

[首页](#) [资讯](#) [精华](#) [论坛](#) [问答](#) [博客](#) [专栏](#) [群组](#) [更多 ▼](#)
[您还未登录！](#) [登录](#) [注册](#)

stamen的程序员之路

- [博客](#)
- [微博](#)
- [相册](#)
- [收藏](#)
- [留言](#)
- [关于我](#)

透透彻彻IoC（你没有理由不懂！）

博客分类：

- [04 SSH](#)
- [10 Spring 3.x企业应用开发实战](#)



spring企业应用

引述：IoC（控制反转：Inverse of Control）是Spring容器的内核，AOP、声明式事务等功能在此基础上开花结果。但是IoC这个重要的概念却比较晦涩隐讳，不容易让人望文生义，这不能不说是一大遗憾。不过IoC确实包括很多内涵，它涉及代码解耦、设计模式、代码优化等问题的考量，我们打算通过一个小例子来说明这个概念。

通过实例理解IoC的概念

贺岁大片在中国已经形成了一个传统，每到年底总有多部贺岁大片纷至沓来让人应接不暇。在所有贺岁大片中，张之亮的《墨攻》算是比较出彩的一部。该片讲述了战国时期墨家人革离帮助梁国反抗赵国侵略的个人英雄主义故事，恢宏壮阔、浑雄凝重的历史场面相当震撼。其中有一个场景：当刘德华所饰演的墨者革离到达梁国都城下，城上梁国守军问到：“来者何人？”刘德华回答：“墨者革离！”我们不妨通过一个Java类为这个“城门叩问”的场景进行编剧，并借此理解IoC的概念：

代码清单3-1 MoAttack：通过演员安排剧本

Java代码  

```
1. public class MoAttack {
2.     public void cityGateAsk() {
3.         //①演员直接侵入剧本
4.         LiuDeHua ldh = new LiuDeHua();
5.         ldh.responseAsk("墨者革离！");
6.     }
7. }
```



我们会发现以上剧本在①处，作为具体角色饰演者的刘德华直接侵入到剧本中，使剧本和演员直接耦合在一起（图3-1）。



图 3-1 剧本和演员直接耦合

一个明智的编剧在剧情创作时应围绕故事的角色进行，而不应考虑角色的具体饰演者，这样才可能在剧本投拍时自由地遴选任何适合的演员，而非绑定在刘德华一人身上。通过以上的分析，我们知道需要为该剧本主人公革离定义一个接口：

代码清单3-2 MoAttack：引入剧本角色

Java代码  

```
1. public class MoAttack {
2.     public void cityGateAsk()
3.     {
4.         //①引入革离角色接口
5.         GeLi geli = new LiuDeHua();
6.
7.         //②通过接口开展剧情
8.         geli.responseAsk("墨者革离！");
9.     }
10. }
```

