ThinkPHP Framework 1.5
Class import & AutoLoad

ThinkPHP 1.5
类库导入和自动加载

编写：ThinkPHP 文档组

最后更新：2008-12-16

目录

1 概述

ThinkPHP 采用了独特的类库导入机制，要掌握 ThinkPHP 的开发，了解如何导入类库文件和项目文件非常重要。清楚哪些情况系统是会自动加载类库的，会给项目开发带来更多的方便。

2 文件约定

ThinkPHP 在类库文件的命名方面有一些约定，遵守这些约定将会让开发更加方便，但是也完全可以通过配置和参数来改变这些默认的约定，这些约定包括：

- 所有类库文件必须使用.class.php 作为文件后缀；
- 类名和文件名保持一致（包括大小写一致）；

3 系统基类库

ThinkPHP 框架的类库文件主要分为两个部分：系统基类库和项目类库。

基类库位于框架系统目录下面的 Lib 目录，这些类库除了框架运行所需要的核心类库，还包括网站和项目开发过程中经常会用到的常用工具类。目前主要包含 Think 核心类库、ORG 扩展类库、Com 扩展类库，其中 Think 核心基类库的作用是完成框架的通用性开发而必须的基础类和常用工具类等，包含有：

Think.Core 核心类库包

Think.Db 数据库类库包

Think.Util 系统工具类库包

Think.Template 内置模板引擎类库包

Think.Exception 异常处理类库包

ORG 和 Com 类库包主要用于基类库的扩展，ORG 类库包主要是第三方的公共类库，而 Com 类库包通常用于公司或者大项目，或者多年的开发经验而积累形成的类库包，主要是内部或者局部范围使用。默认情况下，ORG 类库包是已经有创建的，并且也包含了一些常用的类库，而 Com 类库包是需要自己来创建的，因此你在系统的 Lib 目录下面是看不到 Com 目录的。

4 项目类库

除了基类库之外，在项目开发过程中我们还会经常使用的是项目类库。项目类库文件位于项目目录下面的 Lib 目录，通常包括项目的控制器类、模型类以及一些项目的公共类等等。通常 Action 和 Model 目录是项目必须的，其他可以根据情况自行增加。

5 导入基类库

ThinkPHP 模拟了 Java 的类库导入机制，采用 import 方法进行类文件的加载。Import 方法是 ThinkPHP 内建的类库和文件导入方法，提供了方便和灵活的文件导入机制，完全可以替代 PHP 的 require 和 include 方法。例如：

```
Import("Think.Util.Session");
Import("App.Model.UserModel");
```

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("ORG.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，但是按照 ThinkPHP 的规范，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。

Import 方法有一个别名是 using，对于对.Net 开发情有独钟的人来说是个不错的选择。

注意：在 Unix 或者 Linux 主机下面是区别大小写的，所以在使用 import 方法或者 using 方法的时候要注意目录名和类库名称的大小写，否则会引入文件失败。

6 导入项目类库

对于 Import 方法，系统会自动识别导入类库文件的位置，ThinkPHP 的约定是 Think、ORG、Com 包的导入以系统基类库为相对起始目录，否则就认为是项目应用类库为起始目录。因此，要导入项目类库文件也很简单，使用下面的方式就可以了，和导入基类库的方式看起来差不多：

```
Import("MyApp.Action.UserAction");
Import("MyApp.Model.InfoModel");
```

通常我们都是在当前项目里面导入所需的类库文件，所以，我们可以使用下面的方式来简化代码

```
Import("@.Action.UserAction");
Import("@.Model.InfoModel");
```

除了看起来简单一些外，还可以方便项目类库的移植。

如果要在当前项目下面导入其他项目的类库，必须保证两个项目的目录是平级的，否则无法使用

```
Import("OtherApp.Model.GroupModel");
```

的方式来加载其他项目的类库。

我们知道，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");
```

来表示。

7 匹配导入

除了正常的导入操作为，还支持模式匹配导入和多文件导入。例如：

导入 Think.Util 目录下面的所有类文件

