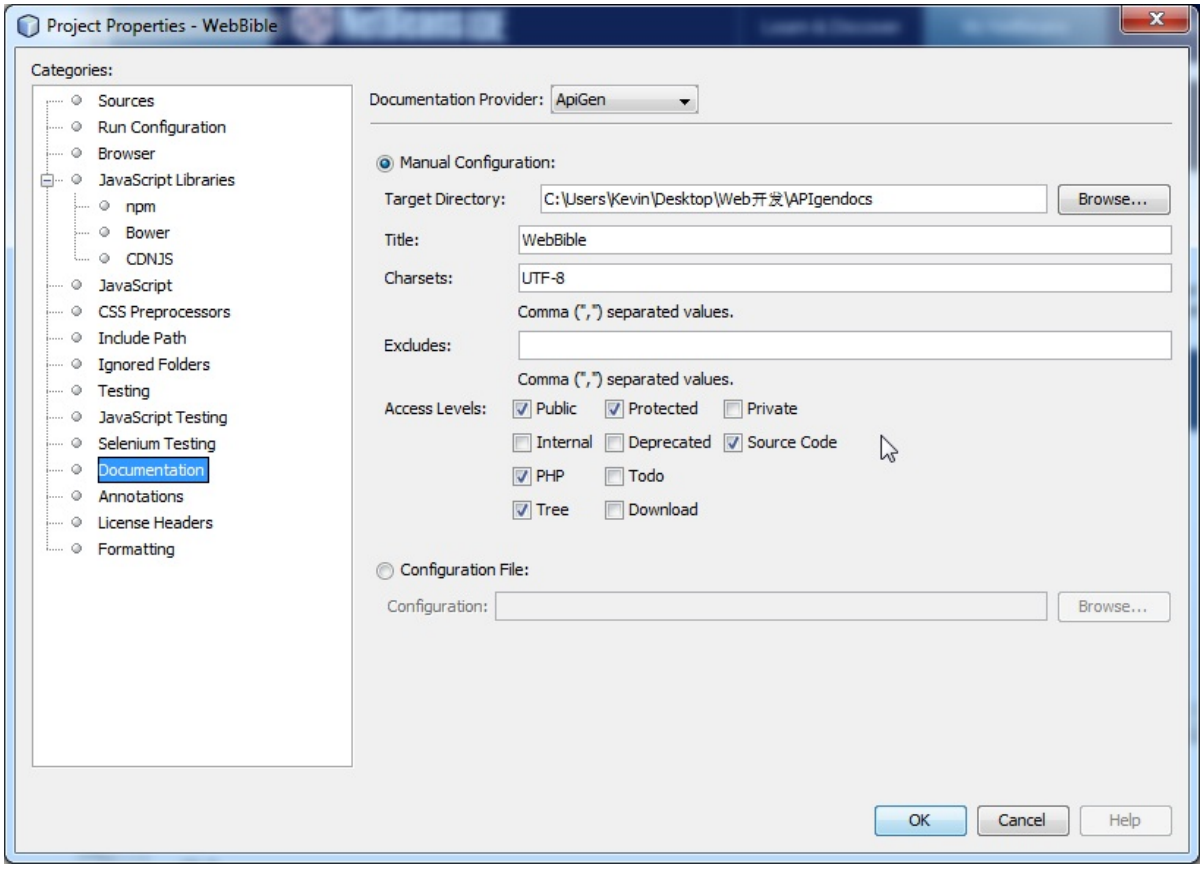

项目属性 - webbible
类别(C):
源
运行配置
浏览器
JavaScript 库
npm
Bower
CDNJS
JavaScript
CSS 预处理程序
包含路径
忽略的文件夹
测试
JavaScript 测试
Selenium 测试
文档
标注
许可证头
格式设置
配置(G): <默认>
新建(N)...
删除(D)
运行方式(R): 本地 Web 站点 (在本地 Web 服务器上运行)
项目 URL(U): http://local.webbible.com
索引文件(I):
浏览(O)...
参数(A):
http://local.webbible.com
高级(V)...
确定
取消
帮助(H)

