ThinkPHP Framework 1.5
SmartURLs & SEO Support

ThinkPHP 1.5
URL 设计和 SEO 支持

编写：ThinkPHP 文档组

最后更新：2008-12-16

目录

1 概述

本文主要描述了 ThinkPHP 的 URL 设计和 SEO 支持。

2模块和操作

ThinkPHP 里面会根据当前的 URL 来分析要执行的模块和操作。这个分析工作由 URL 调度器来实现，官方内置了 Dispatcher 来完成该调度。在 Dispatcher 调度器中，会根据

http://servername/appName/moduleName/actionName/params

来获取当前需要执行的项目（appName）、模块（moduleName）和操作（actionName），在某些情况下，appName 可以不需要（通常是网站的首页，因为项目名称可以在入口文件中指定，这种情况下，appName 就会被入口文件替代）

每个模块名称是一个 Action 文件，类似于我们平常所说的控制器，系统会自动寻找项目类库 Action 目录下面的相关类，如果没有找到，会尝试搜索应用目录下面的组件类中包含的模块类，如果依然没有，则抛出异常。

而 actionName 操作是首先判断是否存在 Action 类的公共方法，如果不存在则会继续寻找父类中的方法，如果依然不存在，就会寻找是否存在自动匹配的模版文件。如果存在模版文件，那么就直接渲染模版输出。

因此应用开发中的一个重要过程就是给不同的模块定义具体的操作。一个应用如果不需要和数据库交互的时候可以不需要定义模型类，但是必须定义 Action 控制器。

Action 控制器的定义非常简单，只要继承 Action 基础类就可以了，例如：

```
Class UserAction extends Action{
}
```

你甚至不需要定义任何操作方法，就可以完成很多默认的操作，因为 Action 基础类已经为你定义了很多常用的操作方法。例如，我们可以执行（如果已经存在对应模板文件的情况）

http://servername/index.php/User/

http://servername/index.php/User/add

如果你需要增加或者重新定义自己的操作方法，增加一个方法就可以了，例如

```
Class UserAction extends Action{
   // 定义一个 select 操作方法，注意操作方法不需要任何参数
   Public function select(){
    // select 操作方法的逻辑实现
    // ……
    $this->display(); // 输出模板页面
   }
}
```

我们就可以执行http://servername/index.php/User/select/了，系统会自动定位当前操作的模板文件

3 URL 模式

我们在上面的执行过程里面看到的 URL 是大多数情况下，其实 ThinkPHP 支持四种 URL 模式，可以通过设置 URL_MODEL 参数来定义，包括普通模式、PATHINFO、REWRITE 和兼容模式。

普通模式 设置 URL_MODEL 为 0

采用传统的 URL 参数模式

http://<serverName>/appName/?m=module&a=action&id=1

PATHINFO 模式 设置 URL_MODEL 为 1

默认情况使用 URL_PATHINFO 模式，ThinkPHP 内置强大的 PATHINFO 支持，提供灵活和友好 URL 支持。

PATHINFO 模式还包括普通模式和智能模式两种：

普通模式 设置 PATH_MODEL 参数为 1

该模式下面 URL 参数没有顺序，例如

http://<serverName>/appName/m/module/a/action/id/1

http://<serverName>/appName/a/action/id/1/m/module

以上 URL 等效

智能模式 设置 PATH_MODEL 参数为 2 （系统默认的模式）

自动识别模块和操作，例如

http://<serverName>/appName/module/action/id/1/ 或者

http://<serverName>/appName/module,action,id,1/

在智能模式下面，第一个参数会被解析成模块名称（或者路由名称，下面会有描述），第二个参数会被解析成操作（在第一个参数不是路由名称的前提下），后面的参数是显式传递的，而且必须成对出现，例如：

http://<serverName>/appName/module/action/year/2008/month/09/day/21/

其中参数之间的分割符号由 PATH_DEPR 参数设置，默认为"/"，例如我们设置 PATH_DEPR 为^的话，就可以使用下面的 URL 访问

http://<serverName>/appName/module^action^id^1/

注意不要使用":" 和"&"符号进行分割，该符号有特殊用途。

略加修改，就可以展示出富有诗意的 URL，呵呵～

如果想要简化 URL 的形式可以通过路由功能（后面会有描述），在系统不支持 PATHINFO 的情况下，一样可以使用 PATHINFO 模式的功能，只需要传入 PATHINFO 兼容模式获取变量 VAR_PATHINFO，默认值为 s，例如

http://<serverName>/appName/?s=/module/action/id/1/ 会执行和上面的 URL 等效的操作

并且也可以支持参数分割符号的定义，例如在 PATH_DEPR 为~的情况下，下面的 URL 有效

http://<serverName>/appName/?s=module~action~id~1

在 PATH_INFO 模式下面，会把相关参数转换成 GET 变量，以及并入 REQUEST 变量，因此不妨碍应用里面的以上变量获取。

REWRITE 模式 设置 URL_MODEL 为 2

该 URL 模式和 PATHINFO 模式功能一样，除了可以不需要在 URL 里面写入口文件，和可以定义.htaccess 文件外。

例如，我们可以增加如下的.htaccess 内容把所有操作都指向 index.php 文件。

```
<IfModule mod_rewrite.c>
RewriteEngine on
RewriteCond %{REQUEST_FILENAME} !-d
RewriteCond %{REQUEST_FILENAME} !-f
RewriteRule ^(.*)$ index.php/$1 [QSA,PT,L]
</IfModule>
```

