Spring框架

版本 5.0.0.RC2

内容

* [1. 以Spring](https://docs.spring.io/spring/docs/5.0.0.RC2/spring-framework-reference/overview.html" \l "overview-getting-started-with-spring)开始
* [2. Spring](https://docs.spring.io/spring/docs/5.0.0.RC2/spring-framework-reference/overview.html" \l "overview) 框架的介绍
* [2.1.](https://docs.spring.io/spring/docs/5.0.0.RC2/spring-framework-reference/overview.html" \l "overview-dependency-injection) 依赖注入和控制反转
* [2.2. 框架](https://docs.spring.io/spring/docs/5.0.0.RC2/spring-framework-reference/overview.html" \l "overview-modules)模块
* [2.3. 使用](https://docs.spring.io/spring/docs/5.0.0.RC2/spring-framework-reference/overview.html" \l "overview-usagescenarios)场景

1. **以Spring开始**

此相关文档提供Spring框架的详细信息.提供了所有特征的相关文档,也提供了Spring拥护的背景基础概念(如”控制反转”).

如果你刚刚开始接触Spring,你可能想要使用Spring boot作为基础应用来使用Spring框架.Spring boot 提供了一个快速的(自用的)方式来创建Spring基础应用.它基于Spring框架,约定高于配置,被设计用于尽可能快速建立应用.

你可以通过 [start.spring.io](https://start.spring.io/) 来创建一个像”创建一个RESTful web服务”初始指导的基础应用.为了使你更好地理解,这些指导都聚焦于任务并且大部分都基于Spring boot.他们也其他来自Spring portfolio,你可以通过他们来解决一个特定问题.

1. **Spring框架的介绍**

Spring框架是一个为了开发Java应用来提供广泛基础建设支撑的Java平台.Spring解决了基础建设的问题,使你能专注于你的应用.

Spring能让你通过”简单Java对象”(POJOs)来创建应用,使得企业服务能不会侵入POJOs.这项功能应用于Java SE和Java EE.

作为一名应用开发人员,你可以通过Spring平台获得如下好处:

不使用事务接口在数据库事务中创建一个Java方法

不使用Servlet接口创建一个HTTP协议的本地Java方法

不使用JMS接口创建一个消息处理的本地Java方法

不使用JMX接口创建一个Java本地方法来进行管理操作

**2.1. 依赖注入和控制反转**

一个Java应用-是某种约束的范围的术语,把应用植入n-tier,服务器端的企业项目-尤其是各项协同应用组成的对象.然后一个应用中的对象应该互相依赖.

尽管Java平台提供了应用开发功能,但它缺少组装基础编译成清晰整体的模块含义并将其留给了设计者和开发者.尽管你能使用像工厂,抽象工厂,建设者,装饰者等设计模式来搭建不同的类和对象实例去创建应用,这些设计模式能仅仅靠名字就能说明他们的用途,描述,如何去使用,如何去定位问题所在.设计模式使得最佳实践成为规范,使得你必须在你的应用中实现接口.

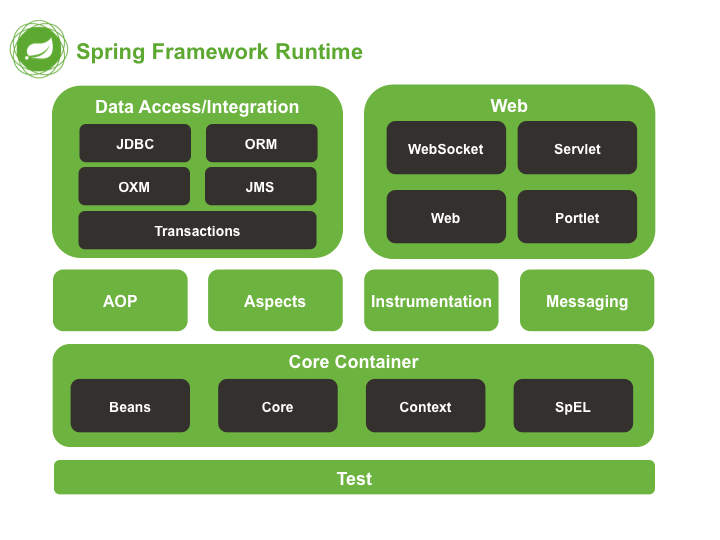
Spring框架的控制反转(IoC)组件将其他完全不同的组件组装成一个完整的待机工作应用.Spring框架将设计模式集成作为第一类对象使得你能协调自己的应用.各种组织和机构使用Spring框架使得工程师易于可持续开发应用.

