[PHP]\_TP5.0\_FastAdmin\_控制器；

--王尔贝(我是害虫)

(整理 <https://doc.fastadmin.net/docs/controller.html>)

--

**一、基类控制器**

FastAdmin中定义了三个基类控制器，分别位于如下目录。

application/common/controller/Api.php //API接口基类控制器；

application/common/controller/Backend.php //后台基类控制器；

application/common/controller/Frontend.php //前台基类控制器；

这里我们主要对Backend.php这个控制器基类做详细解读，因为我们后台管理的所有控制器都继承于它，而Frontend.php功能和Backend.php功能类似，而Api.php控制器主要用于API接口开发，这里就不再进行过多介绍。

--

**二、基础结构**

在后台中我们的控制器都必须继承自\app\common\controller\Backend这个基类，其中控制器的index/add/edit/del/multi/recyclebin/destroy/restore/import/selectpage全都是可选的方法，基类的这些方法使用traits进行引入，具体位置在application/admin/library/traits/Backend.php中，CRUD生成的标准控制器如下描述。

<?php

namespace app\admin\controller;

/\*\*

\* 文章管理

\*

\* @icon fa fa-list

\* @remark 用于统一管理网站的所有文章

\*/

class Article extends Backend

{

protected $model = null;

protected $noNeedLogin = [];

protected $noNeedRight = ['selectpage'];

public function \_initialize()

{

parent::\_initialize();

}

/\*\*

\* 默认生成的控制器所继承的父类中有index/add/edit/del/multi/destroy/restore/recyclebin八个方法，

\* 因此在当前控制器中可不用编写增删改查的代码,如果需要自己控制这部分逻辑，

\* 需要将application/admin/library/traits/Backend.php中对应的方法复制到当前控制器,然后进行修改。

\*/

}

基类中所定义的方法如下，以下方法都是通过application/admin/library/traits/Backend.php引入的。

class Backend extends Controller{

/\*\*

\* 查看

\*/

public function index(){}

/\*\*

\* 添加

\*/

public function add($ids = ""){}

/\*\*

\* 编辑

\*/

public function edit($ids = ""){}

/\*\*

\* 删除

\*/

public function del($ids = ""){}

/\*\*

\* 批量更新

\*/

public function multi($ids = ""){}

/\*\*

\* 回收站

\*/

public function recyclebin(){}

/\*\*

\* 真实删除

\*/

public function destroy($ids = ""){}

/\*\*

\* 还原

\*/

public function restore($ids = ""){}

/\*\*

\* 导入

\*/

protected function import(){}

/\*\*

\* 下拉筛选

\*/

public function selectpage()

{

return parent::selectpage();

}

}

我们在开发过程中最好注释好每一个控制器和控制器的方法，因为我们后期可以使用php think menu -c all-controller一键生成后台管理的菜单，注释支持@icon/@remark/@internal这三个属性，分别表示图标/备注/忽略，如果是protected或private的方法在后期一键生成时会自动忽略。控制器支持目录层级，如果在使用目录层级的时候，请注意当前控制器的命名空间，比如当前控制器文件位置是application/admin/controller/member/Teacher.php，当Teacher.php的命名空间请务必是"namespace app\admin\controller\member;"，如果命名空间不对会报找不到控制器的错误。

--

**三、属性和方法**

**//// 属性**

当我们的控制器继承自app\common\controller\Backend以后，我们就可以使用以下属性。

/\*\*

\* 无需登录的方法,同时也就不需要鉴权了

\* @var array

\*/

protected $noNeedLogin = [];

/\*\*

\* 无需鉴权的方法,但需要登录

\* @var array

\*/

protected $noNeedRight = [];

/\*\*

\* 布局模板

\* @var string

\*/

protected $layout = 'default';

/\*\*

\* 权限控制类

\* @var Auth

\*/

protected $auth = null;

/\*\*

\* 快速搜索时执行查找的字段

\*/

protected $searchFields = 'id';

/\*\*

\* 是否是关联查询

\*/

protected $relationSearch = false;

/\*\*

\* 是否开启数据限制

\* 支持auth/personal

\* 表示按权限判断/仅限个人

\* 默认为禁用,若启用请务必保证表中存在admin\_id字段

\*/

protected $dataLimit = false;

/\*\*

\* 数据限制字段

\*/

protected $dataLimitField = 'admin\_id';

/\*\*

\* 是否开启Validate验证

\*/

protected $modelValidate = false;

