关于spring框架中的ioc的幽默解释 PDF转换可能丢失图片或格式，建议阅读原文 https://www.100test.com/kao_ti2020/182/2021_2022__E5_85_B3_E4_BA_8Espri_c104_182513.htm oC就是Inversion of Control，控制反转。在Java开发中，IoC意味着将你设计好的类交给系统去控制，而不是在你的类内部控制。这称为控制反转。下面我们以几个例子来说明什么是IoC 假设我们要设计一个Girl和一个Boy类，其中Girl有kiss方法，即Girl想要Kiss一个Boy。那么，我们的问题是，Girl如何能够认识这个Boy？在我们中国，常见的MM与GG的认识方式有以下几种 1 青梅竹马；2 亲友介绍；3 父母包办那么哪一种才是最好呢？青梅竹马：Girl从小就知道自己的Boy。public class Girl {void kiss(){Boy boy = new Boy().}} 然而从开始就创建的Boy缺点就是无法在更换。并且要负责Boy的整个生命周期。如果我们的Girl想要换一个怎么办？（笔者严重不支持Girl经常更换Boy）亲友介绍：由中间人负责提供Boy来见面public class Girl {void kiss(){Boy boy = BoyFactory.createBoy().}} 亲友介绍，固然是好。如果不满意，尽管另外换一个好了。但是，亲友BoyFactory经常是以Singleton的形式出现，不然就是，存在于 Globals，无处不在，无处不能。实在是太繁琐了一点，不够灵活。我为什么一定要这个亲友掺和进来呢？为什么一定要付给她介绍费呢？万一最好的朋友爱上了我的男朋友呢？父母包办：一切交给父母，自己不用费吹灰之力，只需要等着Kiss就好了。/Java/Files/2007-3/27/1036382661.gif>.public class Girl {void kiss(Boy boy){// kiss boyboy.kiss().}} Well，这是对Girl最好的方法，只要想办法贿赂了Girl的父母，并把Boy交给他。那么我

们就可以轻松的和Girl来Kiss了。看来几千年传统的父母之命还真是有用哦。至少Boy和Girl不用自己瞎忙乎了。这就是IOC，将对象的创建和获取提取到外部。由外部容器提供需要的组件。我们知道好莱坞原则：“ Do not call us, we will call you. ”意思就是， You, girlie, do not call the boy. We will feed you a boy。我们还应该知道依赖倒转原则即 Dependence Inversion Princinple， DIP。Eric Gamma说，要面向抽象编程。面向接口编程是面向对象的核心。组件应该分为两部分，即Service, 所提供功能的声明Implementation, Service的实现好处是：多实现可以任意切换，防止“ everything depends on everything ”问题．即具体依赖于具体。 所以，我们的Boy应该是实现Kissable接口。这样一旦Girl不想kiss可恶的Boy的话，还可以kiss可爱的kitten和慈祥的grandmother
。/Java/Files/2007-3/27/1036399644.gif>.二、IOC的typeIoC的Type指的是Girl得到Boy的几种不同方式。我们逐一来说明。IOC type 0：不用IOCpublic class Girl implements Servicable {private Kissable kissable.public Girl() {kissable = new Boy().}public void kissYourKissable() {kissable.kiss().}}Girl自己建立自己的Boy，很难更换，很难共享给别人，只能单独使用，并负责完全的生命周期。IOC type 1，先看代码：public class Girl implements Servicable {Kissable kissable.public void service(ServiceManager mgr) {kissable = (Kissable) mgr.lookup(“ kissable ”).}public void kissYourKissable() {kissable.kiss().}} 这种情况出现于Avalon Framework。一个组件实现了Servicable接口，就必须实现service方法，并传入一个ServiceManager。其中会含有需要的其它组件。只需要在service方法中初始化需要

的Boy。另外，J2EE中从Context取得对象也属于type 1。它依赖于配置文件 … IOC type 2：public class Girl {private Kissable kissable.public void setKissable(Kissable kissable) {this.kissable = kissable.}public void kissYourKissable() {kissable.kiss().}} Type 2出现于Spring Framework，是通过JavaBean的set方法来将需要的Boy传递给Girl。它必须依赖于配置文件。IOC type 3public class Girl {private Kissable kissable.public Girl(Kissable kissable) {this.kissable = kissable.}public void kissYourKissable() {kissable.kiss().}}这就是PicoContainer的组件 。通过构造函数传递Boy给Girl。PicoContainer container = new DefaultPicoContainer().container.registerComponentImplementation(Boy.class).container.registerComponentImplementation(Girl.class).Girl girl = (Girl) container.getComponentInstance(Girl.class).girl.kissYourKissable().