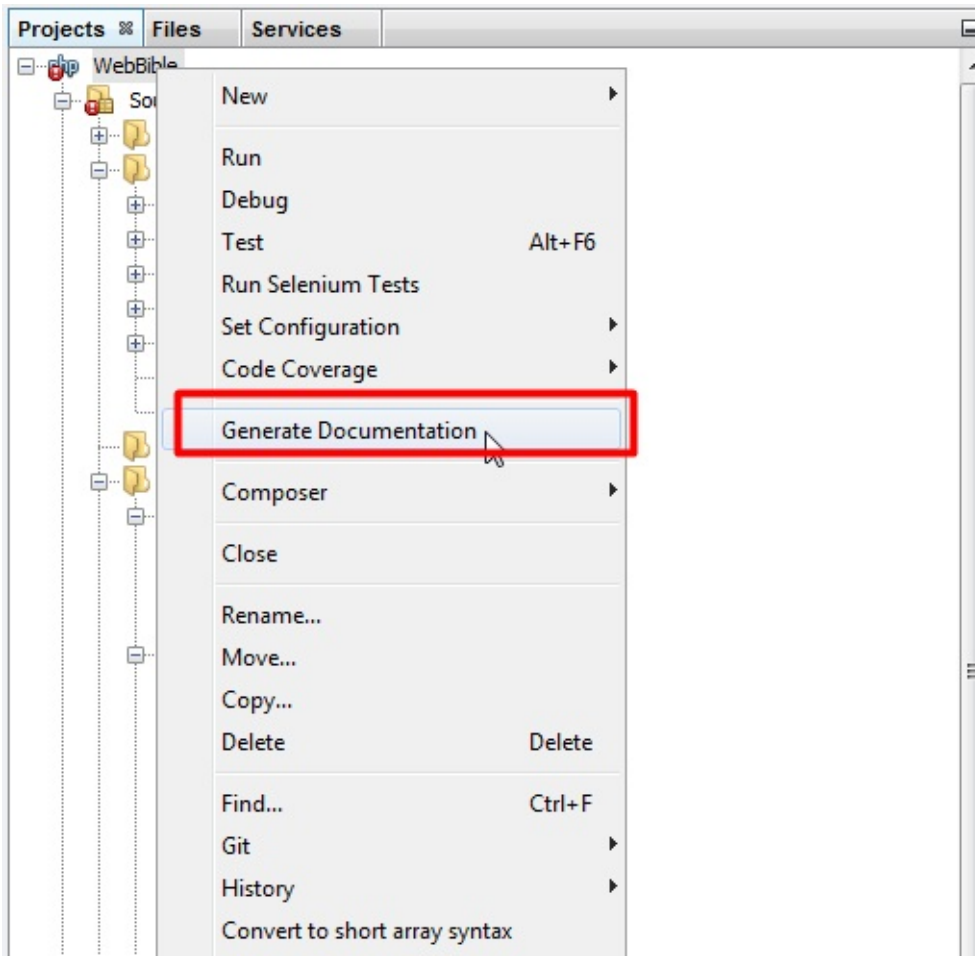
Netbeans开发工具配置

- 安装Netbeans
- 安装有用的插件
  php
  composer
  git

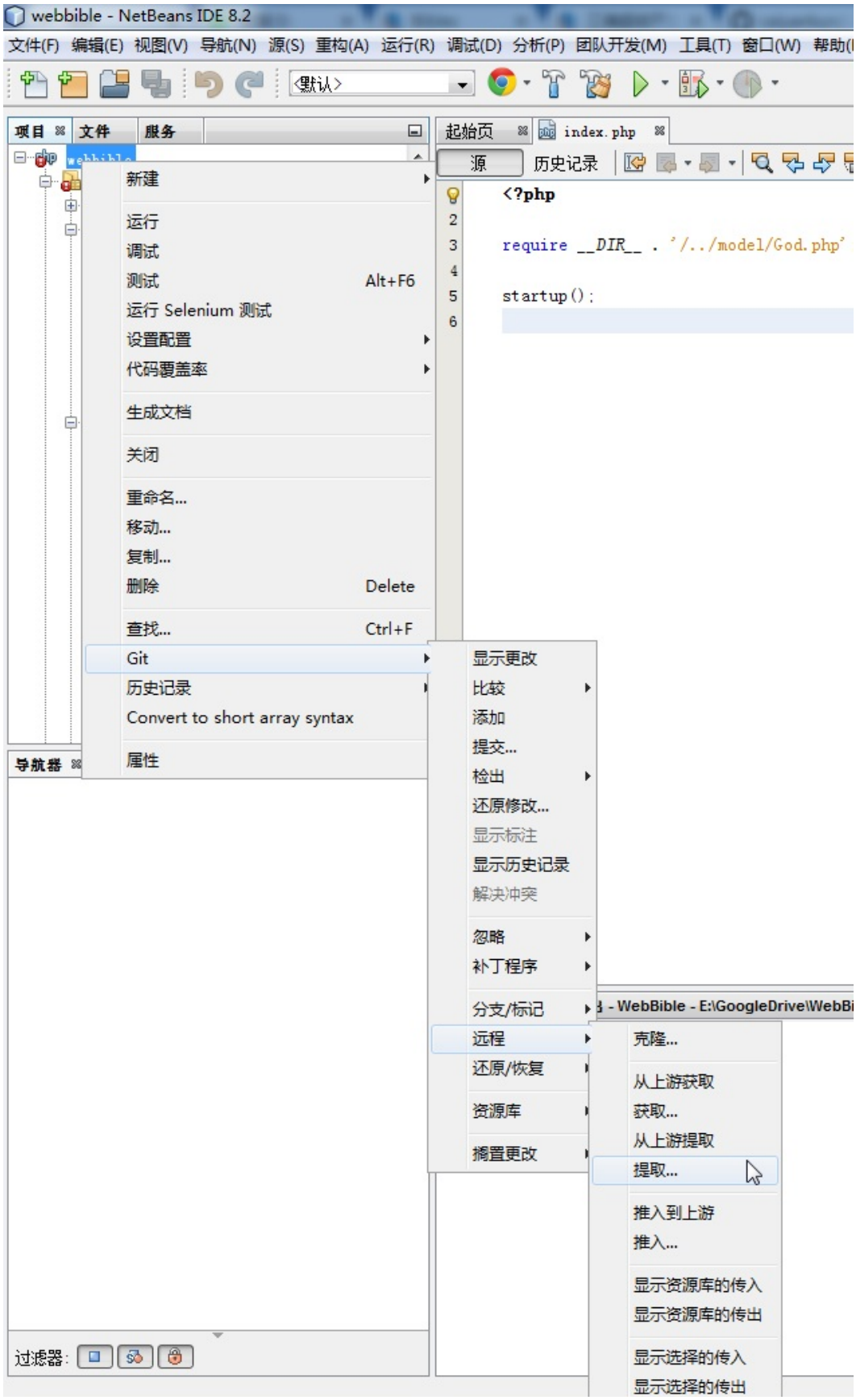
- 安装文档生成工具APIGen
  下载
  http://apigen.org/apigen.phar
  把这个文件拷贝到 C:\Windows\System32
  然后在相同的目录下创建apigen.bat 内容为：
  @C:\wamp64\bin\php\php5.6.31\php.exe "%~dp0apigen.phar" %*
  在cmd中运行apigen，如果能成功显示版本信息，则说明安装成功
  命令行使用方法如下：
  apigen generate --source "D:\web\ruionline" --destination "D:\web\ruionline\doc"
  Netbeans使用如下：




Project Properties - WebBible
Categories:
Sources
Run Configuration
Browser
JavaScript Libraries
npm
Bower
CDNJS
JavaScript
CSS Preprocessors
Include Path
Ignored Folders
Testing
JavaScript Testing
Selenium Testing
Documentation
Annotations
License Headers
Formatting
Documentation Provider: ApiGen
Manual Configuration:
Target Directory: C:\Users\Kevin\Desktop\Web开发\APIgendocs
Browse...
Title: WebBible
Charsets: UTF-8
Comma (",") separated values.
Excludes:
Comma (",") separated values.
Access Levels: Public Protected Private
Internal Deprecated Source Code
PHP Todo
Tree Download
Configuration File:
Configuration:
Browse...
OK
Cancel
Help



- 在github上创建一个库，记下链接；
- 从Git库检出：Git Pull，填写相应的链接


webbible - NetBeans IDE 8.2
文件(F) 编辑(E) 视图(V) 导航(N) 源(S) 重构(A) 运行(R) 调试(D) 分析(P) 团队开发(M) 工具(T) 窗口(W) 帮助(
<默认>
项目
文件
服务
起始页
index.php
源
历史记录
<?php
require __DIR__ . '/../model/God.php'
startup();
新建
运行
调试
测试 Alt+F6
运行 Selenium 测试
设置配置
代码覆盖率
生成文档
关闭
重命名...
移动...
复制...
删除 Delete
查找... Ctrl+F
Git
历史记录
Convert to short array syntax
属性
显示更改
比较
添加
提交...
检出
还原修改...
显示标注
显示历史记录
解决冲突
忽略
补丁程序
分支/标记
远程
还原/恢复
资源库
搁置更改
- WebBible - E:\GoogleDrive\WebBi
克隆...
从上游获取
获取...
从上游提取
提取...
推入到上游
推入...
显示资源库的传入
显示资源库的传出
显示选择的传入
显示选择的传出
导航器
过滤器:

pull完成后如果不是一个工程文件会提示你是否要新建一个工程

- 如果你是本地创建的文件想push到github，那么先要pull（合并）然后在push

框架的使用

惰性加载机制

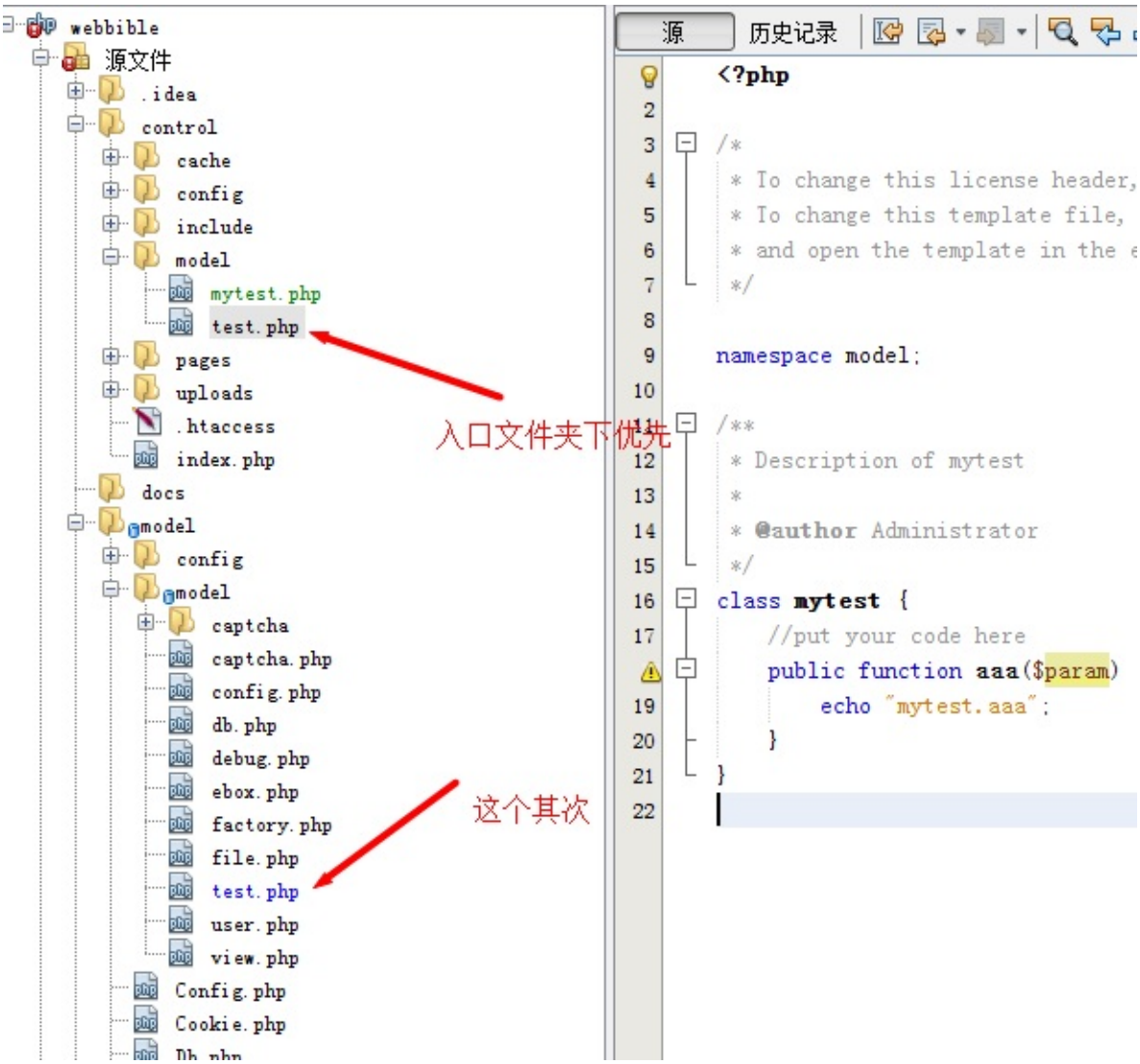
- 惰性加载机制是在god.php实现的
- 惰性加载能成功的前提是：
  1. 编写的php类要与其文件名严格相同，
  2. 而这个文件名相对于框架的MODEL文件夹的相对路径与这个类所在的命名空间的路径是严格一致的

