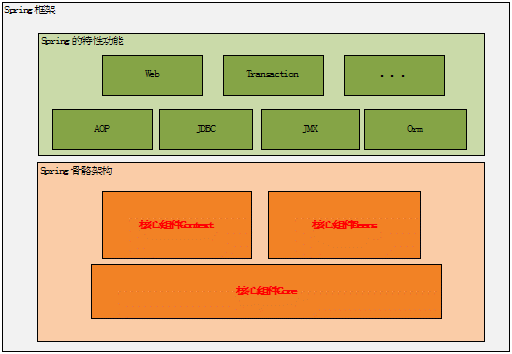
# Spring

# Spring的骨骼架构

Spring总共有十几个组件,但真正核心的组件只有几个,下面是spring框架的总体架构图:



核心有三个

Core context beans

# 设计理念

前面介绍了Spring的三个核心组件，如果再在它们三个中选出核心的话，那就非Beans组件莫属了，为何这样说，其实Spring就是面向Bean的编程（BOP,Bean Oriented Programming），Bean在Spring 中才是真正的主角。

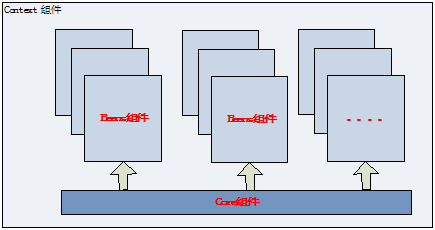
Bean在Spring中作用就像Object对OOP的意义一样，没有对象的概念就像没有面向对象编程，Spring中没有Bean也就没有Spring存在的意义。就像一次演出舞台都准备好了但是却没有演员一样。为什 么要Bean这种角色Bean或者为何在Spring如此重要，这由Spring框架的设计目标决定，Spring为何如此流行，我们用Spring的原因是什么，想想你会发现原来Spring解决了一个非常关键的问题他可以让 你把对象之间的依赖关系转而用配置文件来管理，也就是他的依赖注入机制。而这个注入关系在一个叫Ioc容器中管理，那Ioc容器中有又是什么就是被Bean包裹的对象。Spring正是通过把对象包装在 Bean中而达到对这些对象管理以及一些列额外操作的目的。

它这种设计策略完全类似于Java实现OOP的设计理念，当然了Java本身的设计要比Spring复杂太多太多，但是都是构建一个数据结构，然后根据这个数据结构设计他的生存环境，并让它在这个环境中 按照一定的规律在不停的运动，在它们的不停运动中设计一系列与环境或者与其他个体完成信息交换。这样想来回过头想想我们用到的其他框架都是大慨类似的设计理念。

前面说Bean是Spring中关键因素，那Context和Core又有何作用呢？前面吧Bean比作一场演出中的演员的话，那Context就是这场演出的舞台背景，而Core应该就是演出的道具了。只有他们在一起才能 具备能演出一场好戏的最基本的条件。当然有最基本的条件还不能使这场演出脱颖而出，还要他表演的节目足够的精彩，这些节目就是Spring能提供的特色功能了。

我们知道Bean包装的是Object，而Object必然有数据，如何给这些数据提供生存环境就是Context要解决的问题，对Context来说他就是要发现每个Bean之间的关系，为它们建立这种关系并且要维护好 这种关系。所以Context就是一个Bean关系的集合，这个关系集合又叫Ioc容器，一旦建立起这个Ioc容器后Spring就可以为你工作了。那Core组件又有什么用武之地呢？其实Core就是发现、建立和维护每 个Bean之间的关系所需要的一些列的工具，从这个角度看来，Core这个组件叫Util更能让你理解。

Beans Core Context 三个组件之间的关系图

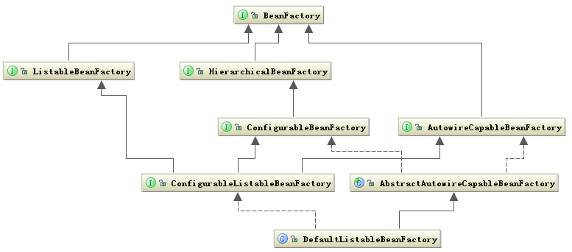


# 核心组件详解

## Bean组件

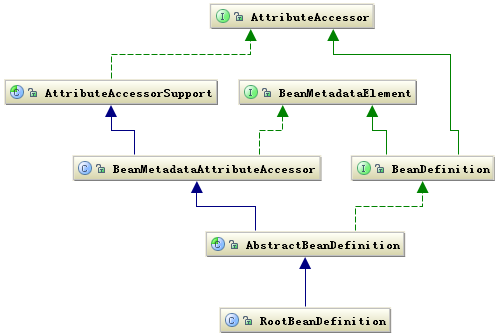
前面已经说明了Bean组件对Spring的重要性，下面看看Bean这个组件式怎么设计的。Bean组件在Spring的org.springframework.beans包下。这个包下的所有类主要解决了三件事：Bean的定义、Bean 的创建以及对Bean的解析。对Spring的使用者来说唯一需要关心的就是Bean的创建，其他两个由Spring在内部帮你完成了，对你来说是透明的。

