# vendor和import的区别

import 方法是 ThinkPHP 内建的统一类库加载和文件导入机制，完全可以替代 PHP 的 require 和 include 方法。一般情况下，由于框架内部采用了自动加载方式，所以大多数情况下面不需要用户手动导入类库文件，通常用于项目类库、扩展类库和第三方类库的导入支持。

import 方法用于导入当前需要而 ThinkPHP 又不能自动载入的文件，如系统基类 ORG 与 Com 目录下的扩展基类，自己的类库等。至于第三方类库，也可以使用 import 方法导入，但建议使用 vendor 方法导入。

import 语法：boolen import(class, baseUrl, ext) ；import('类库名', '起始路径', '类库后缀')。

|  |  |
| --- | --- |
| class | 必须，表示要导入的类库，采用命名空间的方式。 |
| baseUrl | 可选，表示导入的基础路径，省略的话系统采用默认的规则。 |
| ext | 可选，表示导入的类库后缀，默认是 .class.php 。 |

vendor 方法专门用于导入第三方类库，默认的起始路径为 ThinkPHP系统目录/Vendor，默认后缀为 .php。

# . 符号文件导入

在 import 方法中，. 符号是用于表示目录层次的，如果需要导入的文件名包含有 . 符号，如 User.Info.class.php ，那么需要以 # 号来代替 . 号：import("@.Action.User#Info");

# 同名文件导入冲突

import 方法具有缓存和检测机制，相同的文件不会重复导入，如果发现导入了不同的位置下面的同名类库文件，系统会提示冲突，例如：

import("Think.Util.Array");

import("App.Util.Array");

上面的情况导入会产生引入两个同名的 Array.class.php 类，即使实际上的类名可能不存在冲突，但是类名和文件名是一致的，所以系统会抛出类名冲突的异常，并终止执行。

**框架目录（TP5/thinkphp/）->基类库（library/）->核心类库（Think/）**

# 导入 ThinkPHP 基类库文件

import 方法导入ThinkPHP 基类库文件，约定导入 Think、ORG、Com 包以 ThinkPHP基类库为相对起始目录：

// 导入 ThinkPHP系统目录/Lib/Think/Util/Session.class.php 文件

import("Think.Util.Session");

// 导入 ThinkPHP系统目录/Lib/ORG/Util/Page.class.php 文件

import("ORG.Util.Page");

可以支持多级目录，例如：

import('Think.Util.U1.ClassA');

import('Think.Util.U1.A2.ClassB');

通过import方法导入类库后，就可以进行类库的实例化操作了。

# 导入扩展类库

扩展类库位于Extend/Library目录下面，这是系统的公共扩展类库目录，目前支持的扩展类库包只有ORG和Com包。

import('ORG.Util.Image');

import('Com.Sina.OAuth');

会导入扩展目录下面的第三方类库（分别是Extend/Library/ORG/Util/Image.class.php和Extend/Library/Com/Sina/OAuth.class.php 类库文件），第三方类库包只能支持ORG和Com两种，下面的子目录可以随意添加。

# 导入项目应用类库

如果没有指定起始导入路径的话，类库包Think、ORG、Com之外的都会被认为是导入项目应用类库，例如：

import("MyApp.Action.UserAction");

import("MyApp.Model.InfoModel");

// 导入 MyApp项目 Lib/Action/UserAction.class.php 文件

import("MyApp.Action.UserAction");

// 导入 MyApp项目 Lib/ORG/Page.class.php 文件

import("MyApp.ORG.Page");

// 导入 MyApp项目 Lib/ORG/My/Image.class.php 文件

import("MyApp.ORG.My.Image");

表示导入MyApp项目的UserAction和InfoModel类库文件，由于通常，我们都是导入当前项目下面的类库，所以可以简写成：

import("@.Action.UserAction");

import("@.Model.InfoModel");

@符号表示导入当前项目下面的类库，这种方式也一定程度上方便了项目类库的代码移植，如果项目名称改变或者移动到其它项目下面的时候，写法不需要改变。

如果是导入其他项目的类库，必须指定项目名称：

import("OtherApp.Action.UserAction");