- 当程序调用一个未知的函数时，会自动跳转到autoload函数尝试加载，之后再调用一次
`spl_autoload_register(__NAMESPACE__ .'\Loader::autoload'); // 注册`

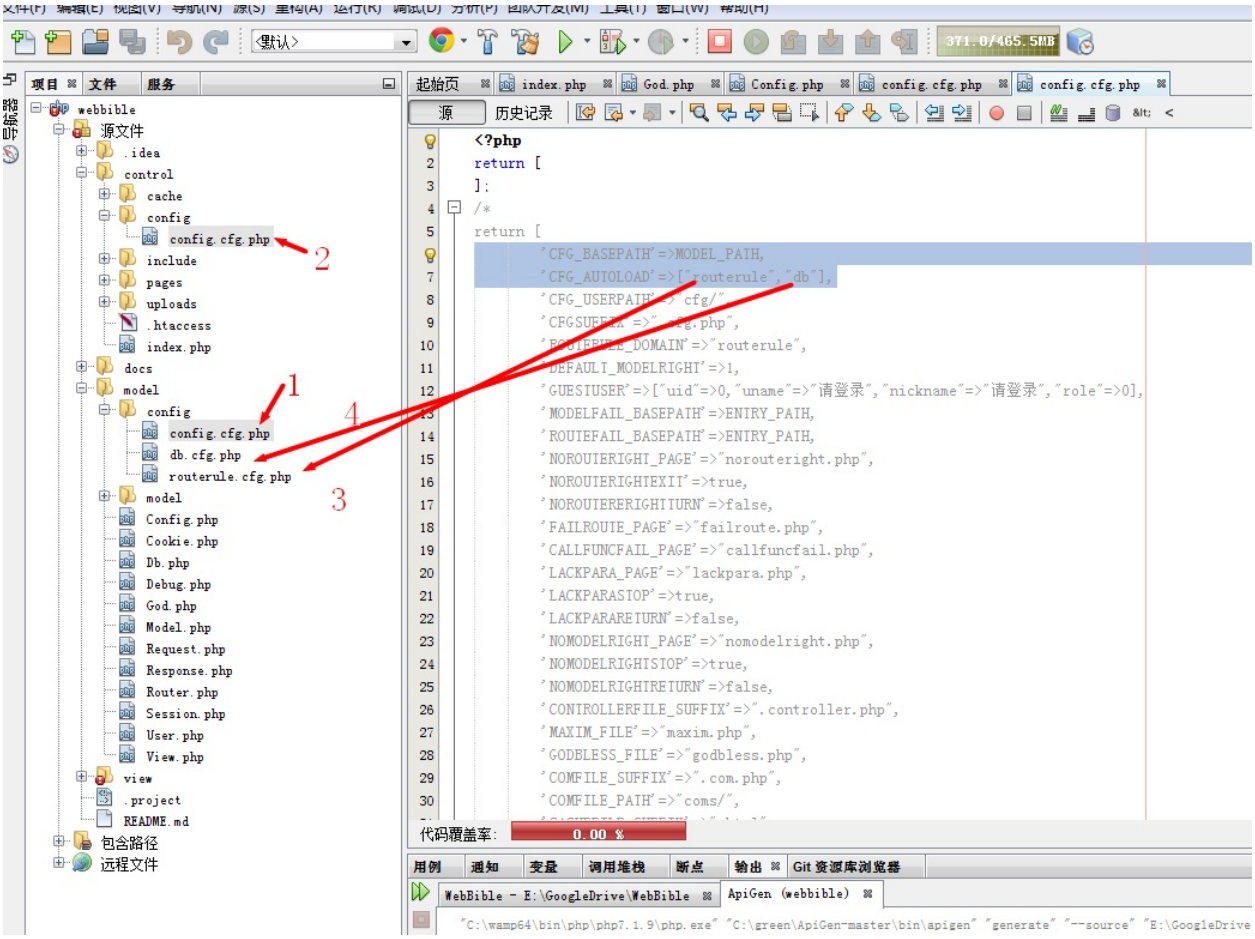
```
//实现类的自动加载功能，通过spl_autoload_register
class Loader
{
	/* 路径映射 */
	public static $namespaceroot = MODEL_PATH;
	public static $loadlog =[];
	/**
	* 自动加载器
	*/
	public static function autoload($class)
	{
	    //echo "<br>Request class:".$class;
	    $file = self::findFile($class);
	    //echo "<br>class file should be:".$file ;
	    if (file_exists($file)) {
	        self::includeFile($file);
	        //echo "<br>loaded:".$file ;
	    }
	}
	/**
	* 解析文件路径
	*/
	private static function findFile($class)
	{
	  $rsv=self::$namespaceroot.$class.".php";
	  $rsv=strtr($rsv,"\\",DIRECTORY_SEPARATOR);
	    return $rsv; // 文件标准路径
	}
	/**
	* 引入文件
	*/
	private static function includeFile($file)
	{
	    if (is_file($file)) {
	        include $file;
	        //echo "<br>loaded file:".$file;
	    }
```

```
        }
}
```

- model命名空间是特殊的命名空间，当框架检测到你要加载一个model命名空间的类时，会优先判断入口文件的model文件夹下是否有相应的文件，如果有就加载之，如果没有，才加载框架的model文件夹下的相应文件。
  这种实现机制保障了不同的网站之间可以有公共的自定义类，也保障了每个网站可以有自己独特的自定义类