```
Import("Think.Util.*");
Import("App.Model.*");
```

注意：使用子目录引入的方式，如果目录下面文件较多会给系统带来较大的目录遍历开销。

Import 方法具有智能的模式匹配导入机制，例如下面的例子使用了更高级的匹配模式导入：

```
Import("Think.*.Array");
Import("ORG.Util.Array*");
Import("ORG.Util.*Map");
Import("App.Util.?Tool");
```

8 可选参数

对于 Import 方法，系统会自动识别导入类库文件的位置，如果是其它情况的导入，需要指定 baseUrl 参数，也就是 import 方法的第二个参数。例如，要导入当前文件所在目录下面的 RBAC/AccessDecisionManager.class.php 文件，可以使用：

```
Import("RBAC.AccessDecisionManager",dirname(__FILE__));
```

ThinkPHP 的类文件命名规则是以.class.php 作为后缀，所以默认的导入只会匹配.class.php 为后缀的文件，如果你使用的第三方类库文件没有使用.class.php 作为类文件命名规范，那么需要指定后缀，例如，导入 Com.Zend.Search 目录下面的所有类文件,但不包括子目录，并且这些类库文件的后缀是.php，我们就需要指定 import 方法的第三个参数 ext：

```
Import("Com.Zend.Search.*",'','.php');
```

9 类库扩展

系统基类库可以很方便的进行扩展，你可以在 ORG 类库目录下面添加自己需要的类库（ThinkPHP 建议所有的类库文件用 class.php 作为后缀，并且文件名和类名相同），你甚至还可以创建属于自己企业的类库，只需要在 ThinkPHP\Lib\目录下面创建 Com 目录，然后在里面增加相应的类库就可以方便的使用 import 方法导入了。

例如，我们在 ThinkPHP\Lib\Com\下面创建了 Sina 目录，并且放了 Util\UnitTest.class.php 类库文件，可以使用下面的方式导入

```
import('Com.Sina.Util.UnitTest');
```

如果你直接使用的是第三方的类库包，而且类名和后缀和 Tp 的默认规则不符合，那我们建议你放到 Vendor 目录下面，使用 vendor 方法来导入（参考后面的导入第三方类库）。

项目类库的扩展，和基类库的扩展一样，我们可以在项目类库目录增加你想要的子目录，例如，我们在 MyApp 的项目目录下面增加 Common 和 Util 目录，就可以这样加载这些目录下面的类库文件了：

```
import('MyApp.Util.UnitTest');
import('@.Common.CommonUtil');
```

10 导入第三方类库

我们知道 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，系统增加了导入第三方类库的功能。第三方类库统一放置在系统的 Vendor 目录下面，并且使用 vendor 方法导入，其参数和 import 方法是一致的，只是默认的值有所变化。

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 方法导入就是：

```
Vendor('Zend.Filter.Dir');
```

11 自动加载

在很多情况下，我们可以利用框架的自动加载功能，完成类库的加载工作，而无需我们手动导入所需要使用的类库。这些情况包括：

✧Think.Core 核心包的类库（框架的执行过程中会自动导入）

✧Think.Util 工具包下面的类库文件

✧当前项目下面的 Action 类库和 Model 类库文件

如果要增加其他的自动导入，我们可以通过配置 **AUTO_LOAD_PATH** 参数来完成。

AUTO_LOAD_PATH 参数是用于设置框架的自动导入的搜索路径的，默认的配置是 Think.Util.，因此才会实现自动导入 Think.Util 工具类库。例如，我们需要增加 ORG.Util.路径作为类库搜索路径，可以使用：

```
'AUTO_LOAD_PATH'=> 'Think.Util.,ORG.Util.',
```

多个搜索路径之间用逗号分割，并且注意定义的顺序代表了搜索的顺序。

如果你的项目需要加载一些公共的类库文件，而又不想在每个类库文件里面显式的使用 import 方法加载，还可以配置 **AUTO_LOAD_CLASS** 参数，例如：

```
'AUTO_LOAD_CLASS'=> '@.Action.CommonAction,@.Model.CommonModel',
```

这样，就会在应用执行的时候自动加载 CommonAction.class.php 和 CommonModel.class.php。