兼容模式 设置 URL_MODEL 为 3

兼容模式是普通模式和 PATHINFO 模式的结合，并且可以让应用在需要的时候直接切换到 PATHINFO 模式而不需要更改模板和程序。兼容模式 URL 可以支持任何的运行环境。

4URL 路由

ThinkPHP 支持 URL 路由功能，要启用路由功能，需要设置 ROUTER_ON 参数为 true。开启路由功能后，系统会自动进行路由检测，如果在路由定义里面找到和当前 URL 匹配的路由名称，就会进行路由解析

和重定向。路由功能需要定义路由定义文件，位于项目的配置目录下面，文件名为 routes.php，定义格式：

```
Return Array(
// 第一种方式 常规路由
'RouteName'=>array('模块名称','操作名称','参数定义','额外参数'),
// 第二种方式 泛路由
'RouteName@'=>array(
    array('路由匹配正则','模块名称','操作名称','参数定义','额外参数'),
),
...更多的路由名称定义
)
```

或者

```
$_routes = Array(
'RouteName'=>array('模块名称','操作名称','参数定义','额外参数'),
'RouteName@'=>array(
    array('路由匹配正则','模块名称','操作名称','参数定义','额外参数'),
),
...更多的路由名称定义
)
```

系统在执行 Dispatch 解析的时候，会判断当前 URL 是否存在定义的路由名称，如果有就会按照定义的路由规则来进行 URL 解析。例如，我们启用了路由功能，并且定义了下面的一个路由规则：

```
'blog'=>array('Blog','index','year,month,day','userId=1&status=1')
```

那么我们在执行

http://<serverName>/appName/blog/2007/9/15/的时候就会实际执行 Blog 模块的 index 操作，后面的参

数/2007/9/15/ 就会依次按照 year/month/day 来解析，并且会隐含传入 userId=1 和 status=1 两个参数。

另外一种路由参数的传入是

http://<serverName>/appName/?r=blog&year=2007&month=9&day=15，会执行上面相同的路由解析，该方式主要是提供不支持 PATHINFO 方式下面的兼容实现。其中 r 由 VAR_ROUTER 参数定义，默认是 r。

泛路由支持

新版引入了泛路由支持，提供了对同一个路由名称的多个规则的支持，使得 URL 的设置更加灵活，例如，我们对 Blog 路由名称需要有多个规则的路由：

```
'Blog@'=>array(
    array('/^\/(\d+)(\/p\/\d)?$/','Blog','read','id'),
    array('/^\/(\d+)\/(\d+)/','Blog','archive','year,month'),
    ),
```

第一个路由规则表示解析 Blog/123 这样的 URL 到 Blog 模块的 read 操作

第二个路由规则表示解析 Blog/2007/08 这样的 URL 到 Blog 模块的 archive 操作

泛路由的定义难度就在路由正则的定义上面，其它参数和常规路由的使用一致。

举个简单路由的例子，如果我们有一个 City 模块，而我们希望能够通过类似下面这样的 URL 地址来访问具体某个城市的操作：

http://<serverName>/index.php/City/shanghai/

shanghai 这个操作方法是不存在的，我们给相关的城市操作定义了一个 city 方法，如下：

```
Class CityAction extends Action{
  public function city(){
  // 读取城市名称
  $cityName = $_GET['name'];
  Echo ('当前城市:'.$cityName);
```

```
  }
}
```

接下来我们来定义路由文件，实现类似于

http://<serverName>/index.php/City/shanghai/

这样的解析，路由文件名称是

```
Return array(
  'City'=>array('City','city','name');
);
```

这样，URL 里面所有的 City 模块都会被路由到 City 模块的 city 操作，而后面的第二个参数会被解析成 $_GET['name']

接下来，我们就可以在浏览器里面输入

http://<serverName>/index.php/City/beijing/

http://<serverName>/index.php/City/shanghai/

http://<serverName>/index.php/City/shenzhen/

会看到依次输出的结果是：

当前城市:beijing

当前城市:shanghai

当前城市:shenzhen

5 URL 伪装

ThinkPHP 支持模块和操作伪装，从而构造伪装的 URL 组成，该功能可以用于对已有的数据表定义成新的模块需要。

需要设置 MODULE_REDIRECT 参数，格式：

Module1:TrueModuleName1,Module2:TrueModuleName2

可以一次定义多个模块伪装，用逗号分割，伪装模块和真实模块名之间用冒号隔开。定义模块伪装后，伪装模块和真实模块名称都可以访问，例如：

假如定义了下面的参数，‘MODULE_REDIRECT’=>’User:Member,Info:Member_info’，那么当我们在浏览器里面运行的时候，

http://<serverName>/AppName/User/和 http://<serverName>/AppName/Member/等效

http://<serverName>/AppName/Info/和 http://<serverName>/AppName/Member_info/等效

操作伪装的定义方式类似，设置 ACTION_REDIRECT 参数即可。

假如定义了下面的参数，‘ACTION_REDIRECT’=>’list:index’，那么当我们在浏览器里面运行的时候，

http://<serverName>/AppName/User/list/和 http://<serverName>/AppName/User/index/等效