配置文件加载机制



这个类是为了方便的按照一定规则读取配置文件，
如果用户没有进行相应的设置，就返回客户调用时指定的值；
这个文件在惰性加载的时候会自动执行一次Autoload：

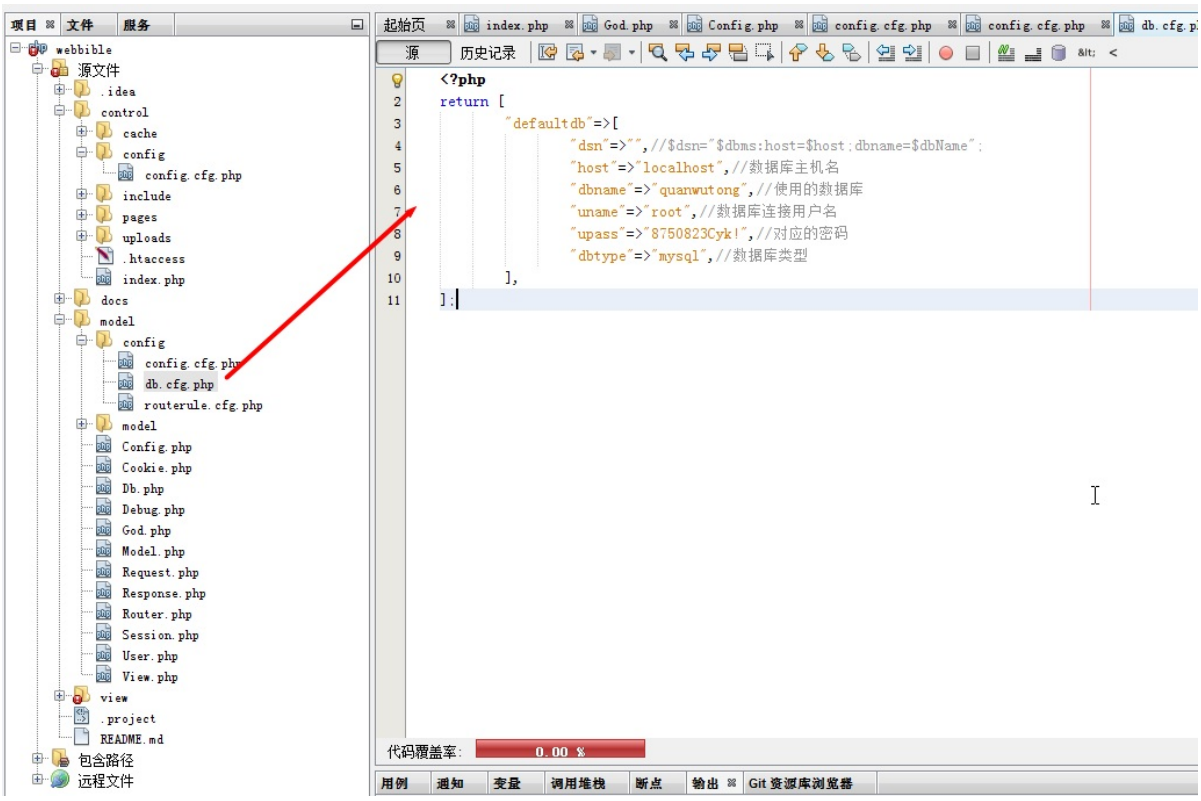
1,首先 Model目录下的主配置文件是必须要加载的；
2,其次 入口文件下的config目录下的主配置文件也是必须要加载的；
3,加载完成后读取到内存中的cfg_userpath,CFG_AUTOLOAD,CFGSUFFIX 这三个变量，然后依次加载CFG_AUTOLOAD 中定义的配置文件；
4,配置文件是有domain限制的，Model主配置文件和用户主配置文件中的配置所属的domain为空，每个额外的单独的配置文件，其domain为该文件名，如db.cfg.php中的变量，将会存放在domain为db的配置中
5,配置存取时都会强制转换为大写，因此在配置文件中，或者在使用配置时，是不用区分大

小写的。

```
    public static function autoload(){
        self::load(MODEL_PATH.MODELCFG_FILE);//加载配置变量
        $cfg_userpath=MODEL_PATH.self::get("cfg_userpath","","");
        $cfgfiles=self::get("CFG_AUTOLOAD","",[]);
        $cfgsuffix=self::get("CFGSUFFIX","",DEFAULT_CFGSUFFIX);
        foreach ($cfgfiles as $cfgfile){
            self::load($cfg_userpath.$cfgfile.$cfgsuffix,$cfgfile);
        }
    }
```

数据库配置机制

- 数据库的配置：



这里可以配置多种数据库配置，在执行的时候可以进行选择指定，如果不指定，会使用默认的数据库；
如：

```
Db::query("select * from user;");//会使用默认数据库
Db::bycfg("anotherdb")::query("select * from user;");//会使用anotherdb的连接
Db::setup([
        "dsn"=>"",//$dsn="$dbms:host=$host;dbname=$dbName";
        "host"=>"localhost",//数据库主机名
        "dbname"=>"test",//使用的数据库
        "uname"=>"root",//数据库连接用户名
        "upass"=>"",//对应的密码
        "dbtype"=>"mysql",//数据库类型
])::query("select * from user;");//会使用anotherdb的连接
```

- 数据库的连接：

数据库会自动连接，直接执行配置或者查询语句就可以，连接机制如下：

Query -》getInstance-》Init

init的时候会自动加载默认配置，如果调用setup或者bycfg，则会覆盖默认配置

~~~
public static function query($str,$assoc=0){

```
    $conn=self::getInstance();
    if(is_null($conn)){
        return false;
    } else {
        self::clear();
        try{
            $pdostate=$conn->query($str,$assoc?PDO::FETCH_ASSOC:PDO::FETC
H_NUM);
            self::$data=[];
            self::$data[0]=$pdostate->fetchAll();
        } catch (PDOException $e) {
            self::$error=$e->getCode();
            self::$info=$e->getMessage();
            return false;
        }
        return true;
    }
}

public static function getInstance(  ) {
        self::init();
    if(!self::$objInstance){
        if(empty(self::$config["dsn"])){
            self::$config["dsn"]=self::$config["dbtype"].":host=".self::$
config["host"].";dbname=".self::$config["dbname"];
        }
        $dsn=self::$config["dsn"];
        $uname=self::$config["uname"];
        $upass=self::$config["upass"];
        //echo json_encode(self::$config);
        //exit(0);
        try {
          self::$objInstance = new PDO($dsn, $uname, $upass);
          self::$objInstance->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMOD
E_EXCEPTION);
          self::$objInstance->query("SET NAMES utf8");
        } catch (PDOException $e) {
          self::$error=$e->getCode();
          self::$info=$e->getMessage();
          self::$objInstance=null;
        }
    }
    return self::$objInstance;
```
~~~

```
}
```

```
private static function init(){
        if(!self::$inited){
        self::$config=array_merge(self::$config,Config::get("defaultdb","
db",[]));
        self::$inited=true;
        }
        return true;
}
```

```
* 数据库的查询：
```

```
public static function query($str,$assoc=0)；//所有的查询都通过这个函数最终执行
public static function callproc ($procname,$params)；//调用数据库的存储过程.
举例：callproc("user.logincheck",[uname=>"cyk",_upass=>"123",_result=>0])
, 函数会将执行拆分成三个语句来执行：select @uname:="cyk",@_upass=123;
callproc..
获取返回值，装在Db::data中；
select @_pass;
获取返回值，装在Db::outvars[]中；
public static function simplecall($procname,$params)；//调用简易存储过程时,
参数是普通数组即可，一样可以带有输出参数，输出参数必须以_开头.举例：simplecall("user
.logincheck",["cyk",123,_result]).该函数不能有输入输出参数
```

```
* 数据库的查询结果：
数据库执行成功与否直接由函数的返回值为true或者false决定；
当判断为执行成功的时候，结果集会存放在

使用query返回的至多是二维数组；存放在Db:;data[0]中；

使用callproc返回的也是三维数组（不带关联）；数据的标题存放在Db::$datatitle中；

如果存储过程带有输出参数，输出参数会放在self::$outvars 中；
直接使用Db::$outvars[参数名]即可访问

* 错误处理机制：

在编写存储过程的时候，如果判断业务执行不成功，存储过程可通过抛出异常的方式告知PHP
```

BEGIN

DECLARE matchnum INT default 0;
call username_check(p_uname,matchnum);
if matchnum 0 then

```
     select uid,uname,ulevel,uoption from user_basic where uname=p_uname
and upass=p_upass;
   else
     SIGNAL SQLSTATE '45001' SET MESSAGE_TEXT = 'Password incorrect!';
   end if;
```

end if;
END

```
PHP在收到异常的时候，首先query函数返回false，然后把错误码和错误信息记录下来：
Db::$error和Db::$info中
```

```
public static function query($str,$assoc=0){
    $conn=self::getInstance();
    if(is_null($conn)){
        return false;
    } else {
        self::clear();
        try{
            $pdostate=$conn->query($str,$assoc?PDO::FETCH_ASSOC:PDO::FETC
H_NUM);
            self::$data=[];
            self::$data[0]=$pdostate->fetchAll();
        } catch (PDOException $e) {
            self::$error=$e->getCode();
            self::$info=$e->getMessage();
            return false;
        }
        return true;
    }
}
```

路由机制

- 路由的执行流程：
  接收用户对框架的一切访问，按照相应的规则进行转发；
  （1）当pathinfo为空时，直接转发到入口文件夹下的myindex.php，不做其他任何操作以节省消耗；参数机制怎么实现？
  （2）当pathinfo不为空时，首先匹配路由权限表（由配置文件定义），对路由进行权限鉴定；
  （3）当后缀为.php时，查找入口文件夹下相应的xxx.yyy.controller.php进行执行；
  （4）当后缀为.func时，查找用户model类中相应的函数进行执行；