**背景**

"问题是,他们反转了什么的控制?" 2004年,Martin Fowler在他的网站提出了关于控制反转的问题 . Fowler 提出了原则使其更加具有自我解释性并与依赖注入一起提出.

**2.2. 框架模块**

Spring的框架其本身由20左右个模块组成.这些模块如下视图中被分组为 核心容器, 数据基础, Web, AOP(面向切面编程),仪表,消息和测试.



*特征1. Spring框架的概述*

如下部分列出了可使用的模块,以提供给他们的作品名及他们设计的论题.作品名与在依赖管理工具中使用的作品ID相关联.

**2.2.1. 核心容器**

核心容器的构成为spring-core,spring-context-support等模块.

spring-beans和spring-core模块拥有IoC控制反转功能和依赖注入功能.BeanFactory是一个由工厂设计模式下的巧妙实现.它移除了需要编写的单例需求并且允许你在实际编程中使用配置和特殊依赖.

上下文(spring-context)模块由Core和Beans构建的基础模块:它像是JNDI方法,使用框架形式来获取对象.上下文模块继承了来自Beans模块的特点并且加入了国际化的支持(使用如,资源捆绑),事务的拓展,资源的加载及透明上下文创造,比如Servlet容器.上下文模块也支持JavaEE特征如EJB及基础信道.

上下文模块的核心要点是应用上下文接口.spring-context-support提供集成常规第三方类库到spring应用上下文的支持.

Spring-expression模块在运行时使用了表达式语言功能来查询和控制对象.它是统一表达式语言(EL)的拓展,如JSP2.1规格.该语言支持设置和获取值,对象属性,方法调用,获取数组的内容,集合的序列,算数逻辑操作符,变量名和通过Spring的IoC容器校验对象名.它作为集合体同样也支持列出规划和选项.

**2.2.2. AOP和仪表**

spring-aop模块使用遵从AOP的面向切面编程来实现了自定义,如方法过滤和切点来减弱代码耦合,实现基础代码分离.使用资源等级的metadata基础建设,你也能将行为信息合并进你的代码,使用类似.NET特性的约定.

分离的spring-aspects模块提供了AspectJ的集成.

spring-instrum模块提供了类的仪表盘支持和类加载实现,而spring-instrument-tomcat模块包含了给Tomcat的Spring仪表盘代理.

#### **2.2.3. 消息**

Spring框架4 内含spring-messaging模块,在Spring综合项目汇中使用了消息,消息信道,消息处理等其他功能来作为一个基于消息的基础模块. 这个模块也包括了一个给消息到方法的映射注解集合,像是基于编程实体的SpringMVC注解.

**2.2.4. 数据集成**

数据集成包含了JDBC,OXM,JMS,ORM和事务模块.

spring-tx模块支持类的程序化和公开化的事务管理,来为你的所有POJOs实现特殊接口(简单Java对象).

spring-jdbc模块使用JDBC的抽象层来去除冗长的解析数据库的错误代码和JDBC代码.

spring-orm模块提供了集成层给常见的对象映射接口,包括JPA和Hibernate.使用spring-orm模块,你能在其他Spring提供的所有功能集合中使用这些O/R映射框架,如之前提到的简单公开事务管理功能.

对象/XML映射实现由spring-oxm来实现,比如JAXB,Castor,XStream等.

spring-jms模块(Java消息服务)包含了生产和消费消息功能.自从Spring Framework 4.1发布后,它提供了spring-messaging模块的集成.

