2016/7/10 控制反转 百度百科

新闻 网页贴吧 知道 音乐图片 视频 地图 百科文库

搜索词条 进入词条

帮助

• 政府间海洋学委员会简称

首页 分类 特色百科 用户 权威合作 **丰**机,百科 3人个 &

ioc是一个多义词,请在下列义项上选择浏览(共5个义项)

ioc

添力

• IOC

• 初始作战能力

- 国际奥林匹克委员会简称
- IBM智慧城市智能运行中心(IOC)

收藏 708 25

## 控制反转 🖍 編

同义词 ioc (IOC) 一般指控制反转

控制反转(Inversion of Control,英文缩写为IoC)是一个重要的面向对象编程的法则来削减计算机程序的耦合问题,也是轻 量级的Spring框架的核心。 控制反转一般分为两种类型,依赖注入(Dependency Injection,简称DI)和依赖查找 (Dependency Lookup)。依赖注入应用比较广泛。

中文名 控制反转 起源时间

2004年

7 实现方式

外文名 Inverse of Control 目 的

削减计算机程序的耦合问题

目录

- 1 起源
- 4 实现初探
- 5 类型
- 3 优缺点

- 2 设计模式
- 6 实现策略

起源

★ 編辑

早在2004年,Martin Fowler就提出了"哪些方面的控制被反转了?"这个问题。他总结出是依赖对象的获得被反转了。基于这 个结论,他为控制反转创造了一个更好的名字: 依赖注入。许多非凡的应用(比HelloWorld.java更加优美,更加复杂)都是由两 个或是更多的类通过彼此的合作来实现业务逻辑,这使得每个对象都需要,与其合作的对象(也就是它所依赖的对象)的引用。 如果这个获取过程要靠自身实现,那么如你所见,这将导致代码高度耦合并且难以测试。

IoC 亦称为 "依赖倒置原理"("Dependency Inversion Principle")。差不多所有框架都使用了"倒置注入(Fowler 2004)技巧, 这 可说是IoC原理的一项应用。SmallTalk,C++, Java 或.NET 等各种面向对象程序语言的程序员已使用了这些原理。

控制反转是Spring框架的核心。

应用控制反转、对象在被创建的时候、由一个调控系统内所有对象的外界实体将其所依赖的对象的引用传递给它。也可以 说,依赖被注入到对象中。所以,控制反转是,关于一个对象如何获取他所依赖的对象的引用,这个责任的反转。

设计模式 **№** 编辑

IoC可以认为是一种全新的设计模式,但是理论和时间成熟相对较晚,并没有包含在GoF中。

Interface Driven Design接口驱动,接口驱动有很多好处,可以提供不同灵活的子类实现,增加代码稳定和健壮性等等,但 是接口一定是需要实现的,也就是如下语句迟早要执行: AInterface a = new AInterfaceImp(); 这样一来,耦合关系就产生了, 如:

```
classA
         AInterface a;
         A(){}
         AMethod()//一个方法
             a = new AInterfaceImp();
10
```

Class A与AInterfaceImp就是依赖关系,如果想使用AInterface的另外一个实现就需要更改代码了。当然我们可以建立一个 Factory来根据条件生成想要的AInterface的具体实现,即:

```
InterfaceImplFactory
   AInterface create(Object condition)
      if(condition = condA)
          return new AInterfaceImpA():
```



直理之迷 🗸 🚃 商城 💟 消息 |



## 词条统计

浏览次数: 304930次 编辑次数: 55次历史版本 最近更新: 2015-12-22 创建者: weiyao\_85

表面上是在一定程度上缓解了以上问题,但实质上这种代码耦合并没有改变。通过IoC模式可以彻底解决这种耦合,它把耦合从代码中移出去,放到统一的XML 文件中,通过一个容器在需要的时候把这个依赖关系形成,即把需要的接口实现注入到需要它的类中,这可能就是"依赖注入"说法的来源了。

loC模式,系统中通过引入实现了loC模式的loC容器,即可由loC容器来管理对象的生命周期、依赖关系等,从而使得应用程序的配置和依赖性规范与实际的应用程序代码分开。其中一个特点就是通过文本的配置文件进行应用程序组件间相互关系的配置,而不用重新修改并编译具体的代码。

当前比较知名的IoC容器有: Pico Container、Avalon、Spring、JBoss、HiveMind、EJB等。

在上面的几个IoC容器中,轻量级的有Pico Container、Avalon、Spring、HiveMind等,超重量级的有EJB,而半轻半重的有容器有JBoss。Jdon等。

可以把IoC模式看做是工厂模式的升华,可以把IoC看作是一个大工厂,只不过这个大工厂里要生成的对象都是在XML文件中给出定义的,然后利用Java 的"反射"编程,根据XML中给出的类名生成相应的对象。从实现来看,IoC是把以前在工厂方法里写死的对象生成代码,改变为由XML文件来定义,也就是把工厂和对象生成这两者独立分隔开来,目的就是提高灵活性和可维护性

loC中最基本的Java技术就是"反射"编程。反射又是一个生涩的名词,通俗的说反射就是根据给出的类名(字符串)来生成对象。这种编程方式可以让对象在生成时才决定要生成哪一种对象。反射的应用是很广泛的,像Hibernate、Spring中都是用"反射"做为最基本的技术手段。

