php之控制反转容器（Ioc Container）

更新于 2017年12月07日 by 白狼 被浏览了 2096 次

上一节文末，我们抛出了一个问题：如何解决注入引起的更复杂的问题?

有些人可能还没明白这个问题是怎么回事。

我们来详细的描述一下:

首先我们假设A依赖B和C，B依赖D，C依赖E，D依赖F，现在我们来看看怎么实例化A?

```
$a = new A(new B(new D(new F)), new C(new E()));
```

是不是感觉要崩溃了，实例化A之前就得先知道它依赖哪些类。

从一开始我们为了解决依赖的问题，到现在依赖却成了最棘手的问题，怎么办?

有人提出：如果我们也有一个类似composer的管理容器，在实例化A的时候，能够解决各个类之间的依赖问题是不是就很ok了?

没错，就是这样。

php有一个IoC容器的概念，其作用，就是帮助我们解决这个问题。

IoC容器又是什么？所谓的IoC容器呢，并没有多么高大上，它其实就是一个类，为了解决依赖问题，IoC应该提供如下功能:

1. 存储定义的类
2. 实例化类

下面来看一个简版的实现

```
Class Container
{
    private $_definitions;

    public function set($class, $definition)
    {
        $this->_definitions[$class] = $definition;
    }

    public function get($class, $params = [])
    {
        $definition = $this->_definitions[$class];
        return call_user_func($definition, $params);
    }
}
```

Container类封装的很简单，set方法用于存储类，get用于实例化类。

还以上一节用户发送邮件为例

```
class EmailSenderBy163
{
    private $_name;
    public function __construct($name = '')
    {
        $this->_name = $name;
    }

    public function send()
    {
    }
}
```

我们看一下如何借用IoC容器操作

```
$container = new Container;
$container->set('EmailSenderBy163', function ($name = '') {
    return new EmailSenderBy163($name);
});

$emailSenderBy163 = $container->get('EmailSenderBy163', '163');

echo "<pre>";
print_r($emailSenderBy163);
```

结果

```
EmailSenderBy163 Object
(
    [_name:EmailSenderBy163:private] => 163
)
```

简单分析下上面代码的含义

1. 首先我们定义了一个容器类 Container
2. 定义了一个私有属性 $_definitions, 用于保存定义的类
3. set 方法，添加类及其定义到 $_definitions 属性中，第一个参数是类名，第二个参数这里是一个回调函数，用于创建类 $class
4. get 方法，传递类名，通过 属性 $_definitions 找到类的定义，并调用 call_user_func 函数调用该定义, 第二个参数 $params 是set时第二个参数即回调函数的参数

当然，这只是一个简版的实现

下面我们扩展一下，如果要实现有依赖其他对象的类，该如何实现呢?

我们把Container的get方法整理一下

```
Class Container
{
    private $_definitions;

    public function set($class, $definition)
    {
        $this->_definitions[$class] = $definition;
    }

    public function get($class, $params = [])
    {
        if (isset($this->_definitions[$class]) && is_callable($this
            $definition = $this->_definitions[$class];
            return call_user_func($definition, $this, $params);
        } else {
            throw new Exception("error");
        }
    }
}
```

注意同刚才的get方法的区别。相比于简版的实现，get方法在判断回调函数上趋于严谨。

User类定义如下

```
class User
{
    private $_emailSenderObject;

    public function __construct(EmailSenderBy163 $emailSenderObject
    {
        $this->_emailSenderObject = $emailSenderObject;
    }
    public function register()
    {
        // other code

        $this->_emailSenderObject->send();
    }
}
```

注意看User类，我们想在实例化User的时候，通过构造函数“注入” EmailSenderBy163 的对象，并赋值给User::$_emailSenderObject属性。

来看看此时我们应该如何用Container实例化 EmailSenderBy163 和 User 。

```
$container = new Container;

$container->set('EmailSenderBy163', function ($container, $name = '
    return new EmailSenderBy163($name);
});
$container->set('User', function ($container, $params = []) {
    return new User($container->get($params[0], $params[1]));
});

echo '<pre>';
print_r($container->get('EmailSenderBy163', ['163']));
print_r($container->get('User', ['EmailSenderBy163', '163']));
```

结果

```
EmailSenderBy163 Object
(
    [_name:EmailSenderBy163:private] => Array
        (
            [0] => 163
        )

)
User Object
(
    [_emailSenderObject:User:private] => EmailSenderBy163 Object
        (
            [_name:EmailSenderBy163:private] => 163
        )

)
```

似乎没什么问题，一个类依赖的其他对象我们也解决了。

但是如果我们要解决文中一开始抛出的问题，还是需要先把所有的类全部set进去，然后get的时候指定其所依赖的类，依然很麻烦。

上一节文末我们让大家课后看一下反射的知识，不知道你熟不熟悉？

我们简单的聊两句：反射，是php5增加的功能，通过反射，可以导出或提取出关于类、方法、属性、参数等的详细信息。

也就是说，我们可以利用反射相关的一系列API，来分析类所依赖的对象，并做自动实例化处理。

限于篇幅，反射相关的API我们就不多做介绍了，不熟悉的可以看一下手册，了解即可。

下面我们基于反射，让我们的Container更强悍一些。

```
Class Container
{
    public function get($class, $params = [])
    {
        return $this->build($class, $params);
    }

    public function build($class, $params)
    {
        $dependencies = [];

        $reflection = new ReflectionClass($class);
        $constructor = $reflection->getConstructor();
        if ($constructor !== null) {
            foreach ($constructor->getParameters() as $param) {
                $c = $param->getClass();
                if ($c !== null) {
                    $dependencies[] = $this->get($c->getName());
                }
            }
        }

        foreach ($params as $index => $param) {
            $dependencies[$index] = $param;
        }

        return $reflection->newInstanceArgs($dependencies);
    }
}
```

同样是User类，现在我们可以这样实例化了

```
$container = new Container;
$user = $container->get('User');
echo '<pre>';
print_r($user);
```

结果

```
User Object
(
    [_emailSenderObject:User:private] => EmailSenderBy163 Object
        (
            [_name:EmailSenderBy163:private] =>
        )

)
```

本节课我们就说这么多，有任何不懂的问题，下面留言哦。

评论区 共36条评论

说点什么吧...