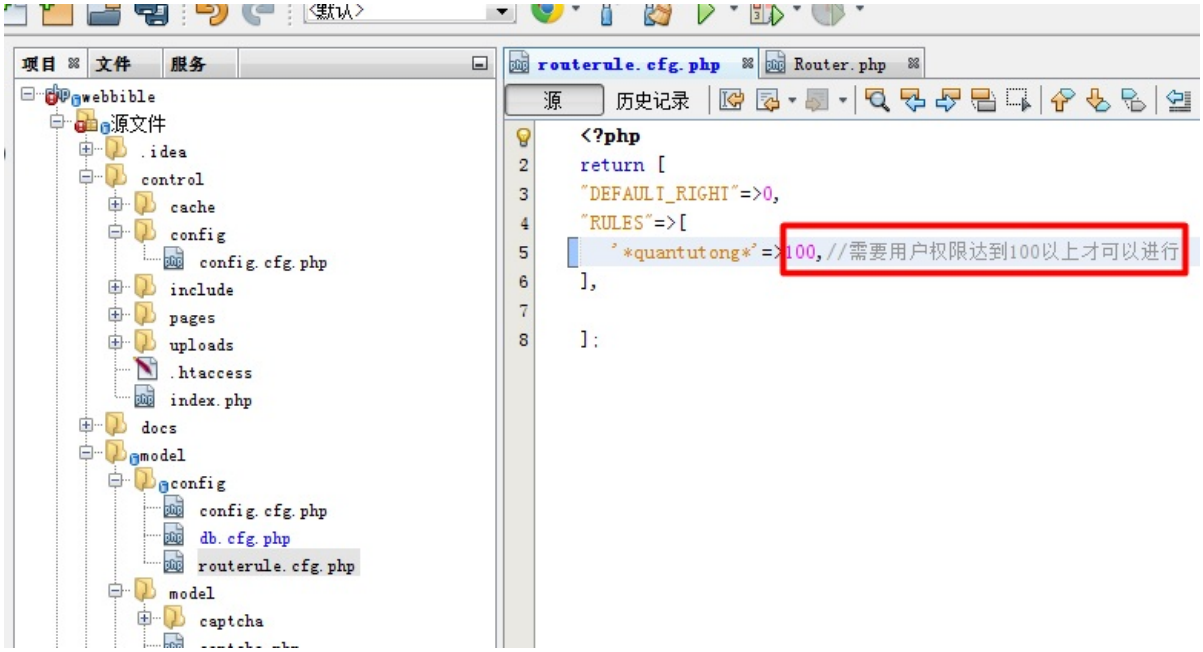
```
    public static function distribute($pathinfo) {
        if(empty($pathinfo)){
                        empty(include ENTRY_PATH ."myindex.php")&&empty(i
nclude ENTRY_PATH ."index.html");
                        return "";
            }
            $pathinfo=preg_replace("/^\//","",$pathinfo);
            self::$pathinfo=$pathinfo;

            (self::checkrouteright($pathinfo)||self::norouteright($pathin
fo))&&
            self::beforedistribute()&&
            self::distriutetoview()||
            self::distriutetomodel()||
            self::faildistribute();
            return self::$result;
    }
```

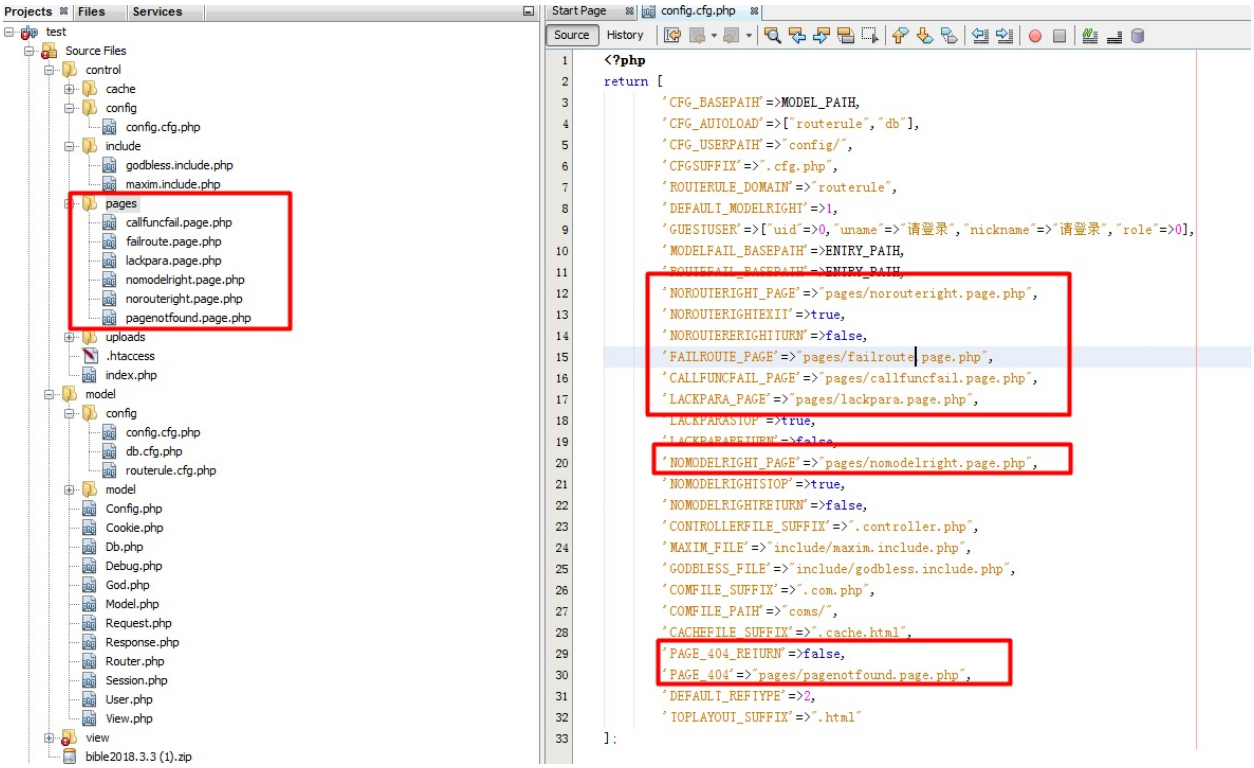
- 路由的权限校验机制：
  框架没有实现鸡肋的路由转发功能
  每个pathinfo默认需要的用户权限为0，即任何人都可以访问
  当在rules列表中用正则表达式定义了匹配当前pathinfo的规则时（只以第一次匹配为准），则是否有权限取决于其后面的键值

```
public static $routeright=0;
public static function checkrouteright($pathinfo){
  $rules=Config::get("rules",ROUTERULE_DOMAIN,[]);
self::$routeright=Config::get("DEFAULT_RIGHT",ROUTERULE_DOMAIN,0);
foreach ($rules as $rule=>$rightlevel){
        if(preg_match($rule,$pathinfo)){
                self::$routeright=$rightlevel;
```

```
                return User::checkright(self::$routeright);
        }
}
  return User::checkright(self::$routeright);
}
```



- 路由失败机制： 路由失败后，跳转到配置文件中定义的页面中
  没有路由权限：
  路由失败：不符合controller文件规则，又不符合用户model函数规则；
  文件不存在；能匹配出controller文件规则，但找不到该controller文件；
  函数不存在：能匹配出函数调用规则，但找不到该函数；