#### **2.2.5. Web**

Web层包含了spring-web,spring-webmvc和spring-websocket模块.

spring-web提供了面向web的集成功能,如IoC容器的初始化和文件上传等.它也支持HTTP客户端及web相关的远程Spring内容.

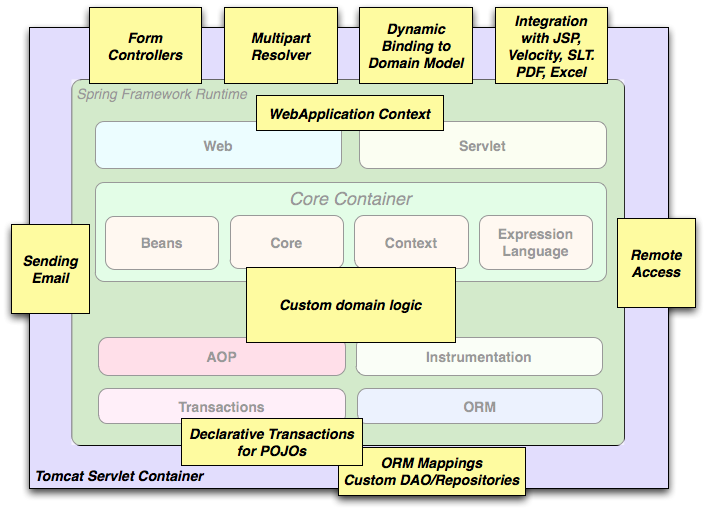
spring-mvc模块包含了Spring的实体(model)-视图(view)-控制(controller)和REST Web服务.Spring的MVC框架使用Spring框架提供了一个清晰的实体代码和web表单分离.

#### **2.2.6. 测试**

spring-test支持使用TestNG或者JUnit的单元测试和集成测试.它提供了Spring上下文和Spring的缓存持久性加载.它也提供了mock objects,使你能分开测试你的代码.

**2.3. 使用场景**

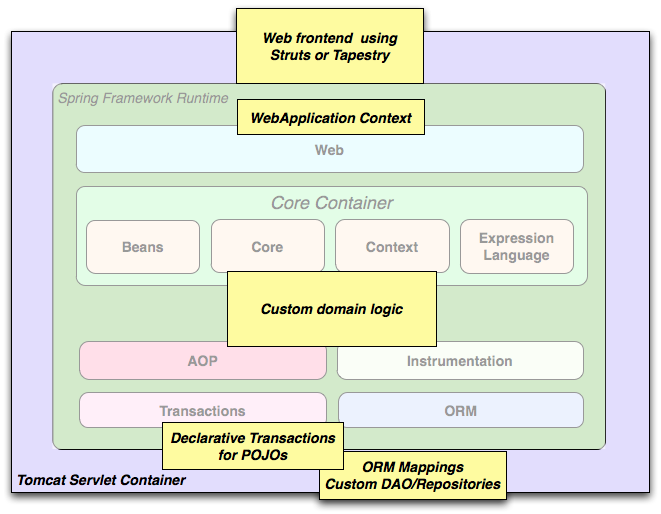
先前所述的构件块为Spring在许多场景中作出了合理的选择,从限制资源的嵌入式应用设备到完全成熟的商业应用,都使用了Spring事务管理基础建设和Web框架集成.



*特征2. 典型的完全成熟的Spring Web应用*

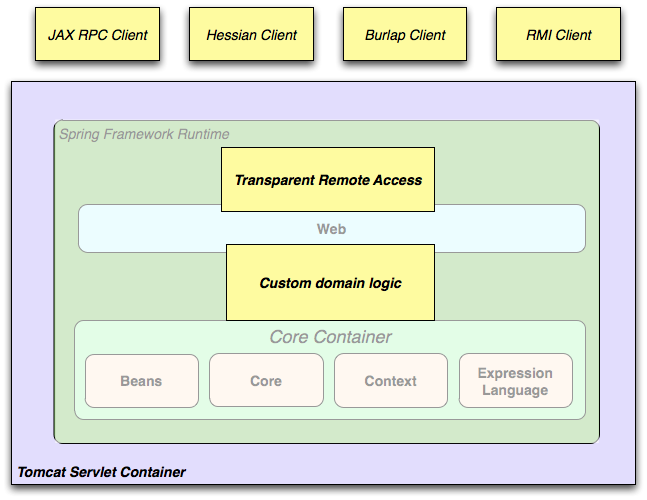
Spring的公开事务管理功能使得web应用完全的事务化,就如同你使用EJB容器管理事务.你所有的业务逻辑能够通过简单的POJOs和Spring的IoC容器管理实现.附加服务包括web层单独校验和email发送,能让你选择性地执行校验规则.Spring的ORM支持是由JPA和Hibernate集成的.当使用Hibernate时,你能可以继续使用你已存在的文件映射及其标准.