Springbean的创建是典型的工厂模式，他的顶级接口是BeanFactory,下图是工厂的继承的层次关系:



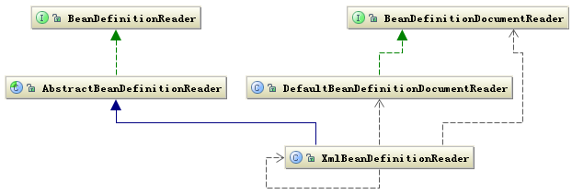
BeanFactory有三个子类：ListableBeanFactory、HierarchicalBeanFactory和Autowire Capable Bean Factory。但是从上图中我们可以发现最终的默认实现类是DefaultListableBeanFactory，他实 现了所有的接口。那为何要定义这么多层次的接口呢？查阅这些接口的源码和说明发现，每个接口都有他使用的场合，它主要是为了区分在Spring内部在操作过程中对象的传递和转化过程中，对对象的 数据访问所做的限制。例如ListableBeanFactory接口表示这些Bean是可列表的，而HierarchicalBeanFactory表示的是这些Bean是有继承关系的，也就是每个Bean有可能有父Bean。 AutowireCapableBeanFactory接口定义Bean的自动装配规则。这四个接口共同定义了Bean的集合、Bean之间的关系、以及Bean行为。

Bean的定义主要有BeanDefinition描述，



Bean的定义就是完整的描述了在Spring的配置文件中你定义的节点中所有的信息，包括各种子节点。当Spring成功解析你定义的一个节点后，在Spring的内部他就被转化 成BeanDefinition对象。以后所有的操作都是对这个对象完成的。

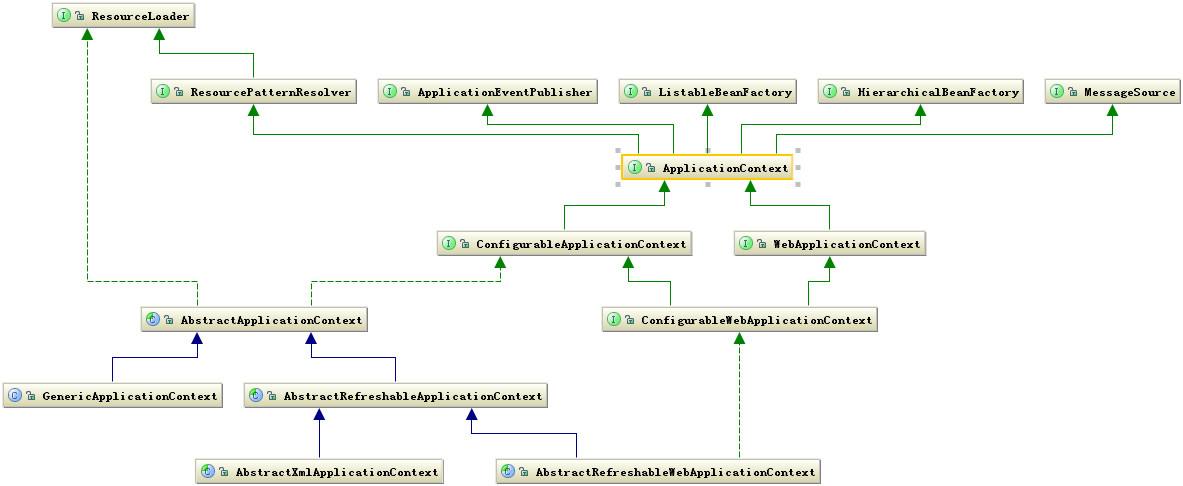
Bean的解析过程非常复杂，功能被分的很细，因为这里需要被扩展的地方很多，必须保证有足够的灵活性，以应对可能的变化。Bean的解析主要就是对Spring配置文件的解析。这个解析过程主要通过 下图中的类完成：



## **Context组件**

Context在Spring的org.springframework.context包下，前面已经讲解了Context组件在Spring中的作用，他实际上就是给Spring提供一个运行时的环境，用以保存各个对象的状态。下面看一下这个 环境是如何构建的。

ApplicationContext是Context的顶级父类，他除了能标识一个应用环境的基本信息外，他还继承了五个接口，这五个接口主要是扩展了Context的功能。下面是Context的类结构图：



从上图中可以看出ApplicationContext继承了BeanFactory，这也说明了Spring容器中运行的主体对象是Bean，另外ApplicationContext继承了ResourceLoader接口，使得ApplicationContext可以访 问到任何外部资源，这将在Core中详细说明。

**ApplicationContext的子类主要包含两个方面：**

ConfigurableApplicationContext表示该Context是可修改的，也就是在构建Context中用户可以动态添加或修改已有的配置信息，它下面又有多个子类，其中最经常使用的是可更新的Context，即 AbstractRefreshableApplicationContext类。