发布

liao**n**@gmail.com

按照示例写的，运行会报下面的错误
Fatal error: Uncaught TypeError: Argument 1 passed to User::__construct() must be an instance of EmailSendBy163, null given in /Users/liaobinbin/opt/htdocs/php_class/Container.php:29

2018-06-29 19:13 回复

白狼

注意看User::__construct方法，我们指定的参数必须是EmailSenderBy163类的实例，而你缺传递了一个null，所以，问题找到了

2018-06-30 13:59 回复

liao**n**@gmail.com

$container = new Container;
$user = $container->get('User');
echo '<pre>';
print_r($user);
这一步做的时候并没有给User类初始化参数，按道理User的初始参数是用反射来生成的，如果还有手动传参数，那我就不明白反射存在的意义

2018-06-30 21:29 回复

白狼

重现不了你的问题，跟着上下文重新来一遍试试吧

2018-07-01 07:47 回复

56**848**@qq.com

什么情况下使用依赖注入呢？以前在编程中为了使用不同类的方法，需要就new一下，是不是都需要改为本节所讲的new container的办法？

2018-01-10 16:44 回复

你二不二二就在呢

用debug分析了下,,传参哪里写的确实不对..

2017-12-07 18:00 回复

白狼

已修改，感谢反馈。

2017-12-07 20:59 回复

你二不二二就在呢

仔细看还是有bug的...

2017-12-06 20:10 回复

你二不二二就在呢

比如递归传参的params

2017-12-06 20:12 回复

你二不二二就在呢

哎,乱了,,不知道改怎么看这个递归啦.

2017-12-06 20:19 回复

你二不二二就在呢

哎,参数这块好乱,,看不懂,,

2017-12-06 20:23 回复

破宝

build函数里的params 参数是用来传什么值的?

2017-12-02 00:10 回复

白狼

初始化传递的参数值。

2017-12-03 19:07 回复

破宝

$dependencies[] = $this->get($c->getName(), $params);这是不是一个递归呀?

2017-12-01 20:32 回复

白狼

如果有其他依赖的类，是递归

2017-12-03 19:06 回复

欢乐豆

狼哥，请教下，基于反射的Container里build方法中
foreach ($params as $index=>$param){
$dependencies[$index]=$param;
}
这个$params难道指的是array(类名=>该类构造函数中的参数值)，比如：$user = $container->get('User', ['EmailSenderBy163' => 'This is name']);为啥不是$user = $container->get('User', ['EmailSenderBy163', 'This is name']);

2017-11-12 22:38 回复

你二不二二就在呢

你这两个都是错的..还让你传EmailSenderBy163,那就不是反射啦.

2017-12-07 18:09 回复

欢乐豆

哦，谢谢了，看到狼哥把文中一处修改了， $dependencies[] = $this->get($c->getName());原来另外一个参数的$params去掉了，这个参数是不是可有可无的？如果要给这个参数赋值的话，是不是一个依赖类名的索引数组啊，遍历以后放到$dependencies这个数组中来?

2017-12-08 14:29 回复

欢乐豆

1.我看后面的提问中，有讲到，get()里面是可以传递参数的，比如EmailSenderBy163::_name 属性可以这样设置$user = $container->get('User', ['EmailSenderBy163' => 'This is name']);
2.传递EmailSenderBy163这个类是为了初始化某个依赖类对应的属性值，要不那么多依赖类，怎么找到你要初始化那个依赖类的属性呢？这个应该跟反射无关吧

2017-12-08 14:38 回复

幽兰逸隐潇尘渊

在那個user類的構造方法的時候爲什麽我用郵件接口interface會報錯，不用類型限制也會報錯

2017-11-05 23:00 回复

白狼

报错信息贴一下

2017-11-06 09:39 回复

幽兰逸隐潇尘渊

其實看了反射之後回來再看你就覺得其實很簡單了

2017-11-05 22:13 回复

nfgn**@163.com

狼哥，照最后这个build()函数的意思，那是不是说通过构造函数的形式实现依赖注入更为通用？

2017-11-01 22:10 回复

1 2 (/comment/more?id=59&type=2&page=2) 下一页 (/comment/more?id=59&type=2&page=2)

友情链接

Bootstrap中文网 (http://www.bootcss.com)

开放CDN服务 (http://www.bootcdn.cn)

Yii2.0官方文档 (http://www.yiiframework.com)

阿里云 (http://www.aliyun.com/) 码云 (http://git.oschina.net/)

Yii中文社区 (http://www.yiichina.com/)

一点PHP (https://www.yidianphp.com) 站点地图 (/sitemap.xml)

联系我

email：bailangzhan@qq.com

QQ群：147533993（已满），群2：659817663

服务商