/\*\*

\* 是否开启模型场景验证

\*/

protected $modelSceneValidate = false;

/\*\*

\* Multi方法可批量修改的字段

\*/

protected $multiFields = 'status';

//

我们可以直接在当前控制器使用$this->属性名来调用所支持的属性，也支持直接在当前控制器定义相关属性来覆盖父类的属性，同时TP5中\think\Controller所支持的属性也全部支持。

//

**//// 方法**

基类app\common\controller\Backend中所支持的方法如下所述。

/\*\*

\* 加载语言文件

\* @param string $name

\*/

protected function loadlang($name)

{

}

/\*\*

\* 渲染配置信息

\* @param mixed $name 键名或数组

\* @param mixed $value 值

\*/

protected function assignconfig($name, $value = '')

{

}

/\*\*

\* 生成查询所需要的条件,排序方式

\* @param mixed $searchfields 快速查询的字段

\* @param boolean $relationSearch 是否关联查询

\* @return array

\*/

protected function buildparams($searchfields = null, $relationSearch = null)

{

}

/\*\*

\* 获取数据限制的管理员ID

\* 禁用数据限制时返回的是null

\* @return mixed

\*/

protected function getDataLimitAdminIds()

{

}

/\*\*

\* Selectpage的实现方法

\*

\* 当前方法只是一个比较通用的搜索匹配,请按需重载此方法来编写自己的搜索逻辑,$where按自己的需求写即可

\* 这里示例了所有的参数，所以比较复杂，实现上自己实现只需简单的几行即可

\*

\*/

protected function selectpage()

{

}

我们可以直接在当前控制器使用$this->方法名()来调用所支持的方法，同时TP5中\think\Controller所支持的方法也全部支持。

--

**四、数据限制**

在后台开发的过程中经常会有这样的一个需求，每个管理员单独管理自己添加的数据或单独管理自己下级管理员添加的数据，管理员之间的数据是不相通的，每个管理员看到的数据是不同的。在FastAdmin中可以很方便的实现此功能。

//

// 首先我们需要在当前控制器添加以下两个属性

protected $dataLimit = 'auth'; //默认基类中为false，表示不启用，可额外使用auth和personal两个值；

protected $dataLimitField = 'admin\_id'; //数据关联字段,当前控制器对应的模型表中必须存在该字段；

//

$dataLimit = false; //表示不启用，显示所有数据;

$dataLimit = 'auth'; //表示显示当前自己和所有子级管理员的所有数据;

$dataLimit = 'personal'; //表示仅显示当前自己的数据;

//

$dataLimitField字段默认为admin\_id，请注意添加该字段类型为int(10)。通过以上配置后，在列表数据的时候将默认添加条件过滤不属自己权限的数据，在添加时会自动维护admin\_id的数据，在编辑、删除的时候会自动控制权限避免越权操作。如果需要将原有的数据加入到FastAdmin后台管理权限控制当中，比如已有的数据已经有标识归属，但这个归属体系并非是FastAdmin的后台管理员体系。在这个时候我们就需要重写基类的getDataLimitAdminIds方法，将此方法返回数据标识的归属ID数组集合，这样即可使用FastAdmin的后台管理权限进行管理。

--

**五、关联查询**

目前FastAdmin后台index方法支持一对一关联查询，比如我们一篇文章有归属分类，我们在列出数据时需要同时列表文章分类名称。

//

// 首先我们需要在当前控制器中添加以下属性

protected $relationSearch = true;

//

// 然后我们修改控制器的index方法

public function index()

{

if ($this->request->isAjax())

{

list($where, $sort, $order, $offset, $limit) = $this->buildparams();

$total = $this->model

->with(["category"])

->where($where)

->order($sort, $order)

->count();

$list = $this->model

->with(["category"])

->where($where)

->order($sort, $order)

->limit($offset, $limit)

->select();

$result = array("total" => $total, "rows" => $list);

return json($result);

}

return $this->view->fetch();

}

//

// 然后在控制器对应的model(非关联model)中添加以下代码

public function category()

{

// 设置预载入查询方式为JOIN方式

return $this->belongsTo('Category', 'category\_id')->setEagerlyType(0);