在①处引入了剧本的角色——革离，剧本的情节通过角色展开，在拍摄时角色由演员饰演，如②处所示。因此墨攻、革离、刘德华三者的类图关系如图 3 2所示：

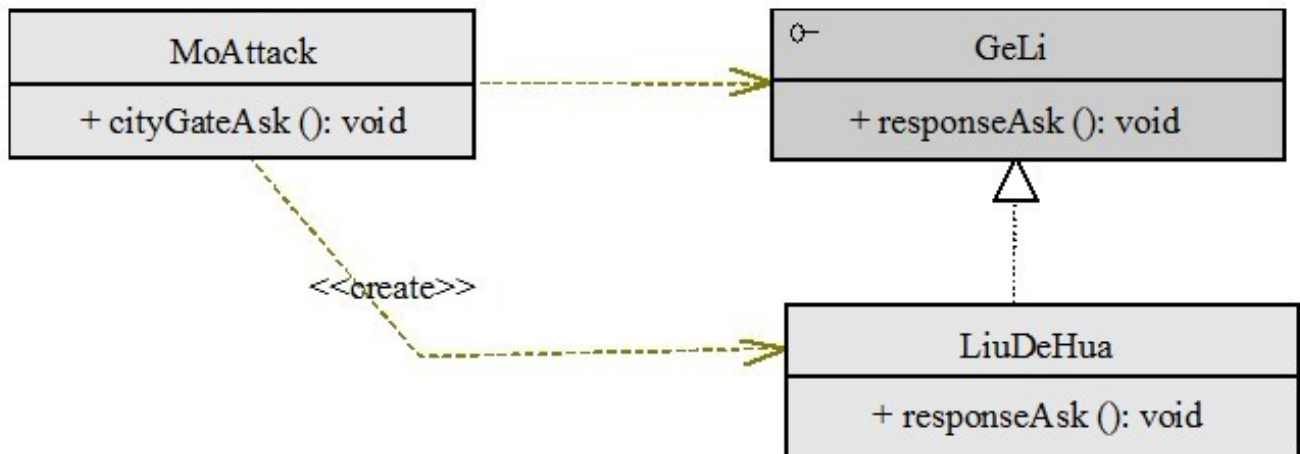


图 3-2 引入角色接口后的关系

可是，从图3 2中，我们可以看出MoAttack同时依赖于GeLi接口和LiuDeHua类，并没有达到我们所期望的剧本仅依赖于角色的目的。但是角色最终必须通过具体的演员才能完成拍摄，如何让LiuDeHua和剧本无关而又能完成GeLi的具体动作呢？当然是在影片投拍时，导演将LiuDeHua安排在GeLi的角色上，导演将剧本、角色、饰演者装配起来（图3-3）。

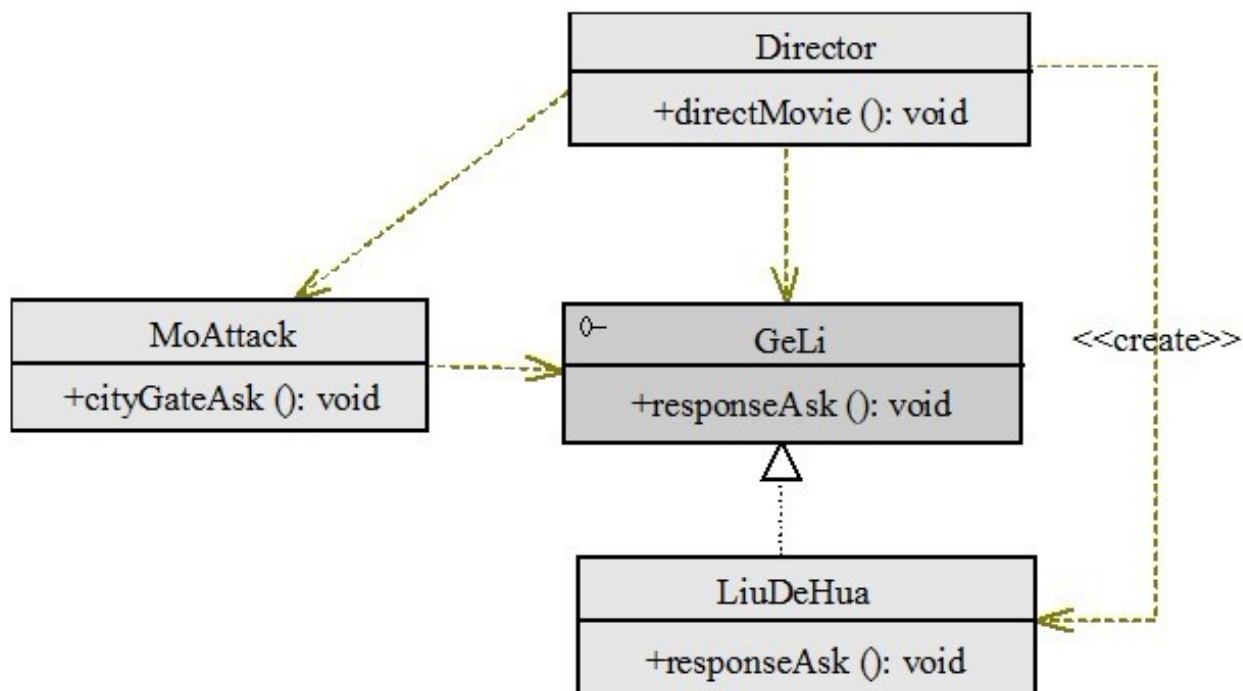


图 3-3 剧本和饰演者解耦了

通过引入导演，使剧本和具体饰演者解耦了。对应到软件中，导演像是一个装配器，安排演员表演具体的角色。

现在我们可以反过来讲解IoC的概念了。IoC（Inverse of Control）的字面意思是控制反转，它包括两个内容：

- 其一是控制
- 其二是反转

那到底是什么东西的“控制”被“反转”了呢？对应到前面的例子，“控制”是指选择GeLi角色扮演者的控制权；“反转”是指这种控制权从《墨攻》剧本中移除，转交到导演的手中。对于软件来说，即是某一接口具体实现类的选择控制权从调用类中移除，转交给第三方决定。