注意：使用这种方式导入其他项目的类库时，必须保证两个项目的目录是平级的，否则需要指定 baseUrl 参数。

# 导入非标准类库文件

这里所说的非标准类库文件，主要是指位于特殊位置或者非.class.php后缀的类库文件。像导入基类库、扩展类库和项目类库都是基于框架规范的目录下面，如果我们需要导入项目的Common目录下面的MyClass.php文件，则可以采用：

import('Common.MyClass',APP\_PATH,'.php');

import('MyClass',APP\_PATH.'Common','.php');

或者要导入当前目录下面的RBAC类库

import("RBAC.AccessDecisionManager",dirname(\_\_FILE\_\_),".php");

还有一种特殊情况，是类库命名的特殊性。按照系统的规则，import方法是无法导入具有点号的类库文件的，因为点号会直接转化成斜线，例如我们定义了一个名称为User.Info.class.php 的文件的话，采用：

import("ORG.User.Info");

方式加载的话就会出现错误，导致加载的文件不是ORG/User.Info.class.php 文件，而是ORG/User/Info.class.php 文件，这种情况下，我们可以使用：import("ORG.User#Info");来导入。

# 第三方类库导入

ThinkPHP 的基类库都是以.class.php 为后缀的，这是系统内置的一个约定，当然也可以通过 import 的参数来控制， 为了更加方便引入其他框架和系统的类库， 系统还提供了一个import方法的别名vendor，专门用于导入第三方类库，并且默认的起始目录和类文件后缀有区别。第三方类库位于系统扩展目录下的Vendor 目录， 例如，我们把 Zend 的 Filter\Dir.php 放到 Vendor 目录下面，这个时候 Dir 文件的路径就是 Vendor\Zend\Filter\Dir.php，我们使用vendor 方法导入只需要使用：

Vendor('Zend.Filter.Dir');就可以导入Dir类库了。  
Vendor方法也可以支持和import方法一样的基础路径和文件名后缀参数，例如：

Vendor('Zend.Filter.Dir',dirname(\_\_FILE\_\_),'.class.php');

# 别名导入

除了命名空间的导入方式外，import方法还可以支持别名导入，要使用别名导入，首先要定义别名，我们可以在项目配置目录下面增加alias.php 用以定义项目中需要用到的类库别名，例如：

return array(

'rbac' =>LIB\_PATH.'Common/Rbac.class.php',

'page' =>LIB\_PATH.'Common/Page.class.php',

);

那么，现在就可以直接使用：

import("rbac");

import("page");

导入Rbac和Page类，别名导入方式禁止使用import方法的第二和第三个参数，别名导入方式的效率比命名空间导入方式要高效，缺点是需要预先定义相关别名。可以为某些需要的类库定义别名，那么无需定义自动加载路径也可以快速的自动加载。

# 别名导入

如果在别名定义文件定义了别名：

'AdvModel' => THINK\_PATH.'/Lib/Think/Core/Model/AdvModel.class.php',

可以使用 import 方法以别名的方式导入对应的类库文件：

import('AdvModel');

系统默认的别名定义文件位于 ThinkPHP 系统目录下的 Commonalias.php，也可以定义项目自己的别名文件。

# **TP5.0中如何使用extend和vendor的第三方类库**

（一）  
1、首先在extend目录下新建文件夹my,然后新建Test.php,代码如下

<?php

namespace my;

class Test {

public function hello()

{

echo 'hello,world';

}

}

?>

2、在controller新建M.php,代码如下

`<?php

namespace app\admin\Controller;

use think\Controller;

use think\loader;

class M extends Controller{

public function one(){

Loader::import('my\Test', EXTEND\_PATH);

$a=new \my\Test;

$a->hello();

}

}

?>

网页输出hello,world，起关键作用的是use think\loader;和 Loader::import(‘my\Test’, EXTEND\_PATH);  
（二）引用vendor类库，把my文件夹复制到vendor下，重写M.php里的one方法

public function one(){

$a = vendor ( 'my.test',VENDOR\_PATH,'.php');

$b = new \Test();

$b->hello();

}