6 URL 组装

系统提供了 URL 方法来根据当前设置的 URL 模式来组装需要的 URL 地址，例如：

url(’read’, ’ Blog’); // 表示 Blog 模块的 read 操作 URL 地址

url(’edit’, ’ Blog’, ’’, ’ App’,array(‘id’=>3)); // 表示 Blog 模块的 edit 操作 URL 地址 后面的参数表示编辑 id 为 3 的记录

如果在模板里面，可以使用下面的模板标签来生成 URL

<url module=’Blog’ action=’read’ />

7 隐藏 index.php

去掉 URL 里面的 index.php 是为了 SEO 的需要，需要服务器开启 URL_REWRITE 模块。

下面的配置过程可以参考下：

✧ httpd.conf 配置文件中加载了 mod_rewrite.so 模块

✧ AllowOverride None 将 None 改为 All

✧ 确保 URL_MODEL 设置为 2

✧ 把.htaccess 文件放到入口文件的同级目录下

```
<IfModule mod_rewrite.c>
RewriteEngine on
RewriteCond %{REQUEST_FILENAME} !-d
RewriteCond %{REQUEST_FILENAME} !-f
RewriteRule ^(.*)$ index.php/$1 [QSA,PT,L]
</IfModule>
```

8 伪静态设计

系统支持伪静态 URL 设置，可以通过设置 HTML_URL_SUFFIX 参数随意在 URL 的最后增加你想要的静态后缀，而不会影响当前操作的正常执行。例如，我们设置 HTML_URL_SUFFIX 为 .shtml 的话，我们可以把下面的 URL

http://<serverName>/Blog/read/id/1

变成

http://<serverName>/Blog/read/id/1.shtml

后者更具有静态页面的 URL 特征，但是具有和前面的 URL 相同的执行效果。

9 默认模块和操作

如果使用 http://<serverName>/index.php，没有带任何模块和操作的参数，系统就会寻找默认模块和默

认操作，通过 DEFAULT_MODULE 和 DEFAULT_ACTION 来定义，系统的默认模块设置是 Index 模块，默认操作设置是 index 操作。也就是说

http://<serverName>/index.php 和

http://<serverName>/index.php/Index 以及

http://<serverName>/index.php/Index/index 等效。

可以在项目配置文件中修改默认模块和默认操作的名称。

10 空操作

空操作是指系统在找不到指定的操作方法的时候，会定位到空操作（_empty）方法来执行，利用这个机制，我们可以实现错误页面和一些 URL 的优化。

例如，我们前面用 URL 路由实现了一个城市切换的功能，下面我们用空操作功能来重新实现。

我们只需要给 CityAction 类定义一个_emtpy （空操作）方法：

```
Class CityAction extends Action{
  Public function _empty(){
  // 把所有城市的操作都解析到 city 方法
  // 注意 city 方法本身是 protected 方法
  $cityName = ACTION_NAME;
  $this->city($cityName);
  }
  Protected function city($name){
  // 和$name 这个城市相关的处理
  Echo ('当前城市:'.$name);
  }
}
```

接下来，我们就可以在浏览器里面输入

http://<serverName>/index.php/City/beijing/
http://<serverName>/index.php/City/shanghai/
http://<serverName>/index.php/City/shenzhen/
会看到依次输出的结果是：

当前城市:beijing

当前城市:shanghai

当前城市:shenzhen

可以看出来，和用 URL 路由实现的效果是一样的，而且不需要定义路由定义文件。

11 空模块

空模块的概念是指当系统找不到指定的模块名称的时候，系统会尝试定位空模块(EmptyAction)，利用

这个机制我们可以用来定制错误页面和进行 URL 的优化。

现在我们把前面的需求进一步，把 URL 由原来的

http://<serverName>/index.php/City/shanghai/
变成

http://<serverName>/index.php/shanghai/
这样更加简单的方式，如果按照传统的模式，我们必须给每个城市定义一个 Action 类，然后在每个

Action 类的 index 方法里面进行处理。

可是如果使用空模块功能，这个问题就可以迎刃而解了。

我们可以给项目定义一个 EmptyAction 类

```
Class EmptyAction extends Action{
  Public function index(){
  // 根据当前模块名称来判断要执行哪个城市的操作
```

```
    $cityName = MODULE_NAME;
    $this->city($cityName);
    }
    Protected function city($name){
    // 和$name 这个城市相关的处理
    Echo ('当前城市:'.$name);
    }
}
```

接下来，我们就可以在浏览器里面输入

http://<serverName>/index.php/beijing/
http://<serverName>/index.php/shanghai/
http://<serverName>/index.php/shenzhen/

会看到依次输出的结果是：

当前城市:beijing

当前城市:shanghai

当前城市:shenzhen