BeanFactory与ApplacationContext

微信公众号java-mindmap

BeanFacotry

BeanFactory提供了一种先进的配置机制来管理任何种类bean(对象)，这种配置机制考虑到任何一种可能的存储方式

ApplicationContext

建立在BeanFactory，具有BeanFactory的所有功能和行为

- MessageSource, 提供国际化的消息访问 — 扩展了MessageResource接口
- 资源访问，如URL和文件
- 事件传播
- 载入多个（有继承关系）上下文 ，使得每一个上下文都专注于一个特定的层次，比如应用的web层

不同点

选用哪个

BeanFactory是延迟加载,如果Bean的某一个属性没有注入，BeanFacotry加载后，直至第一次使用调用getBean方法才会抛出异常

ApplicationContext则在初始化自身是检验，这样有利于检查所依赖属性是否注入；所以通常情况下我们选择使用ApplicationContext

特性表

特性	BeanFactory	ApplicationContext
Bean 实例化/装配	Yes	Yes
自动 BeanPostProcessor 注册	No	Yes
自动 BeanFactoryPostProcessor 注册	No	Yes
便捷的 MessageSource 访问(i18n)	No	Yes
ApplicationEvent 发送	No	Yes

关系

BeanFactory提供了配置框架和基本的功能

ApplicationContext建立在BeanFactory之上，并增加了其他的功能

一般来说，ApplicationContext是BeanFactory的完全超集， 任何BeanFactory功能和行为的描述也同样被认为适用于ApplicationContext

😊ioc容器

微信公众号java-mindmap

- 概念
  - 在每个框架的中都有一个容器的概念，所谓的容器就是将常用的服务，封装起来，然后，用户只需要遵循一定的规则，就可以达到统一、灵活、安全、方便、快速的目的
  - 具有依赖注入功能的容器，负责实例化、定位、配置应用程序中的对象及建立这些对象间的依赖
- Bean的概念
  - 由IoC容器管理的那些组成你应用程序的对象我们就叫它Bean
  - Bean就是由Spring容器初始化、装配及管理的对象，除此之外，bean就与应用程序中的其他对象没有什么区别了
- 元数据BeanDefinition
  - 确定如何实例化Bean、管理Bean之间的依赖关系以及管理Bean，这就需要配置元数据，在Spring中由BeanDefinition代表
- 如何工作的
  - 以xml配置方式，简单解释
  - 1 准备配置文件
    - 配置文件中声明Bean定义也就是为Bean配置元数据。
  - 2 由IoC容器进行解析元数据
    - IoC容器的Bean Reader读取并解析配置文件，根据定义生成BeanDefinition配置元数据对象，IoC容器根据BeanDefinition进行实例化、配置及组装Bean
  - 3 实例化IoC容器
    - 由客户端实例化容器，获取需要的Bean
- hello world

```
@Test
  public void testHelloWorld() {
    //1、读取配置文件实例化一个IoC容器
    ApplicationContext context = new ClassPathXmlApplicationContext("helloworld.xml");
     //2、从容器中获取Bean，注意此处完全“面向接口编程，而不是面向实现”
    HelloApi helloApi = context.getBean("hello", HelloApi.class);
     //3、执行业务逻辑
    helloApi.sayHello();
  }
```

😊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

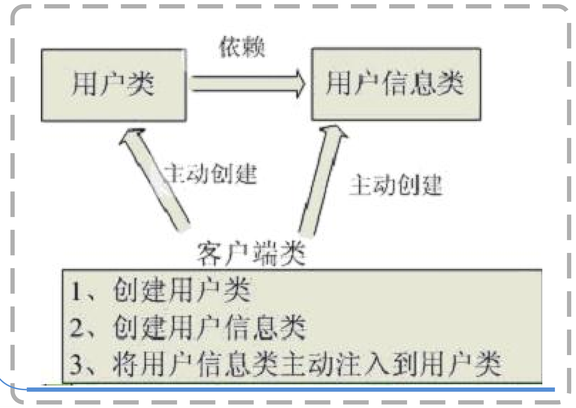
- 传统应用程序是由我们自己在对象中主动控制去直接获取依赖对象，也就是正转
- 而反转则是由容器来帮忙创建及注入依赖对象

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