}

更多的关联用户可以参考TP5关联模型的章节：关联模型( <https://www.kancloud.cn/manual/thinkphp5/142357> )。

//

// 我们在控制器对应的JS中可以直接使用category.id、category.name等关联表的字段

// 初始化表格

table.bootstrapTable({

url: $.fn.bootstrapTable.defaults.extend.index\_url,

columns: [

[

{field: 'state', checkbox: true, },

{field: 'id', title: 'ID', operate: '='},

{field: 'title', title: \_\_('Title'), operate: 'LIKE %...%'},

{field: 'category.image', title: \_\_('Image'), operate: false, formatter: Table.api.formatter.image},

{field: 'category.name', title: \_\_('Name'), operate: '='},

{field: 'ip', title: \_\_('IP'), operate: '='},

{field: 'createtime', title: \_\_('Create time'), formatter: Table.api.formatter.datetime, operate: 'RANGE', addclass: 'datetimerange'},

{field: 'operate', title: \_\_('Operate'), table: table, events: Table.api.events.operate, formatter: Table.api.formatter.operate}

]

],

});

--

**六、数据校验**

在FastAdmin中默认的add/edit方法可以使用模型验证，验证器位于application/admin/validate/模型名.php中，模型验证默认是关闭的状态，如果需要启用，我们需要在当前控制器定义以下属性。

// 启用数据验证

protected $modelValidate = true; //是否开启Validate验证，默认是false关闭状态;

protected $modelSceneValidate = true; //是否开启模型场景验证，默认是false关闭状态;

当开启模型验证后，我们的添加和修改操作都会首先进行模型验证，验证不通过将会抛出错误信息，具体的模型验证规则可以参考TP5官方文档的模型验证规则：<https://www.kancloud.cn/manual/thinkphp5/129355>。场景验证可以参考TP5场景验证章节：<https://www.kancloud.cn/manual/thinkphp5/129322>。

--

**七、权限控制**

// 在基类中我们有定义以下两个属性;

protected $noNeedLogin = []; //无需登录的方法,同时也就不需要鉴权了;

protected $noNeedRight = []; //无需鉴权的方法,但需要登录;

比如我们有定义一个方法mywork，而这个方法是不需要登录即可访问的，则我们需要在当前的控制器定义“protected $noNeedLogin = ['mywork'];”。比如我们有定义一个方法mytest，而这个方法是需要登录后任何管理员都可以访问，则我们需要在当前的控制器定义“protected $noNeedRight = ['mytest'];”。如果我们需要动态定义，请务必放在调用父类的\_initialize方法之前，否则是不会生效的。

--

**八、视图渲染**

基类app\common\controller\Backend会默认渲染以下几个对象到视图中。

// 渲染站点配置

$this->assign('site', $site);

// 渲染配置信息

$this->assign('config', $config);

// 渲染权限对象

$this->assign('auth', $this->auth);

// 渲染管理员对象

$this->assign('admin', Session::get('admin'));

我们可以在视图中使用{$site.name}、{$config.modulename}、{$auth.id}、{$admin.username}来获取我们所需要的数据。

//

// $site所支持的数据对应为application/extra/site.php(后台配置保存入数据库后写入配置文件)

// $config所支持的数据

'site' => $site中的'name', 'cdnurl', 'version', 'timezone', 'languages'字段,

'upload' => application/extra/upload.php中数据,

'modulename' => 'admin',

'controllername' => 控制器名,

'actionname' => 方法名,

'jsname' => 控制器JS所加载的路径,

'moduleurl' => 后台module的链接,

'language' => 当前语言,

'fastadmin' => application/config.php中fastadmin的配置

//

$auth是一个对象，所对应的类是application/admin/library/Auth.php。

$admin是一个当前管理员登录的session数据，存储有管理员的用户名、昵称、ID、头像等信息。

//

// 如果我们需要在JS中使用以上数据，则使用如下方式来获取相关的配置信息。

Config.site.name

Config.modulename

//

如果我们需要自己在控制器中透传数据到JS中去，则可以使用控制器的assignconfig方法来透传，使用“$this->assignconfig('demo', ['name'=>'名称']);”，然后我们就可以在JS中使用“Config.demo.name”来获取对应的数据。

--

**九、模板布局**

控制器默认全部采用模板布局，因此我们的页面都会自动加上头部和尾部，如果我们有特殊的页面不需要采用模板布局，我们可以使用“$this->view->engine->layout(false);”来关闭当前方法的模板布局。如果我们需要使用自己的模板布局，在当前控制器定义“protected $layout = '布局模板';”即可。请注意如果采用了自己的模板布局或禁用了模板布局，将无法使用FastAdmin的JS按需加载和Config变量访问。

--