Projects
Files
Services
test
Source Files
control
cache
config
config.cfg.php
include
godbless.include.php
maxim.include.php
pages
callfuncfail.page.php
failroute.page.php
lackpara.page.php
nomodelright.page.php
norouteright.page.php
pagenotfound.page.php
uploads
.htaccess
index.php
model
config
config.cfg.php
db.cfg.php
routerule.cfg.php
model
Config.php
Cookie.php
Db.php
Debug.php
God.php
Model.php
Request.php
Response.php
Router.php
Session.php
User.php
View.php
view
bible2018.3.3 (1).zip
Start Page
config.cfg.php
Source
History
<?php
return [
'CFG_BASEPATH'=>MODEL_PATH,
'CFG_AUTOLOAD'=>["routerule","db"],
'CFG_USERPATH'=>"config/",
'CFGSUFFIX'=>".cfg.php",
'ROUTERULE_DOMAIN'=>"routerule",
'DEFAULT_MODELRIGHT'=>1,
'GUESTUSER'=>["uid"=>0,"uname"=>"请登录","nickname"=>"请登录","role"=>0],
'MODELFAIL_BASEPATH'=>ENTRY_PATH,
'ROUTEFAIL_BASEPATH'=>ENTRY_PATH,
'NOROUTERIGHT_PAGE'=>"pages/norouteright.page.php",
'NOROUTERIGHTEXIT'=>true,
'NOROUTERERIGHTIURN'=>false,
'FAILROUTE_PAGE'=>"pages/failroute.page.php",
'CALLFUNCFAIL_PAGE'=>"pages/callfuncfail.page.php",
'LACKPARA_PAGE'=>"pages/lackpara.page.php",
'LACKPARASTOP'=>true,
'LACKPARARETURN'=>false,
'NOMODELRIGHT_PAGE'=>"pages/nomodelright.page.php",
'NOMODELRIGHTSTOP'=>true,
'NOMODELRIGHTRETURN'=>false,
'CONTROLLERFILE_SUFFIX'=>".controller.php",
'MAXIM_FILE'=>"include/maxim.include.php",
'GODBLESS_FILE'=>"include/godbless.include.php",
'COMFILE_SUFFIX'=>".com.php",
'COMFILE_PATH'=>"coms/",
'CACHEFILE_SUFFIX'=>".cache.html",
'PAGE_404_RETURN'=>false,
'PAGE_404'=>"pages/pagenotfound.page.php",
'DEFAULT_REFTYPE'=>2,
'TOPLAYOUT_SUFFIX'=>".html"
];

用户及权限机制

- 框架在运行的时候，一定会绑定一个用户身份，而这个用户身份一定会有它的权限值。
- 这个权限值在路由，Model执行时会被自动拿来和路由，model执行需要的权限进行对比，失败后进行报错。
- 开发者可以在任何代码处检查用户的权限值，在不满足的情况下进行跳转。

```
User::checkright(100)||Response::returnnoright("You don't hav
```

- 在入口文件开始执行的时候，框架默认会以一个匿名的用户身份运行，在进行登录，退出动作的时候，用户身份会自动更改。

路由时自动检测权限：

pathinfo映射到Model方法时也检查权限，在用户定义的类中，只有继承了Model类的才会有自动权限校对。没有继承的，一律不校对。

```
define("MODELFAIL_BASEPATH",Config::get('MODELFAIL_BASEPATH',"",ENTRY_PATH));
define("DEFAULT_MODELRIGHT",Config::get('DEFAULT_MODELRIGHT',"",1));
define("LACKPARASTOP",Config::get('LACKPARASTOP',"",true));
define("LACKPARARETURN",Config::get('LACKPARARETURN',"",false));
define("NOMODELRIGHTSTOP",Config::get('NOMODELRIGHTSTOP',"",true));
define("NOMODELRIGHTRETURN",Config::get('NOMODELRIGHTRETURN',"",false));

!defined('NOMODELRIGHT_PAGE')&&define('NOMODELRIGHT_PAGE',Config::get('NOMODELRIGHT_PAGE',"","pages/nomodelright.page.php"));
!defined('LACKPARA_PAGE')&&define('LACKPARA_PAGE',Config::get('LACKPARA_PAGE',"","pages/lackpara.page.php"));
!defined('CALLFUNCFAIL_PAGE')&&define('CALLFUNCFAIL_PAGE',Config::get('CALLFUNCFAIL_PAGE',"","pages/callfuncfail.page.php"));
```

```
class Model
{
    public $accesslevel=null;
    public static $failcode=null;
    public static $result=null;
    function __construct() {
        is_null($this->accesslevel)&&($this->accesslevel=DEFAULT_MODELRIGHT);
        User::checkright($this->accesslevel)||$this->noright();
    }
```

在配置文件中可以修改默认Model映射的权限
'DEFAULT_MODELRIGHT'=>1,

调试及日志机制

/实现Debug功能，拦截地址栏输入，实现访问任务的分发：
（0）记录程序的执行过程;
（1）错误，调试帮助；
（2）有开关可以控制；
*/

```
defined("DEBUG_ENABLE")||define("DEBUG_ENABLE",false); //是否使能Debug功能
defined("DEBUG_ON")||define("DEBUG_ON",false);//记录日志的时候是否直接打印出来
defined("LOG_GLUE")||define("LOG_GLUE","<br>\r\n");//打印的时候，每一条日志之间的
class Debug
{
    public static $on=DEBUG_ON;//on: 记录+输出; off: 只记录;
    public static $log=[];
    public static $logcount=0;
    public static $logglue=LOG_GLUE;
    public static function turnon(){...4 行 }
    public static function turnoff(){...4 行 }
    public static function logglue($glue=null){...8 行 }
    public static function log($obj){...15 行 }
    public static function dump($spliter="<br>\r\n"){...4 行 }
}
```

配置文件中配置 DEBUG_ENABLE

在任何需要记录日志信息的地方执行：
`DEBUG_ENABLE && Debug::log("内容");`

Session&Cookie机制

- Session和Cookie被封装了起来，不需要人为启动，在调用的时候会自动启动;
- Session在自动启动的时候会自动判断Cookie中是否有autosession=>sessionid 这样的字段，如果有的话会自动使用该Sessionid； 这样就相当于实现了session生命周期的自动延续,一直延续到cookie失效；

```
savesession($timer=-3600)
hassession()
getsession()
```

- 用户登录保存机制：

用户登录，其实就是在Session中写入一个_user变量，其值为一个User对象；所有时候，要获取用户信息，只要从Session中取_user变量的值即可

session在浏览器生命周期内一直有效，然后当用户关闭浏览器后会自动失效。通过登录的时候设置的savesession($timer=-3600)可延长session的有效期直到cookie失效，实现了用户登录状态的保存。这种方式不仅能保存用户的登录状态，而且保存了session中所有的变量，（例如，如果购物车是通过session实现的，那么也会保存下来）

User::checkright()->User::getrole()->User::info()->Session::has("_user")->Session::boot()->Session->init()->Cookie::getsession()->session_id(Cookie::get("autosession"))

cookie机制：

```
/**
     * Cookie清空
     * @param string|null $prefix cookie前缀
     * @return mixed
     */
    public static function clear($prefix = null)

        /**
     * Cookie 设置、获取、删除
     *
     * @param string $name  cookie名称
     * @param mixed  $value cookie值
     * @param mixed  $option 可选参数 可能会是 null|integer|string
     *
     * @return mixed
```

```
     * @internal param mixed $options cookie参数
     */
    public static function set($name, $value = '', $option = null)

        /**
     * 判断Cookie数据
     * @param string        $name cookie名称
     * @param string|null   $prefix cookie前缀
     * @return bool
     */
    public static function has($name, $prefix = null)

        /**
     * Cookie获取
     * @param string        $name cookie名称
     * @param string|null   $prefix cookie前缀
     * @return mixed
     */
    public static function get($name, $prefix = null)

        /**
     * Cookie删除
     * @param string        $name cookie名称
     * @param string|null   $prefix cookie前缀
     * @return mixed
     */
    public static function delete($name, $prefix = null)
```

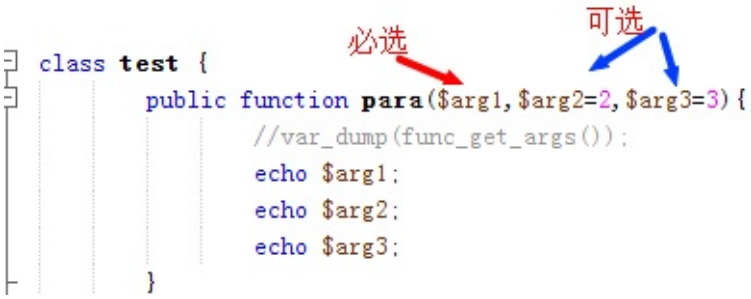
```
* 原始cookie set函数
```

```
setcookie(name,value,expire,path,domain,secure)
参数     描述
name     必需。规定 cookie 的名称。
value    必需。规定 cookie 的值。
expire   可选。规定 cookie 的有效期。
path     可选。规定 cookie 的服务器路径。
domain   可选。规定 cookie 的域名。
secure   可选。规定是否通过安全的 HTTPS 连接来传输 cookie。
```