Hibernate会话工厂配置.表单控制无间隙地集成了web层和实体层,移除了ActionForms或其他HTTP参数到实体层的转换.



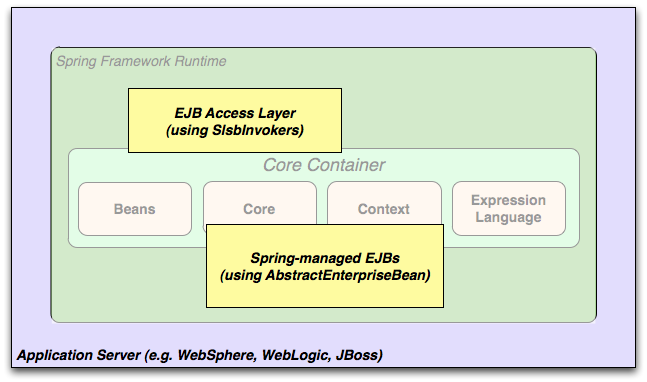
*特征3. 使用第三方web框架的Spring中间层*

有时候事实并不允许你使用完全不同的框架,Spring框架并不强制让你只使用Spring;它不是一个要么只用Spring或者要么不用Spring的解决方案.已经存在的框架如Structs框架,Tapestry框架等能够与Spring的中间层相结合,能让你使用Spring事务功能.你只要将你的业务逻辑使用一个应用上下文或Web应用上下文来集成到Web层.



*特征4. 远程使用场景*

当你需要通过web服务连接已经存在的代码时,你能使用Spring的Hessian,Rmi或HttpInvokerProxyFactoryBean类来开启远程,连接到已经存在的应用并不困难.



*特征5. EJBs - 包装已存在的POJOs*

Spring框架也为企业JavaBeans提供了连接抽象层,使你能重用已存在的POJOs并在会话beans中包装他们,在可拓展的,安全失败的web应用中,他们可能需要公开的安全.

#### **2.3.1. 依赖管理和命名规范**

依赖管理和依赖注入是不同的东西.为了导入Spring的优良功能到你的应用(像依赖注入)你需要分配所有的类库(jar文件),将他们导入到你的路径和编译环境内,这些依赖不是注入的实际组件,而是文件系统(典型的)内的资源.依赖管理处理设计这些资源的定位,存储和添加他们到路径.依赖能被直接导入(如,我的应用依赖Spring运行包)或间接导入(如,我的应用依赖commons-dbcp).间接导入的依赖被认为是传递的,它的依赖管理十分困难.

如果你准备使用Spring,去复制了它包含Spring小功能的jar类库.你需要将Spring模块打包,尽可能分离依赖,比如你不需要web应用,你就不需要spring-web模块.为了指向Spring类库模块,我们使用命名规范spring-\*或者spring-\*.jar等名字.实际的jar文件名字就和你平常使用的带有版本号的模块名一样(如,spring-core-5.0.0.RC2.jar).

每个Spring框架的release版本将会发布在以下几个地方:

Maven中央仓库,其并不需要做任何特殊配置,是默认的仓库.许多通用的Spring依赖的类库都可以从Maven中央仓库获取,你也可以选择不同版本的Spring版本,这十分方便.jar文件的名字使用spring-\*-<版本号>.jar且Maven的groupId为org.springframework.