因为IoC确实不够开门见山，因此业界曾进行了广泛的讨论，最终软件界的泰斗级人物Martin Fowler提出了DI（依赖注入：Dependency Injection）的概念用以代替IoC，即让调用类对某一接口实现类的依赖关系由第三方（容器或协作类）注入，以移除调用类对某一接口实现类的依赖。“依赖注入”这个名词显然比“控制反转”直接明了、易于理解。



IoC的类型

从注入方法上看，主要可以划分为三种类型：构造函数注入、属性注入和接口注入。Spring支持构造函数注入和属性注入。下面我们继续使用以上的例子说明这三种注入方法的区别。

构造函数注入

在构造函数注入中，我们通过调用类的构造函数，将接口实现类通过构造函数变量传入，如代码清单3-3所示：

代码清单3-3 MoAttack：通过构造函数注入革离扮演者



Java代码  

```
1. public class MoAttack {
2.     private GeLi geli;
3.     //①注入革离的具体扮演者
4.     public MoAttack(GeLi geli) {
```

```
5.         this.geli = geli;
6.     }
7.     public void cityGateAsk() {
8.         geli.responseAsk("墨者革离！");
9.     }
10. }
```

MoAttack的构造函数不关心具体是谁扮演革离这个角色，只要在①处传入的扮演者按剧本要求完成相应的表演即可。角色的具体扮演者由导演来安排，如代码清单3-4所示：

代码清单3-4 Director：通过构造函数注入革离扮演者

Java代码  



```
1. public class Director {
2.     public void direct() {
3.         //①指定角色的扮演者
4.         GeLi geli = new LiuDeHua();
5.
6.         //②注入具体扮演者到剧本中
7.         MoAttack moAttack = new MoAttack(geli);
8.         moAttack.cityGateAsk();
9.     }
10. }
```

在①处，导演安排刘德华饰演革离的角色，并在②处，将刘德华“注入”到墨攻的剧本中，然后开始“城门叩问”剧情的演出工作。

属性注入

有时，导演会发现，虽然革离是影片《墨攻》的第一主角，但并非每个场景都需要革离的出现，在这种情况下通过构造函数注入相当于每时每刻都在革离的饰演者在场，可见并不妥当，这时可以考虑使用属性注入。属性注入可以有选择地通过Setter方法完成调用类所需依赖的注入，更加灵活方便：



代码清单3-5 MoAttack：通过Setter方法注入革离扮演者

Java代码  

```
1. public class MoAttack {
2.     private GeLi geli;
3.     //①属性注入方法
4.     public void setGeli(GeLi geli) {
5.         this.geli = geli;
6.     }
7.     public void cityGateAsk() {
8.         geli.responseAsk("墨者革离");
9.     }
10. }
```

MoAttack在①处为geli属性提供一个Setter方法，以便让导演在需要时注入geli的具体扮演者。

代码清单3-6 Director：通过Setter方法注入革离扮演者

Java代码  



```
1. public class Director {
2.     public void direct() {
3.         GeLi geli = new LiuDeHua();
```

```
4.         MoAttack moAttack = new MoAttack();
5.
6.         //①调用属性Setter方法注入
7.         moAttack.setGeli(geli);
8.         moAttack.cityGateAsk();
9.     }
10. }
```

和通过构造函数注入革离扮演者不同，在实例化MoAttack剧本时，并未指定任何扮演者，而是在实例化MoAttack后，在需要革离出场时，才调用其setGeli()方法注入扮演者。按照类似的方式，我们还可以分别为剧本中其他诸如梁王、巷淹中等角色提供注入的Setter方法，这样，导演就可以根据所拍剧段的不同，注入相应的角色了。

接口注入



将调用类所有依赖注入的方法抽取到一个接口中，调用类通过实现该接口提供相应的注入方法。为了采取接口注入的方式，必须先声明一个ActorArrangable接口：

Java代码  

```
1. public interface ActorArrangable {
2.     void injectGeli(GeLi geli);
3. }
```