前后台交互规范

原则：

1. 完全的前后端分离；
2. 后端与数据库操作分离；
   规范：
   - 所有的数据库操作都用存储过程实现，PHP不实现任何数据库操作语句；
   - 业务执行成功与否由存储过程自行判断，只要业务执行不成功，就抛出自定义异常告知php。PHP在调用存储过程中，只要没有检测到异常，就判定为业务执行成功。然后再读取存储过程返回的数据。
   - 前台向后台提交数据时要么用地址栏的地址，要么用Ajax，前台向后台传递数据时有多种方式地址栏？形式的GET，POST，pathinfo方式
   - 提交的数据与映射到的后台函数的参数是一一对应的，这个对应动作和参数赋值是在Model::callfunccontrol中实现的。
     比如以pathinfo方式提交时，pathinfo的各个数据会依顺序依次传递给映射函数的参数；
     以POST或者get方式提交时，与参数名称相同的变量会被赋值给映射函数的参数；
     当存在映射函数的某些没有默认值的参数得不到赋值时，会报错“lack para”



调用方式举例：

- 当使用了不止一种方式给后台提交数据时，其给映射函数的赋值优先级是：
  pathinfo>POST/GET>默认参数值

其实现机制是：
1，首先按函数的定义获取一个参数名称列表；
2，对参数名称进行逐一遍历：
2.1 如果当前参数名有默认值，就先把默认值填入参数列表；
2.2 如果当前参数名有通过？的形式将参数通过get方法传递，就将其值覆盖掉默认参数值；
2.3 如果客户有通过pathinfo模式传递参数值，该值就会最终按顺序覆盖掉相应的参数值
2.4 如果某个函数名对上面三个都没成功，进行报错

- 后台向前台返回的所有业务性质的数据，全部通过Response::returntaskok() 或者 Response::returntaskfail() 方式返回
  其目的是固定封装返回的JSON为如下格式：
  `{"_taskresult":"我是数据","_taskstat":{"code":0,"info":"我是信息"}}`
  只有返回的数据是这种结构，而且code为0时，前端才能判定该业务是执行成功的（如登录成功，注册成功等），
  当成功时，前台提取_taskresult为需要的数据；
  当失败时，前台提取_taskstat.info作为错误的提示；

实战：开发网站后台应用

例子：开发more.sanmantech.com 网站的后台（假定该域名已经指向本机地址），实现用户注册，登录验证的后台功能
实现方法：
1，安装框架，设置好开发环境；
2，复制一下control文件夹，命名为more.sanmantech.com;

3，配置apache的documentroot或者虚拟主机documentroot 为more.sanmantech.com 文件夹
4，在more.sanmantech.com 的 model目录下创建一个测试文件test.php

```
<?php
namespace model;

class test {
    public function para($arg1,$arg2=2,$arg3=3){
        echo $arg1;
        echo $arg2;
        echo $arg3;
    }
}
```

注意点：

- 命名空间必须为model；
- 类名必须与文件名一致，为test；
- 类名里面的每个成员函数即可为一个后台应用映射函数，对应一个后台业务处理的网址；
- 这个测试的para函数实现的后台功能为将前台提交的三个参数打印到浏览器
  5，在浏览器中输入网址查看是否正常：

```
http://local.webbible.com/test/para.func  //输出lack para
http://local.webbible.com/test/para.func/a  //输出 a23
http://local.webbible.com/test/para.func?arg1=4 //输出423
```

6, 在more.sanmantech.com目录的config目录下新建/修改config.cfg.php 修改配置文件的基地址和自动加载db配置文件

```
<?php
return [
  'CFG_BASEPATH'=>ENTRY_PATH,  //配置框架加载本网站独有的配置文件
  'CFG_AUTOLOAD'=>["db"], //配置自动加载额外的db.cfg.php两个配置文件
  'CFG_USERPATH'=>"config/",//额外的配置文件所在的路径（和本文件在同一目录）
  'CFGSUFFIX'=>".cfg.php",//额外的配置文件的后缀
];
```

7,在more.sanmantech.com目录的config目录下新建/修改db.cfg.php，配置数据库的连接

```
<?php
return [
  "defaultdb"=>[
      "dsn"=>"",//$dsn="$dbms:host=$host;dbname=$dbName";
      "host"=>"localhost",//数据库主机名
      "dbname"=>"user",//使用的数据库,在使用存储过程方式的工程中，这个字段无关
紧要
      "uname"=>"test",//数据库连接用户名
      "upass"=>"1234Abcd！",//对应的密码
      "dbtype"=>"mysql",//数据库类型
  ],
];
```

假定user数据库中已经包含两个存储过程，一个是用于登录检查，一个是用于注册新用户：

login(p_uname,p_upass)

```
BEGIN
 DECLARE matchnum INT default 0;
 call username_check(p_uname,matchnum);
 if matchnum <1 then
     SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'No such user!';
 else
     select count(*) into matchnum from user_basic where uname=p_uname
 and upass=p_upass;
     if matchnum>0 then
       select uid,uname,ulevel,uoption from user_basic where uname=p_u
name and upass=p_upass;
     else
       SIGNAL SQLSTATE '45001' SET MESSAGE_TEXT = 'Password incorrect!
';
     end if;
 end if;
```

```
  END
```

register(p_uname,p_upass)

```
BEGIN
  declare userexist int default 0;
  call username_check(p_uname,userexist);
  if userexist>0 then
       SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'User already exist!';
  else
       insert into user_basic set uname=p_uname,upass=p_upass;
       if row_count()<1 THEN
                SIGNAL SQLSTATE '45001' SET MESSAGE_TEXT = 'Insert failed
! Affected row is 0';
       end if;
  end if;
END
```

8,在more.sanmantech.com下的model文件夹下创建user.php

```
<?php
namespace model;

class user
{
    public function login($uname,$upass) {
        if(!\Db::simplecall("user.login",array($uname,md5($upass)))){
            \Response::returntaskfail(null,\Db::$error, \Db::$info);
        } else {
            \Response::returntaskok(\Db::tabledata());
        }
    }
    public function register($uname,$upass) {
        if(!\Db::simplecall("user.register",array($uname,md5($upass)))){
            \Response::returntaskfail(null,\Db::$error, \Db::$info);
        } else {
            \Response::returntaskok(\Db::tabledata());
        }
    }
}
```

9,后台基本功能已经实现，在浏览器中进行测试

输入一个不存在的用户：
`http://local.webbible.com/user/login.func/aaa/bbb`
返回：
`{"_taskresult":null,"_taskstat":{"code":"45000","info":"SQLSTATE[45`

输入：
`http://local.webbible.com/user/register.func/aaa/bbb`
返回：
`{"_taskresult":[[]],"_taskstat":{"code":0,"info":""}}`

再次输入：
`http://local.webbible.com/user/login.func/aaa/bbb`
返回：
`{"_taskresult":[["42","aaa","1",null]],"_taskstat":{"code":0,"info"`

附录1：常量

- 使用define定义的常量是没有命名空间限制的，所有的命名空间可以直接访问；
- 使用cost定义的常量是有命名空间显示的，只能带着命名空间访问；
- 如果多处可能出现对同一个常量的声明，一般都先判断是否声明，因此其值为第一次定义时的值

God.php

```
define('WEBROOT_PATH',$_SERVER['DOCUMENT_ROOT']);//整个站点的根目录,假定是后面不带\的
define('ENTRY_PATH',dirname($_SERVER['SCRIPT_FILENAME']).'/');//入口文件 Path,有的时候没入口文件呢?
define('PATH_INFO',$_SERVER['PATH_INFO']);//截取Pathinfo信息
define('MODEL_PATH',__DIR__.'/');//模型库目录,命名空间的顶部空间
define('FRAMEROOT_PATH',MODEL_PATH.'../');//框架根目录
define('VIEW_PATH',FRAMEROOT_PATH.'view/');//VIEW目录(旧工厂目录)
define('CONTROL_PATH',FRAMEROOT_PATH.'control/');//VIEW目录(旧工厂目录)
define('STATIC_PATH',VIEW_PATH.'static/');
define('COMMON_PATH',FRAMEROOT_PATH.'common/');
define('3COM_PATH',COMMON_PATH.'3com/');
define('CACHE_PATH',ENTRY_PATH.'cache/');
define('HTMLROOT_PATH',str_replace(WEBROOT_PATH,'',ENTRY_PATH)); //入口文件目录相对于站点目录的相对路径，因为站点目录不一定是部署在根目录
define('HTMLSTATIC_PATH',HTMLROOT_PATH."static/"); //入口文件目录相对于站点目录的相对路径，因为站点目录不一定是部署在根目录
define('HTML3COM_PATH',HTMLROOT_PATH."static/"); //入口文件目录相对于站点目录的相对路径，因为站点目录不一定是部署在根目录
define('HTMLCACHE_PATH',HTMLROOT_PATH."cache/"); //入口文件目录相对于站点目录的相对路径，因为站点目录不一定是部署在根目录
```