在公共的Maven仓库,有附加的最终GA版本,这个jar文件命名和中央仓库的一样,为的是可以让开发人员获取开发版本的Spring,或轻松下载到捆绑的Spring zip文件.

首先你需要知道的是如何管理你的依赖:我们通常推荐使用一个自动的管理系统如Maven,Gradle或者Ivy,不过你也可以通过手动下载jar文件来达成你的目的.

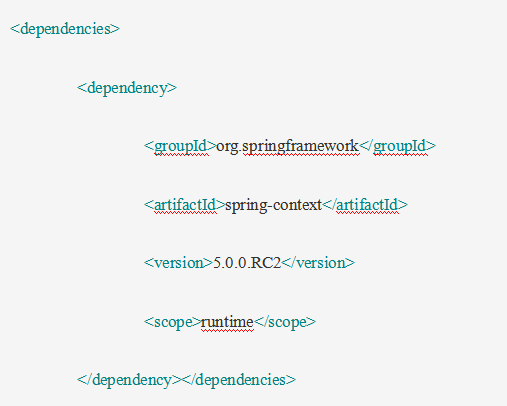
##### Spring依赖与依赖Spring

尽管Spring集成和支持很大的商业应用场景和其他拓展工具,但它也保持了最小依赖的绝对约束:你不应该下载和手动定位(尤其是自动下载)一个大的jar类库来使用Spring框架.为了基础依赖注入,仅仅只能有一个强制额外的依赖-日志(见下方详细介绍).

接下来我们简单叙述Spring的应用配置.首先考虑使用Maven,其次考虑使用Gradle,最后考虑使用Ivy.所有情况中,如果有东西不清楚,可以参考依赖系统的文档,或者查询一些样例代码,Spring本身使用Gradle来管理依赖,而且我们的样例大部分使用Gradle或者Maven.

##### **Maven Dependency Management**

如果你在使用Maven作为依赖管理工具,你甚至不需要手动添加日志依赖.打个比方,创建一个应用并使用依赖注入来配置应用,你的Maven应该如下所示:



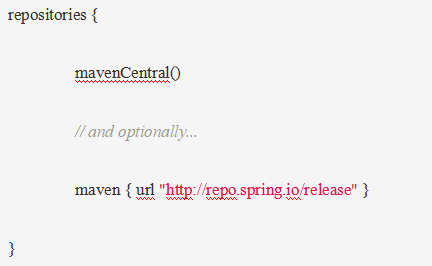
没错,在运行状态下,你并不需要编译Spring接口,你可以特地只使用基本的依赖注入.

上方是Maven中央仓库的样例.想要使用开发版的Spring Maven仓库,你需要在你的Maven配置中特殊配置仓库地址来使用.

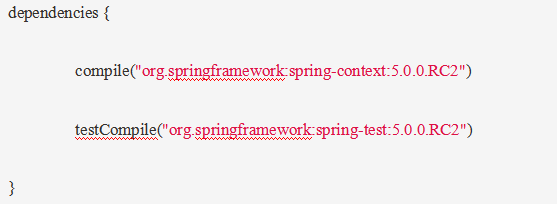


##### Gradle 依赖管理

如果要使用Gradle来构件Spring,需要在仓库中包含合适的URL.如下所示.



你可以改变仓库的URL来获取合适的版本(release版,里程碑版,snapshot版).一旦一个仓库被配置完毕,你能使用流行的的Gradle来配置他们.



##### Ivy 依赖管理

如果你更喜欢使用Ivy 来管理依赖,其配置是差不多相似的.

想要配置Ivy来获取Spring仓库,你需要在你的ivysetting.xml中添加如下代码.



##### 获取Zip 文件包

尽管使用依赖管理工具作为构件系统是管理Spring框架的推荐方式,但它仍然可以通过下载zip文件包来达成.

zip文件包发布在Spring Maven仓库中(存在Maven仓库中只是为了方便起见,你并非需要Maven来进行下载).

请打开浏览器进行下载:

[http://repo.spring.io/release/org/springframework/spring](https://repo.spring.io/release/org/springframework/spring)

选择你想要的文件和版本.文件包以-dist.zip作为后缀.比如spring-framework-{版本号}-RELEASE-dist.zip.文件包也同样有里程碑版和snapshots版.