在过去,反射编程方式相对于正常的对象生成方式要慢10几倍,这也许也是当时为什么反射技术没有普遍应用开来的原因。但经SUN改良优化后,反射方式生成对象和通常对象生成方式,速度已经相差不大了(但依然有一倍以上的差距)。

**优缺点** 

loC最大的好处是什么?因为把对象生成放在了XML里定义,所以当我们需要换一个实现子类将会变成很简单(一般这样的对象都是实现于某种接口的),只要修改XML就可以了,这样我们甚至可以实现对象的热插拔(有点像USB接口和SCSI硬盘了)。



ос

Q

点损耗是微不足道的,除非某对象的生成对效率要求特别高。(3)缺少IDE重构操作的支持,如果在Eclipse要对类改名,那么你还需要去XML文件里手工去改了,这似乎是所有XML方式的缺憾所在。

IOC关注服务(或应用程序部件)是如何定义的以及他们应该如何定位他们依赖的其它服务。通常,通过一个容器或定位框架来获得定义和定位的分离,容器或定位框架负责:

保存可用服务的集合

提供一种方式将各种部件与它们依赖的服务绑定在一起

为应用程序代码提供一种方式来请求已配置的对象(例如,一个所有依赖都满足的对象),这种方式可以确保该对象需要的所有相关的服务都可用。

现有的框架实际上使用以下三种基本技术的框架执行服务和部件间的绑定:

类型1(基于接口):可服务的对象需要实现一个专门的接口,该接口提供了一个对象,可以重用这个对象查找依赖(其它服务)。早期的容器Excalibur使用这种模式。

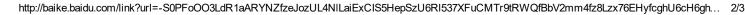
类型2 (基于setter): 通过JavaBean的属性(setter方法)为可服务对象指定服务。HiveMind和Spring采用这种方式。

类型3 (基于构造函数): 通过构造函数的参数为可服务对象指定服务。PicoContainer只使用这种方式。HiveMind和Spring也使用这种方式。

实现策略

✔ 编辑

loC是一个很大的概念,可以用不同的方式实现。其主要形式有两种:



分享

◇依赖查找:容器提供回调接口和上下文条件给组件。EJB和Apache Avalon 都使用这种方式。这样一来,组件就必须使用 容器提供的API来查找资源和协作对象,仅有的控制反转只体现在那些回调方法上(也就是上面所说的类型1):容器将调用这些 回调方法,从而让应用代码获得相关资源。

◇依赖注入:组件不做定位查询,只提供普通的Java方法让容器去决定依赖关系。容器全权负责的组件的装配,它会把符合 依赖关系的对象通过JavaBean属性或者构造函数传递给需要的对象。通过JavaBean属性注射依赖关系的做法称为设值方法注入 (Setter Injection);将依赖关系作为构造函数参数传入的做法称为构造器注入(Constructor Injection)

♪ 编辑 实现方式

实现数据访问层

数据访问层有两个目标。第一是将数据库引擎从应用中抽象出来,这样就可以随时改变数据库—比方说,从微软SQL变成 Oracle。不过在实践上很少会这么做,也没有足够的理由未来使用实现数据访问层而进行重构现有应用的努力。 [1]

第二个目标是将数据模型从数据库实现中抽象出来。这使得数据库或代码开源根据需要改变,同时只会影响主应用的一小部 分——数据访问层。这一目标是值得的,为了在现有系统中实现它进行必要的重构。

模块与接口重构

依赖注入背后的一个核心思想是单一功能原则(single responsibility principle)。该原则指出,每一个对象应该有一个特定 的目的, 而应用需要利用这一目的的不同部分应当使用合适的对象。这意味着这些对象在系统的任何地方都可以重用。但在现有 系统里面很多时候都不是这样的。[1]

随时增加单元测试

把功能封装到整个对象里面会导致自动测试困难或者不可能。将模块和接口与特定对象隔离,以这种方式重构可以执行更先 进的单元测试。按照后面再增加测试的想法继续重构模块是诱惑力的,但这是错误的。[1]

使用服务定位器而不是构造注入

实现控制反转不止一种方法。最常见的办法是使用构造注入,这需要在对象首次被创建是提供所有的软件依赖。然而,构造 注入要假设整个系统都使用这一模式,这意味着整个系统必须同时进行重构。这很困难、有风险,且耗时。[1]

- 参考资料
- 1. 如何实现对现有应用的依赖注入 . TechTarget[引用日期2015-07-29]

词条标签: 计算机学, 互联网

- 1 起源
- 2 设计模式
- 3 优缺点
- 4 实现初探
- 5 类型
- 6 实现策略
- 7 实现方式

新手上路

成长任务 编辑入门 编辑规则 百科术语

♥ 我有疑问

我要质疑 我要提问 参加讨论 意见反馈 ₽ 投诉建议

举报不良信息 未通过词条申诉 投诉侵权信息 封禁杳询与解封

©2016Baidu 使用百度前必读 | 百科协议 | 百度百科合作平台