WebApplicationContext顾名思义，就是为web准备的Context他可以直接访问到ServletContext，通常情况下，这个接口使用的少。

再往下分就是按照构建Context的文件类型，接着就是访问Context的方式。这样一级一级构成了完整的Context等级层次。

总体来说ApplicationContext必须要完成以下几件事：

◆标识一个应用环境

◆利用BeanFactory创建Bean对象

◆保存对象关系表

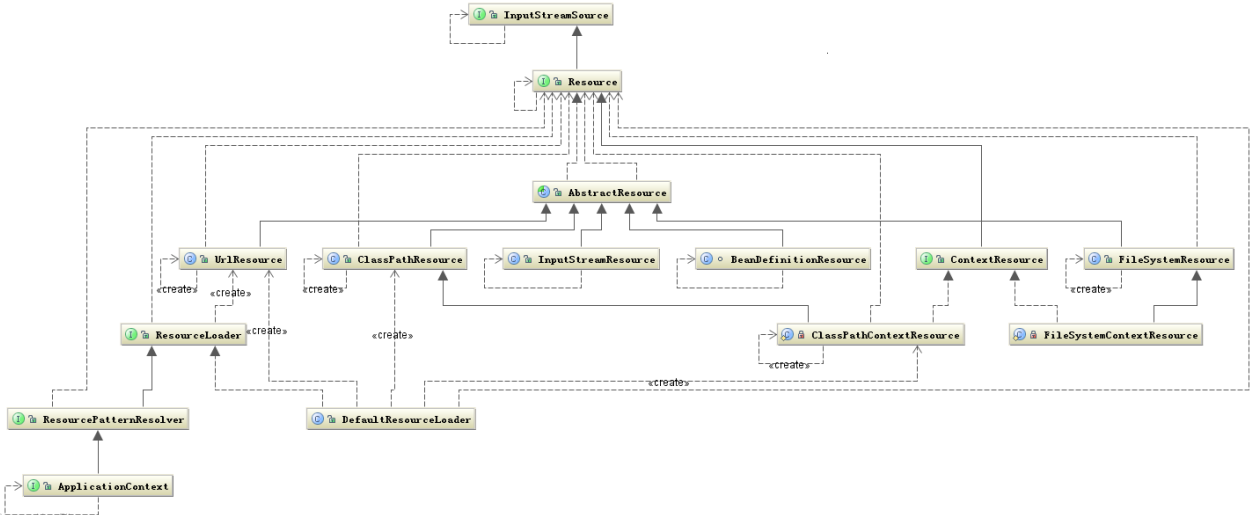
◆能够捕获各种事件

Context作为Spring的Ioc容器，基本上整合了Spring的大部分功能，或者说是大部分功能的基础。

## Core组件

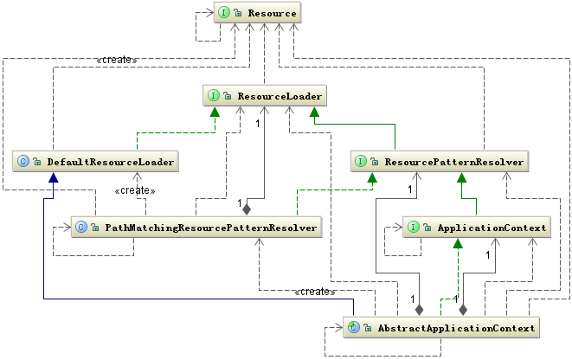
Core组件作为Spring的核心组件，他其中包含了很多的关键类，其中一个重要组成部分就是定义了资源的访问方式。这种把所有资源都抽象成一个接口的方式很值得在以后的设计中拿来学习。下面就 重要看一下这个部分在Spring的作用。

下图是Resource相关的类结构图：



从上图可以看出Resource接口封装了各种可能的资源类型，也就是对使用者来说屏蔽了文件类型的不同。对资源的提供者来说，如何把资源包装起来交给其他人用这也是一个问题，我们看到Resource 接口继承了InputStreamSource接口，这个接口中有个getInputStream方法，返回的是InputStream类。这样所有的资源都被可以通过InputStream这个类来获取，所以也屏蔽了资源的提供者。另外还有一 个问题就是加载资源的问题，也就是资源的加载者要统一，从上图中可以看出这个任务是由ResourceLoader接口完成，他屏蔽了所有的资源加载者的差异，只需要实现这个接口就可以加载所有的资源， 他的默认实现是DefaultResourceLoader。

下面看一下Context和Resource是如何建立关系的？首先看一下他们的类关系图



从上图可以看出，Context是把资源的加载、解析和描述工作委托给了ResourcePatternResolver类来完成，他相当于一个接头人，他把资源的加载、解析和资源的定义整合在一起便于其他组件使用。 Core组件中还有很多类似的方式。

# IOC容器如何工作

前面介绍了Core组件、Bean组件和Context组件的结构与相互关系，下面这里从使用者角度看一下他们是如何运行的，以及我们如何让Spring完成各种功能，Spring到底能有那些功能，这些功能是如 何得来的，下面介绍。

## **如何创建BeanFactory工厂**

正如图2描述的那样，Ioc容器实际上就是Context组件结合其他两个组件共同构建了一个Bean关系网，如何构建这个关系网？构建的入口就在AbstractApplicationContext类的refresh方法中。