#### **2.3.2. 日志**

Spring日志已经为Spring 5重新修订过:它仍然是基于Apache通用日志的接口,也被认为是Jakarta通用日志.然而,spring-core在spring-jcl模块中参考自一个自定义的通用日志桥接,使用一个Spring特定的日志工厂来手动或自动实现Log4j2,SLEFJ 的桥接,或者是JDK自带的日志(JUL). 这个实现就像是JCL-over-SLE4J桥接而非是类似JBoss通用日志目标的动态检测.

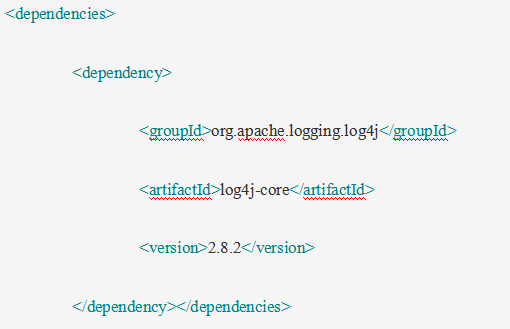
当然,好处是不需要再额外添加像是JCL-over-SLF4J,也不需要再手动排除标准的日志jar包.相反,他只是在Spring的自动探测模式中有效:只需要把Log4j 2.x或者SLF4J放在编译路径下,不需要额外的桥接jar包,或者默认的日志(JUL).另一点,默认的Hibernate将会使用相同的日志系统作为其输出日志.

如果Log4j和SLF4J两者同时加入依赖,那么Log4j日志系统将会被优先使用(因为它直接匹配JCL的签名和本地’致命’方法),类似JBoss的日志也会提供.Log4j会和SLF4J会互相协调,但请浏览Log4j和SLF4J的官方网站确保两者之间不会冲突.

##### **使用 Log4j 2.x**

[Log4j 2](https://logging.apache.org/log4j) 是Log4j它自身重写过的项目. 在Spring 5中, 当编译路径中存在Log4j 2.x时,嵌入的日志桥接会自动协调Log4j 2.x.

所以要使用Spring中的Log4j,你只需要将Log4j放到编译路径并提供其配置文件(log4j.xml,log4j.properties或其他支持的格式).如果是Maven用户,最小依赖为:



如果你想要使用SLF4J来代替Log4j,则所需依赖如下:



##### **通过Logback使用SLF4J**

Java(SLF4J)的简单日志外观是一个被其他Spring库常用的接口.特别是使用Logback(SLF4J接口的本地方法实现和Spring自动路径检测):



你也可以选择配置SLF4J来代替Log4j或JUL,尤其是其他默认使用SLF4J的其他库文件.特别之处,如果不是特别重要的话,请使用一致的日志外观;它最终只会被相同的日志所执行.所以当Spring直接切换到Log4j时,其他库可能会通过SLF4J绑定到Log4j或者是JUL.

##### **使用JUL (java.util.logging)**

在不提供Log4j或SLF4J接口的情况下,Spring会默认使用JUL做为日志系统.所以Spring并不需要特殊的依赖:只需使用Spring中包含依赖就能输出日志,也能在独立的应用中(通过JDK版本使用自定义或默认JUL设置)或是使用应用服务器的日志系统(及其系统的JUL设置).

JUL模块在JDK9版本不予提供,因为其不包含java.base.在Spring使用Log4j或SLF4J时,不再其应用场景使用JUL接口,会是的Spring会选择JUL作为默认的日志提供者,记住在java.logging模块下激活.

##### **WebSphere下的通用日志**

Spring应用在IBM的WebSphere应用服务器(WAS)下运行可能会出现两种情况:

在”parent first”类加载机制模型(WAS默认)下,应用汇一直使用服务器提供的通用日志版本,来代替WAS日志的子系统(实际是依赖JUL).一个应用依赖于不同的JCL,无论是Spring 5的或是JCL-over-SLF4J桥接,在被包含日志提供者的情况下会被有效地无视.

