Yii 2.0最显著的特征之一就是引入了命名空间，因此对于自定义类的引入方式也同之前有所不同。这篇文章讨论一下如何利用Yii 2.0的自动加载机制，向系统中引入自定义类和命名空间。本文旨在抛砖引玉，如果有理解不当敬请指正，欢迎大家把自己的方法拿出来分享。  
我们希望被引入的类应该达成一下两点：

* 在应用中的任意位置可以使用该类名或命名空间，而不用显式调用require()/include()。
* 利用Yii的autoloader，仅在类被调用时加载，以遵循Yii按需加载的原则，节省资源。
* 我们使用Yii 2.0基础模板作为演示环境，项目根目录命名为basic（后文中会写成/），这是根目录结构：
* basic
* ├── assets
* ├── commands
* ├── config
* ├── controllers
* ├── mail
* ├── models
* ├── runtime
* ├── tests
* ├── vendor
* ├── views
* └── web

加载自定义类

I. 定义类文件

建立目录 /libs 并建立文件Freedom.php。

<?php

**class** **Freedom**

{

**public** **static** **function** **yell**()

{

**echo** "I am FREE!";

}

}

### II. 向Yii::$classMap添加映射

打开配置文件/config/web.php，在文件头部向Yii::$classMap属性添加类映射。

<?php

...

Yii::$classMap['Freedom'] = '@app/libs/Freedom.php';

...

$config = [

...

];

**return** $config;

**注意：** 不要对Yii::$classMap使用=直接赋值，因为该属性中定义了Yii的一些核心类映射，直接赋值会导致这些映射丢失而Yii autoloader加载不到核心类。

### III. 使用自定义类

见证奇迹的时刻。在系统中尝试调用这个类，我们使用SiteController::actionIndex()为例。

<?php

...

**use** **Freedom**; // 别忘了导入这个类，或者在后面调用的时候使用"\Freedom"。

**class** **SiteController** **extends** **Controller**

...

**public** **function** **actionIndex**()

{

Freedom::yell();

}

}

好了，刷新一下试试。

### IV. 如果你还关心为什么

那现在我们来需要介绍一下Yii::$classMap究竟是个毛。这货实际上是一个关联数组，数组键为“去掉前导反斜线的完全权限定类名”，对应值为定义了该类的文件路径，其中文件路径支持路径别名。在代码中调用到尚未加载的类时，Yii的audoloader会扫描这个属性以获得需要加载的类文件名。

所以，我们把刚刚定义的类加入到这个映射数组中，它就可以被Yii延迟加载了。事实上我们可以在任何位置添加这个映射，只要在目标类被调用之前就可以。应用的主配置文件是一个比较理想的位置，因为配置文件加载在Yii.php之后，可以在其中访问到Yii类（有兴趣的同学可以去看一下入口脚本），而且配置数据可以集中在一个文件里。

另外，由于我们定义的类在根命名空间下，所以“去掉前导反斜线的完全权限定类名”就只剩下Freedom了。如果你的类使用了命名空间，只需要在数组键里写上完全限定名称就行了(e.g. ['custom\classes\Freedom'])。

加载整个命名空间

有时候我们需要写一组相互关联的类，如果这些类存在依赖关系的话像上面这样给每个类配置映射会……非常不体面。如果你定义命名空间下的类时遵循 [PSR-4](http://www.php-fig.org/psr/psr-4) 标准，我们可以一次引入整个命名空间。  
这次我们要使用的属性是\yii\base\Application::$aliases。它也是一个关联数组，将一个路径别名映射到一个目录或者另一个已经存在的路径别名。其中数组键是要指定的别名，对应值是目标路径。  
我们只需要在建立一个命名空间别名，把它映射到保存这个命名空间下所有类的根目录，就可以了。当然这个根目录以下的文件结构和类定义要遵循[PSR-4](http://www.php-fig.org/psr/psr-4)，不然autoloader是找不到对应文件的。  
试一下：

### I. 定义命名空间和类文件

我们决定在/libs/vendors目录下定义一组以命名空间组织的类，其根目录命名为free-classes，这组类的全部在命名空间free\_classes下。注意这里我故意使根目录名与根命名空间名不一致以表示映射根目录不一定要和命名空间同名。  
创建文件/libs/vendors/free-classes/persons/Slave.php，没有目录请自行创建。

<?php

**namespace** **free\_classes**\**persons**;

**class** **Slave**

{

**public** **static** **function** **isFree**()

{

var\_dump("I'm FREE now, thank you!");

}

}

创建文件/libs/vendors/free-classes/vehicles/cars/Porsche.php。

<?php

**namespace** **free\_classes**\**vehicles**\**cars**;

**class** **Porsche**

{

**public** **static** **function** **isFree**()

{

var\_dump('Are you kidding?!');

}

}

**注意：** free-classes以下的目录名和结构都要遵循PSR-4标准。

### II. 配置\yii\web\Application::$aliases

这里要说一下，如果我们的命名空间为namespace\subnamespace，那么我们应该设置的路径别名就是@namespace/subnamespace（详解参照 [PSR-4](http://www.php-fig.org/psr/psr-4) ）。  
打开配置文件/config/web.php，配置Application的aliaes属性：

<?php

Yii::$classMap['Freedom'] = '@app/libs/Freedom.php';

...

$config = [

'id' => 'basic',

...

'aliases' => [

'@free\_classes' => '@app/libs/vendors/free-classes'

],

...

];

**return** $config;

### III. 使用命名空间下的类

又要见证奇迹了，还是选在SiteController::actionIndex()里。

<?php

...

**use** **free\_classes**\**persons**\**Slave**; // 还是别忘了导入

**use** **free\_classes**\**vehicles**\**cars**\**Porsche**;

**class** **SiteController** **extends** **Controller**

...

**public** **function** **actionIndex**()

{

// Freedom::yell();

Slave::isFree();

Porsche::isFree();

}

}

OK!

composer.json实现类的自动加载

用这种方法就是以扩展的方式来加载自定义的东西。