# thinkphp

* 配置



配置格式：php , yaml , ini , xml , xml , json

通过在入口文件中定义CONF\_EXT常量来指定格式

配置读取：C函数

C(‘’)读取全部配置项

C(‘name’)读取配置项名称为name的配置值

C(‘name’,null,’default’)读取 配置项name的配置值，如未配置返回default

C(‘parent.sub’)读取二维数组配置

配置的名称不区分大小写，但二维数组配置读取的参数，parent是严格区分大小写

大写字母 + \_

动态配置：

C(‘name’,$value)设置配置项名称为name的配置值

C($config\_array)批量配置

$config\_array中 Key为配置项名称，对应Value为配置值

扩展配置：

通过设置LOAD\_EXT\_CONFIG配置项的值来加载当前模块的自定义配置文件

* 控制器

1. 就是一个php类，继承thinkphp的核心类Controller
2. public外部访问
3. ACTION\_SUFFIX操作方法定义修改
4. 空控制器、空操作方法
5. 操作绑定到类
6. 调用四种方式：自动定位、new实例化、A函数、R函数

URL模式

四种：通过URL\_MODEL配置

0：普通模式 ( http://domain/index.php?m=m&c=c&a=a )

1：PathInfo ( http://domain/index.php?m/c/a )

2：ReWrite ( http://domain/m/c/a )

3：兼容模式

PathInfo和Rewrite需要运行环境的支持

URL动态生成

使用U函数可以动态生成模式匹配的URL:

U(‘地址表达式’,[‘参数’],[‘伪静态后缀’],[‘显示域名’])

-地址表达式：[模块/控制器/操作#锚点@域名]?参数1=值1&参数2=值2…

-第二个参数支持数组和字符串两种定义方式

-类unix系统对URL大小写是敏感的，通过配置URL\_CASE\_INSENSITIVE实现不区分大小写的URL访问

伪静态

为了满足更好的SEO效果，使用：

URL\_HTML\_SUFFIX 配置伪静态后缀

URL\_DENY\_SUFFIX 配置禁止访问的URL后缀

本地Apache服务器实现Rewrite模式

Rewrite是通过URL重写隐藏应用的入口文件index.php:

Apache开启mod\_rewrite.so模块

AllowOverride 设置为 All

定义分布式配置文件 htaccess

重启Apache服务后生效

实现更接近生产环境的URL

通过Apache开启mod\_vhost\_alias.so模块

配置httpd-vhosts.conf

添加hosts主机头实现转发

重启Apache服务后生效

跳转与AJAX数据返回

以下几种跳转和响应：

redirect重定向

参数（地址表达式 附加参数 跳转时间s 提示信息）

success成功提示跳转

参数（提示信息 跳转地址 跳转时间s）

error错误提示跳转

参数（提示信息 跳转地址 跳转时间s）

ajaxReturn实现Ajax数据返回

参数（返回数据 数据格式）

控制器定义了六个常量，用来快速判断请求的类型：

IS\_GET、IS\_POST、IS\_PUT、IS\_DELETE、IS\_AJAX、REQUEST\_METHOD（枚举）

不使用jquery的话，AJAX请求要添加识别参数：通过VAR\_AJAX\_SUBMIT配置

获取输入变量

以下几种方式：

$\_GET/$\_POST/$\_SERVER(不安全)

I函数（Input单词的首字母），方便实现输入变量读取，并自动进行数据安全过滤

Input函数：

I(‘变量类型.变量名’,[‘变量不存在默认值’],[‘过滤方法’],[‘额外数据源‘])

API补充

Action参数绑定是通过直接绑定URL地址中的变量作为操作方法的参数，可以简化方法的定义甚至路由的解析。

Action参数绑定功能默认是开启的，原理是把URL中的参数（不包括模块、控制器和操作名）和操作方法中的参数进行绑定。

* 视图

1.基本结构

*//入口文件绑定模块文件，新建一个模块  
define*(**'BIND\_MODULE'**,**'Views'**);  
*//关闭目录安全文件生成  
define*(**'BUILD\_DIR\_SECURE'**,**false**);

知识点：

1.1.默认视图层（DEFAULT\_V\_LAYER）

1.2.模板文件后缀（TMPL\_TEMPLATE\_SUFFIX）

1.3.连接符（TMPL\_FILE\_DEPR）

1.4.视图目录（VIEW\_PATH）

**'DEFAULT\_V\_LAYER'** => **'Template'**,  
**'TMPL\_TEMPLATE\_SUFFIX'** => **'.jike'**,  
**'TMPL\_FILE\_DEPR'** => **'\_'**,  
**'VIEW\_PATH'** => **'./Theme/Views/'**,

2.视图变量赋值和渲染(动态网页基础)

知识点：

* 1. 使用控制器的assign方法为视图变量赋值：

assign(‘name’,$value)

* 1. 渲染视图（控制器display、fetch和show方法）
  2. display(‘[模板文件]’[,’字符串编码’][,’输出类型’])
  3. fetch(‘[模板文件]’[,’字符串编码’][,’输出类型’])
  4. show(‘渲染内容’[,’字符串编码’][,’输出类型’])
  5. T函数：T([资源://][模块@][主题/][控制器/]操作,[视图分层])

3.主题

知识点：

3.1 设置默认主题（DEFAULT\_THEME）

3.2 控制器theme方法

3.3 多主题切换

3.4 差异主题（TMPL\_LOAD\_DEFAULTTHEME）

* 模板（上）

1. 变量输出

知识点：

1.1自定义变量输出：{$value}

1.2 默认值输出：{$value|default=’默认值’}

1.3系统变量($Think.server/env/post/get/cookie/session/request)