然后，MoAttack实现ActorArrangable接口提供具体的实现：



代码清单3-7 MoAttack：通过接口方法注入革离扮演者

Java代码  

```
1. public class MoAttack implements ActorArrangable {
2.     private GeLi geli;
3.     //①实现接口方法
4.     public void injectGeli (GeLi geli) {
5.         this.geli = geli;
6.     }
7.     public void cityGateAsk() {
8.         geli.responseAsk("墨者革离");
9.     }
10. }
```

Director通过ActorArrangable的injectGeli()方法完成扮演者的注入工作。

代码清单3-8 Director：通过接口方法注入革离扮演者

Java代码  



```
1. public class Director {
2.     public void direct() {
3.         GeLi geli = new LiuDeHua();
4.         MoAttack moAttack = new MoAttack();
5.         moAttack.injectGeli (geli);
6.         moAttack.cityGateAsk();
7.     }
8. }
```

由于通过接口注入需要额外声明一个接口，增加了类的数目，而且它的效果和属性注入并无本质区别，因此我们不提倡采用这种方式。

通过容器完成依赖关系的注入

虽然MoAttack和LiuDeHua实现了解耦，MoAttack无须关注角色实现类的实例化工作，但这些工作在代码中依然存在，只是转移到Director类中而已。假设某一制片人想改变这一局面，在选择某个剧本后，希望通过一个“海选”或者第三中介机构来选择导演、演员，让他们各司其职，那剧本、导演、演员就都实现解耦了。

所谓媒体“海选”和第三方中介机构在程序领域即是一个第三方的容器，它帮助完成类的初始化与装配工作，让开发者从这些底层实现类的实例化、依赖关系装配等工作中脱离出来，专注于更有意义的业务逻辑开发工作。这无疑是一件令人向往的事情，Spring就是这样的一个容器，它通过配置文件或注解描述类和类之间的依赖关系，自动完成类的初始化和依赖注入的工作。下面是Spring配置文件的对以上实例进行配置的配置文件的片断：

Xml代码  



```
1. <?xml version="1.0" encoding="UTF-8" ?>
2. <beans xmlns="http://www.springframework.org/schema/beans"
3.       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4.       xmlns:p="http://www.springframework.org/schema/p"
5.       xsi:schemaLocation="http://www.springframework.org/schema/beans
6.       http://www.springframework.org/schema/beans/spring-beans-
7.       3.0.xsd">
8.       <!--①实现类实例化-->
9.       <bean id="geli" class="LiuDeHua"/>
10.      <bean id="moAttack" class="com.baobaotao.ioc.MoAttack"
11.          p:geli-ref="geli"/><!--②通过geli-ref建立依赖关系-->
12. </beans>
```

通过new XmlBeanFactory(“beans.xml”)等方式即可启动容器。在容器启动时，Spring根据配置文件的描述信息，自动实例化Bean并完成依赖关系的装配，从容器中即可返回准备就绪的Bean实例，后续可直接使用之。

Spring为什么会有这种“神奇”的力量，仅凭一个简单的配置文件，就能魔法般地实例化并装配好程序所用的Bean呢？这种“神奇”的力量归功于Java语言本身的类反射功能。

这些文章摘自于我的《Spring 3.x企业应用开发实战》的第3章，我将通过连载的方式，陆续在此发出。欢迎大家讨论。

- [查看图片附件](#)

分享到:  

[学习Spring必学的Java基础知识\(1\)----反射](#) | [单元测试系列之5: 使用unitils测试Servic ...](#)

- 2012-04-18 11:01
- 浏览 30045
- [评论\(14\)](#)
- 论坛回复 / [浏览](#) (54 / 22126)
- 分类: [开源软件](#)
- [相关推荐](#)

参考知识库



[HTML5知识库](#) 479 关注 | 145 收录



[OpenCV知识库](#) 261 关注 | 835 收录



[软件测试知识库](#) 329 关注 | 244 收录



[C++知识库](#) 1057 关注 | 697 收录

评论

14 楼 [mvweimm](#) 2016-05-23

楼主说的相当的好啊，例子相当的不错，最喜欢看这种把理论，用很通俗的场景或白话讲解出来的文章，这也能说明作者的对理论的理解相当的深入😎

13 楼 [xinzhong](#) 2016-02-22

不明觉厉。但还是进一步加深了理解，可惜我现在不用spring。谢谢分享！

12 楼 [fran_maomao](#) 2015-09-08

感谢楼主分享 受益匪浅

11 楼 [旋风魔帝](#) 2015-08-10



10 楼 [飞鸟830413](#) 2014-05-26

精通Spring2. x-企业应用开发详解第三章内容。预知下回分解，请买本书看吧。不过楼主应该注明出处。

9 楼 [mengfei1001](#) 2013-04-03



8 楼 [a20071426](#) 2012-09-24

good!