在”parent last”机制模型(默认是一个常用的Servlet容器而不是一个WAS外部配置的容器),一个应用提供的通用日志变体会被使用,在你的应用中,使得你会调用一个本地日志提供者,如Log4j或者是Logback.以防不调用本地的日志提供者,Spring(像是通用日志)或默认使用JUL,有效地在”parent first”情景下记录WebSphere的日志子系统.

最后,我们建议在”parent last”模型下部署Spring应用因为它会自然而然地考虑到本地提供者和服务器的日志子系统.

**核心技术**

这部分参考文档覆盖了Spring框架的所有技术.其中大部分是关于Spring框架的控制反转(IoC)容器,然后是与IoC容器紧密相关的面向切面编程(AOP)技术紧密结合的.Spring框架有它自己的AOP框架,十分易于理解,能成功覆盖80%的Java企业项目编程的范围.

Spring集成AspectJ(当前最好用的AOP)后,能成熟地实现Java企业项目.

## 1. IoC容器

### 1.1.IoC容器和Bean的介绍

这个章节主要介绍Spring框架是如何实现控制反转(IoC)的.IoC其本身也被认为是依赖注入(DI),能使得对象与其他对象之间传递构造参数,如使用工厂设计模式或是在对象构造时赋予其实例属性.容器会在bean生成后注入它的依赖,所以被叫作控制反转(IoC),bean本身能控制其实例化或直接使用类的构造来定位依赖,或是使用服务定位机制来达成.

org.springframework.context和org.springframework.beans包基于Spring框架的IoC容器的[BeanFactory](https://docs.spring.io/spring-framework/docs/5.0.0.RC2/javadoc-api/org/springframework/beans/factory/BeanFactory.html)接口,能够管理任何类型的对象.应用上下文 是一个BeanFactory的子接口.它添加了更简单的Spring的AOP功能:消息资源处理(用于国际化),事件发布;应用层特殊上下文,比如用于web应用的Web应用上下文.

简单的来说,BeanFactory提供了配置框架和基础的搭建功能,而且应用上下文增加了更多企业化的基础搭建功能.应用上下文是一个BeanFactory的完全superset,它除此之外用于Spring的IoC容器的描述.更多关于使用BeanFactory代替应用上下文的信息,请参考BeanFactory.

在Spring中,你的应用中对象的支柱是被Spring的IoC容器所管理的,统称为beans.一个bean是一个实例化的对象,被Spring的IoC容器所管理.换句话说,一个bean仅仅只是你的应用中众多对象中的一个对象.Beans中的依赖被配置容器使用的metadata所反射.

### **1.2. 容器简介**

org.springframework.context.ApplicationContext 接口提供了Spring的IoC容器并能进行对象实例化,配置和beans的分配.容器能够获取对象的实例,配置,和通过读取配置metadata来进行分配.metadata配置被卸载XML文件,Java注解或是Java代码中.它能使你在应用中表示对象的组成和对象之间的互相依赖.

几个应用上下文接口的实现被应用在Spring的包装外面.独立于应用,为的是创建[ClassPathXmlApplicationContext](https://docs.spring.io/spring-framework/docs/5.0.0.RC2/javadoc-api/org/springframework/context/support/ClassPathXmlApplicationContext.html) 或 [FileSystemXmlApplicationContext](https://docs.spring.io/spring-framework/docs/5.0.0.RC2/javadoc-api/org/springframework/context/support/FileSystemXmlApplicationContext.html) 实例.当使用XML的metadata配置后,你能通过使用Java注解或者Java代码作为metadata来构造容器,开启附加metadata格式的支持.

在大部分应用场景中,用户代码并不需要实例一个或多个Spring的IoC容器.比如在一个web应用场景中,在web.xml中,一个简单的8线样板XML描述能被满足.如果你在使用eclipse的Spring工具套装 ,你能够通过快捷键和简单的鼠标点击来其实创建配置.

下面的视图展示了Spring是如何工作的.你的应用类文件与metadata配置相结合,为的是在应用上下文创建和初始化后,你能可以进行完全的手动配置,执行系统或者应用.