1.4使用运算符需要注意：数学运算，点语法失效

1.5模板解析定界符(TMPL\_L\_DELIM、TMPL\_R\_DELIM)

1. 使用函数

知识点：

2.1变量是第一个参数：{$value|function[=?,?,?]}，例如{$value|trim}

2.1变量不是第一个参数：{$value|str\_replace=?,?,###}}

2.3嵌套使用多个函数：{$value|func1|func2|func3}，优先级从左到右

2.4回归原生PHP：{:expression}

1. 包含文件和模板替换

知识点：

3.1包含文件：<include file=”模板表达式/文件路径[,…]” key=value/> [key]

如：<**include file="Public/navbar" home="主页"**/>

<**li class="active"**><**a href="#"**>[home]<**span class="src-only"**>(current)</**span**></**a**></**li**>

3.2导入资源文件：import、load、js、css

如<**import file="Js.bootstrap" type="js"** />  
<**load href="\_\_PUBLIC\_\_/js/jquery.js"**/>  
<**css href="\_\_PUBLIC\_\_/css/bootstrap.css"** />  
<**js href="\_\_CDN\_\_/js/bootstrap.js"** />

3.3 模板替换：TMPL\_PARSE\_STRING

如*// 模板替换规则定义* **'TMPL\_PARSE\_STRING'**=>**array**(  
 **'\_\_CDN\_\_'**=>**'./Cdn'**,  
 ),

3.4一些默认的替换规则：

\_\_ROOT\_\_，\_\_APP\_\_，\_\_MODULE\_\_，\_\_CONTROLLER\_\_，\_\_ACTION\_\_，\_\_SELF\_\_，\_\_PUBLIC\_\_

* 模型：定义与实例化

1模型的定义

知识点：

1.1模型是以PHP类的方式定义，继承自相应的模型类

1.2约定模型名称是不包括前缀的数据表名称，UserModel对应jike\_user数据表

1.3模型的属性：tablePrefix、tableName、trueTableName、dbName、connection

1.4实例化：new UserModel()、M(‘User’)、D(‘User’)和控模型

如：*//new*$user\_model=**new** \Models\Model\UserModel();  
  
*//M*$user\_m\_model=M(**'User'**);  
  
*//D*$user\_d\_model=D(**'User'**);  
  
*//控模型*$empty\_model=**new** \Think\Model();  
$empty\_m\_model=M();  
$empty\_d\_model=D();

2模型数据库信息配置

知识点：

2.1数据库信息配置

2.2切换数据库：db方法

2.3分布式数据库：DB\_DEPLOY\_TYPE

2.4读写分离：DB\_RW\_SEPARATE、DB\_MASTER\_NUM、DB\_SLAVE\_NO

如：*/\* 数据库设置 \*/***'DB\_TYPE'** => **'mysql'**, *// 数据库类型***'DB\_HOST'** => **'localhost'**, *// 服务器地址***'DB\_NAME'** => **'jike'**, *// 数据库名***'DB\_USER'** => **'root'**, *// 用户名***'DB\_PWD'** => **'root'**, *// 密码***'DB\_PORT'** => **'3306'**, *// 端口***'DB\_PREFIX'** => **'jike\_'**, *// 数据库表前缀***'DB\_CHARSET'** => **'utf8'**, *// 数据库编码默认采用utf8***'DB\_DEPLOY\_TYPE'** => 0, *// 数据库部署方式:0 集中式(单一服务器),1 分布式(主从服务器)***'DB\_RW\_SEPARATE'** => **false**, *// 数据库读写是否分离 主从式有效***'DB\_MASTER\_NUM'** => 1, *// 读写分离后 主服务器数量***'DB\_SLAVE\_NO'** => **''**, *// 指定从服务器序号***'DB\_2'**=>**array**(  
 **'DB\_TYPE'** => **'mysql'**, *// 数据库类型* **'DB\_HOST'** => **'localhost'**, *// 服务器地址* **'DB\_NAME'** => **'mysql'**, *// 数据库名* **'DB\_USER'** => **'root'**, *// 用户名* **'DB\_PWD'** => **'root'**, *// 密码* **'DB\_PORT'** => **'3306'**, *// 端口*),

3模型的字段定义

知识点：

3.1字段缓存

为什么要缓存字段信息：数据库中字段信息存入缓存

保存在哪里：缓存目录下的Data/\_fields/

以什么形式保存：数据库名称[.表前缀]+模型名称（小写）.php

如何关闭：DB\_FIELDS\_CACHE

获取模型的字段信息：getDbFields方法

如：**'DB\_FIELDS\_CACHE'** => **true**, *//启用字段缓存*

3.2字段定义

模型的fields属性

使用\_type定义字段数据类型

通过pk属性定义表的主键

* 模型：数据库操作

1. CURD操作

知识点：

* 1. Create（新增insert），使用add方法，注意并不是create方法
  2. Update（更新），使用save方法
  3. Read（读取），使用find、select
  4. Delete（删除），使用delete方法

1. 连贯操作

知识点：

2.1

1. 模型中定义命名范围

知识点：

3.1

1. 使用查询语言

知识点：

4.1

1. ActiveRecord模式

知识点：

5.1

1. 定义字段映射

知识点：

6.1

* 模板（下）
* 模型数据验证和填充