7 楼 [xiaojil23pt](#) 2012-07-26



6 楼 [zlf3865072](#) 2012-07-05

23432

5 楼 [shanliangdesishen9](#) 2012-06-13

谢谢分享😊 感觉受益匪浅

4 楼 [gouerli](#) 2012-06-08

写的很不错，对于ioc的原理介绍的很清晰，比spring官方文档中好理解多了，对于接口注入的方式我倒是第一次听说，谢谢博主的分享！

3 楼 [wh2006xtt](#) 2012-05-16

用到Spring的依赖注入的项目都完成了，对其原理还是晕的，读了此文明白些了。

0(∩_∩)0谢谢分享

2 楼 [xautjzd](#) 2012-04-28

最近初学spring，看IoC看了半天没看懂，晕乎乎的，对其流程依然不解，但是看了此文章后便清楚多了，虽然还是没完全消化，但是起码有了一个大致的了解，感谢您的无私奉献！

1 楼 [fount](#) 2012-04-28

spring使用有一段时间了，看看这系列的文章真是受益匪浅啊，谢谢分享。

发表评论



[您还没有登录, 请您登录后再发表评论](#)



stamen

- 浏览: 685243 次
- 性别:
- 来自: 厦门
- 我现在离线

最近访客

[更多访客>>](#)

ITEYE

[JoeLeeWU](#)

ITEYE

[yanggaol23](#)

ITEYE

[withcourageto](#)

ITEYE

[xiaoniaol23](#)

博客专栏

Spring 3.x



[Spring 3.x企业实...](#)

浏览量: 221047

文章分类

- [全部博客 \(128\)](#)
- [01 云计算 \(3\)](#)
- [02 企业应用 \(24\)](#)
- [03 WebService \(2\)](#)
- [04 SSH \(38\)](#)
- [05 程序测试 \(12\)](#)
- [06 软件人文 \(1\)](#)
- [07 Linux \(5\)](#)
- [08 Java基础 \(14\)](#)
- [10 Spring 3.x企业应用开发实战 \(26\)](#)
- [99 杂项 \(13\)](#)
- [09 构建及版本 \(9\)](#)

- [90 休闲 \(0\)](#)
- [20 行业资讯 \(0\)](#)
- [11 Rop \(3\)](#)
- [开源代码 \(7\)](#)
- [系统运维 \(3\)](#)
- [11 Web前端 \(2\)](#)
- [aa 短博 \(1\)](#)

社区版块

- [我的资讯 \(0\)](#)
- [我的论坛 \(448\)](#)
- [我的问答 \(0\)](#)

存档分类

- [2016-04 \(1\)](#)
- [2015-12 \(1\)](#)
- [2015-07 \(3\)](#)
- [更多存档...](#)

评论排行榜

- [如何通过项目配置文件指定Log4J的配置文件](#)
- [打造易于部署的WEB应用项目](#)
- [一个批处理文件N个知识点](#)

最新评论

- [dengbin50](#): 从无到有，层层深入。注重基础和原理，很不错！
[学习Spring必学的Java基础知识\(8\)——国际化信息](#)
- [liuxiang_eyes](#): 现在还真是碰到Hibernate+JPA的情况，这种情况您有研 ...
[Spring的事务管理难点剖析\(5\):联合军种作战的混乱](#)
- [myweimm](#): 楼主说的相当的好啊，例子相当的不错，最喜欢看这种把理论，用很通俗 ...
[透透彻彻IoC（你没有理由不懂！）](#)
- [gongrunlian](#): <div class="quote_title ...
[Spring的事务管理难点剖析\(4\):多线程的困惑](#)
- [飞天奔月](#): 没代码显示?
[一个批处理文件N个知识点](#)

声明：ITeye文章版权属于作者，受法律保护。没有作者书面许可不得转载。若作者同意转载，必须以超链接形式标明文章原始出处和作者。

© 2003-2016 ITeye.com. All rights reserved. [京ICP证110151号 京公网安备110105010620]