😊Spring ioc

微信公众号java-mindmap

ioc概念

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。

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。

深入分析

1. 谁依赖于谁 — 当然是应用程序依赖于IoC容器
2. 为什么需要依赖 — 应用程序需要IoC容器来提供对象需要的外部资源
3. 谁注入谁 — 很明显是IoC容器注入应用程序某个对象，应用程序依赖的对象
4. 注入了什么 — 就是注入某个对象所需要的外部资源（包括对象、资源、常量数据）

与new对象的区别

正转与反转

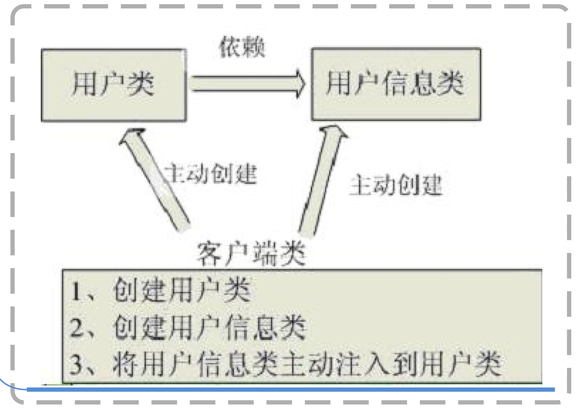
- 传统应用程序是由我们自己在对象中主动控制去直接获取依赖对象，也就是正转
- 而反转则是由容器来帮忙创建及注入依赖对象

ioc优缺点

- 优点：实现组件之间的解耦，提高程序的灵活性和可维护性
- 缺点：对象 生成因为是使用反射编程，在效率上有些损耗。但相对于IoC提高的维护性和灵活性来说，这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

图例说明

传统new方式



使用ioc后