Rooter.php：

```
defined("CONTROLLERFILE_SUFFIX")||define("CONTROLLERFILE_SUFFIX",Config::get('CONTROLLERFILE_SUFFIX',"",".controller.php"));
define("NOROUTERIGHTEXIT",Config::get('NOROUTERIGHTEXIT',"",true)); //当发现没有路由权限时是否要停止执行脚本
define("NOROUTERERIGHTTURN",Config::get('NOROUTERERIGHTTURN',"",false));//当发现没有路由权限时返回什么值
define("ROUTERULE_DOMAIN",Config::get('ROUTERULE_DOMAIN',"","routerule"));
!defined('FAILROUTE_PAGE')&&define('FAILROUTE_PAGE',Config::get('FAILROUTE_PAGE',"","pages/failroute.page.php"));
!defined('NOROUTERIGHT_PAGE')&&define('NOROUTERIGHT_PAGE',Config::get('NOROUTERIGHT_PAGE',"","pages/failroute.page.php"));
```

Model.php

```
define("MODELFAIL_BASEPATH",Config::get('MODELFAIL_BASEPATH',"",ENTRY_PAT
H));
define("DEFAULT_MODELRIGHT",Config::get('DEFAULT_MODELRIGHT',"",1));
define("LACKPARASTOP",Config::get('LACKPARASTOP',"",true));
define("LACKPARARETURN",Config::get('LACKPARARETURN',"",false));
define("NOMODELRIGHTSTOP",Config::get('NOMODELRIGHTSTOP',"",true));
define("NOMODELRIGHTRETURN",Config::get('NOMODELRIGHTRETURN',"",false));

!defined('NOMODELRIGHT_PAGE')&&define('NOMODELRIGHT_PAGE',Config::get('NO
MODELRIGHT_PAGE',"","pages/nomodelright.page.php"));
!defined('LACKPARA_PAGE')&&define('LACKPARA_PAGE',Config::get('LACKPARA_P
AGE',"","pages/lackpara.page.php"));
!defined('CALLFUNCFAIL_PAGE')&&define('CALLFUNCFAIL_PAGE',Config::get('CA
LLFUNCFAIL_PAGE',"","pages/callfuncfail.page.php"));
```

View.php

```
defined("CONTROLLERFILE_SUFFIX")||define("CONTROLLERFILE_SUFFIX",Config::
get('CONTROLLERFILE_SUFFIX',"",".controller.php"));
define("CACHEFILE_SUFFIX",Config::get('CACHEFILE_SUFFIX',"",".cache.html"
));
define("MAXIM_FILE",Config::get('MAXIM_FILE',"","include/maxim.include.ph
p"));
define("GODBLESS_FILE",Config::get('GODBLESS_FILE',"","include/godbless.i
nclude.php"));
define("COMFILE_SUFFIX",Config::get('COMFILE_SUFFIX',"",".com.php"));
define("COMFILE_PATH",Config::get('COMFILE_PATH',"","coms/"));
define("DEFAULT_REFTYPE",Config::get('DEFAULT_REFTYPE',"",2));//0：分开引用
；1:直接内嵌；2：组合成一个大的JS，CSS进行引用。前提是模板要同时重建
!defined("PAGE_404")&&define("PAGE_404",Config::get('PAGE_404',"","pages/
pagenotfound.page.php"));

!defined("FACTORY_PATH")&&define("FACTORY_PATH",Config::get('FACTORY_PATH
',"",ENTRY_PATH."factory/"));
```

Debug.php

```
defined("DEBUG_ENABLE")||define("DEBUG_ENABLE",false); //是否使能Debug功能
defined("DEBUG_ON")||define("DEBUG_ON",false);//记录日志的时候是否直接打印出来
defined("LOG_GLUE")||define("LOG_GLUE","<br>\r\n");//打印的时候，每一条日志之
间的
```

附录2：配置项

- 配置文件后缀默认为xxx.cfg.php

```
<?php
return [
        'CFG_BASEPATH'=>MODEL_PATH,//额外的配置文件的基地址，框架默认的两个基本
的配置文件不受以下四个变量控制
        'CFG_AUTOLOAD'=>["routerule","db"],//需要自动加载的额外的配置文件，每个
文件名是一个domain,如果要连接数据库，必须放db在里面且名称不可更改
        'CFG_USERPATH'=>"config/",//额外的配置文件相对于基地址的路径
        'CFGSUFFIX'=>".cfg.php",//额外的配置文件的后缀名字
        'ROUTERULE_DOMAIN'=>"routerule",//路由权限规则表在配置中的哪个domain
        'DEFAULT_MODELRIGHT'=>1,//默认的继承自Model类的类执行所需要的用户权限
        'REQUEST_LIMIT'=>false,//配置全局的是否限制只响应POST请求
        'GUESTUSER'=>["uid"=>0,"uname"=>"请登录","nickname"=>"请登录","role
"=>0],//当用户未登录时查询用户信息所显示的内容
        //'MODELFAIL_BASEPATH'=>ENTRY_PATH,丢弃
        //'ROUTEFAIL_BASEPATH'=>ENTRY_PATH,丢弃
        'NOROUTERIGHT_PAGE'=>"pages/norouteright.page.php",//没有路由权限时
显示的页面
        'NOROUTERIGHTEXIT'=>true,//没有路由权限时，是否停止执行后面的脚本
        //'NOROUTERERIGHTTURN'=>false,丢弃
        'FAILROUTE_PAGE'=>"pages/failroute.page.php",//路由失败时显示的页面
        'CALLFUNCFAIL_PAGE'=>"pages/callfuncfail.page.php",//调用映射函数失
败时显示的页面
        'LACKPARA_PAGE'=>"pages/lackpara.page.php",//缺乏必须的参数时显示的页
面
        'LACKPARASTOP'=>true,//缺乏参数时，是否停止执行后面的脚本
        'LACKPARARETURN'=>false,//缺乏参数时，当选择继续执行后面脚本时lackpara
检测函数的返回值
        'NOMODELRIGHT_PAGE'=>"pages/nomodelright.page.php",//没有权限执行映
射函数时显示的页面
        'NOMODELRIGHTSTOP'=>true,//没有权限执行映射函数时是否停止执行后面的代码
        'NOMODELRIGHTRETURN'=>false,//没有权限执行映射函数时当选择继续执行后面的
代码时检测函数的返回值
        'CONTROLLERFILE_SUFFIX'=>".controller.php",//控制器函数的后缀
        'MAXIM_FILE'=>"include/maxim.include.php",//这个文件中存放PHP数组，当
通过框架显示前台页面时，框架自动把这些变量转换为js变量
        'GODBLESS_FILE'=>"include/godbless.include.php",//这个文件中存放对前
台框架运行必须加载的js，css，通过框架显示前台页面时，框架会自动加载这个
        'COMFILE_SUFFIX'=>".com.php",//xxx
        'COMFILE_PATH'=>"coms/",//xxx
        'CACHEFILE_SUFFIX'=>".cache.html",      //xxx
        'PAGE_404_RETURN'=>false,//xxx
        'PAGE_404'=>"pages/pagenotfound.page.php",//找不到前台页面时显示的页
面
        'DEFAULT_REFTYPE'=>2,//xxx
        'TOPLAYOUT_SUFFIX'=>".html"//顶级模版文件的后缀
];
```

db.cfg.php

```
<?php
return [
        "defaultdb"=>[
                "dsn"=>"",//$dsn="$dbms:host=$host;dbname=$dbName";
                "host"=>"localhost",//数据库主机名
                "dbname"=>"quanwutong",//使用的数据库,在使用存储过程方式的工程
中，这个字段无关紧要
                "uname"=>"root",//数据库连接用户名
                "upass"=>"1234Abcd!",//对应的密码
                "dbtype"=>"mysql",//数据库类型
        ],
];
```

routerule.cfg.php

```
<?php
return [
"DEFAULT_RIGHT"=>0,
"RULES"=>[
   '*quantutong*'=>100,//需要用户权限达到100以上才可以进行
],

];
```