[SpringFramework详解（一](https://blog.csdn.net/cuihongyan867727/article/details/70147697?utm_medium=distribute.pc_relevant.none-task-blog-BlogCommendFromMachineLearnPai2-1.nonecase&depth_1-utm_source=distribute.pc_relevant.none-task-blog-BlogCommendFromMachineLearnPai2-1.nonecase)

A close up of a green screen

Description automatically generated

下面具体介绍每个部分的具体功能和组成模块：

#### 1.Core Container

由spring-core、spring-beans、spring-context、spring-context-support和spring-expression这几个模块组成。

* spring-core&spring-beans模块是整个框架的基础部分，主要特性是Ioc容器和DI，而Ioc容器的的核心类是BeanFactory接口，它是应用工厂模式的实现。
* spring-context模块是基于Core和Beans模块构建的，它继承了Beans模块的特性并增加了对国际化（例如使用资源包）、事件传递、资源加载和透明创建上下文（例如被servlet容器创建），同时也支持了Java EE的例如EJB、JMX和远程处理等的特性。其中ApplicationContext接口是整个模块的关键点。而spring-context-support模块提供了将第三方功能集成进Spring 应用上下文中，例如caching (EhCache, Guava, JCache), mailing (JavaMail), scheduling (CommonJ, Quartz) and template engines (FreeMarker, JasperReports, Velocity)。
* spring-expression模块为运行时查询和操纵对象图提供了强大的EL（基于JSP 2.1规范中定义的unified EL），支持存取属性值、实行分配、方法调用、访问数组及集合的内容、逻辑和算术运算、命名变量以及在Spring Ioc容器中通过名称查找对象。

#### 2.AOP & Intrumentation

由spring-aop、spring-aspects、spring-instrument和spring-instrument-tomcat模块组成。

* spring-aop模块通过使用aop提供的方法拦截器和切点来解耦具体业务代码和通用业务功能。其中spring-aspects模块是Spring在切面编程上对AspectJ的集成。
* spring-instrument模块提供了对特定应用服务器classloader和class instrumentation的支持，其中spring-instrument-tomcat模块提供了Spring对tomcat服务器的代理检测。

#### 3.Messaging

包括spring-message模块是Spring集成项目（例如 Message, MessageChannel, MessageHandler等）为基于消息传递的应用服务的关键抽象，包含一系列的从消息映射到方法的注解，类似于SpringMVC基于编程模型的注解。

#### 4.Data Access/Integration

数据访问和集成层由JDBC、ORM、OXM、JMS和Transaction模块组成。

* spring-jdbc模块提供了jdbc抽象层，抽离了繁琐的jdbc代码和针对特定数据库的错误代码。
* spring-tx模块对实现了特定接口的类提供了编程式和声明式事务管理，并且对所有的POJO类都支持这种特性。
* spring-orm模块集成了通用对象关系映射接口，包括JPA、JDO和Hibernate等。使用spring-orm模块可以使这些orm框架可以和Spring提供的其他所有特性联合使用，例如前边提到的简单声明式事务管理特性。
* spring-oxm模块提供了支持Object/XML 映射实现的抽象层，这些oxm实现包括JAXB, Castor, XMLBeans, JiBX and XStream。
* spring-jms模块包括生产和消费消息的特性。从Spring-Framework 4.1开始提供了和spring-message模块集成的支持。

#### [5.Web](http://5.web/)

Web层由spring-web、spring-webmvc、spring-websocket和spring-webmvc-portlet组成。

* spring-web模块提供了面向Web的基础功能的集成，例如多文件上传和使用Servlet监听Ioc容器的初始化特性，和面向Web的应用上下文。同时也包含了HTTP客户端以及Spring远程调用中和Web相关的部分。
* spring-webmvc（web-servlet）包含了spring mvc的实现和为web应用提供的REST 服务的实现。它使领域模型和web表单完全分离，同时也可以集成其它spring-framework的特性。
* spring-webmvc-portlet（web-portlet）提供了在Portlet环境中使用的mvc实现，并反映了spring-webmvc模块的功能。

#### 6.Test

spring-test模块通过JUnit和TestNG提供了单元测试和集成测试。它提供了一致性加载和缓存Spring上下文，也提供了用于单独测试使用的模拟对象。

[Spring核心组件详解](https://blog.csdn.net/wzk646795873/article/details/79764870?utm_medium=distribute.pc_relevant.none-task-blog-BlogCommendFromMachineLearnPai2-1.nonecase&depth_1-utm_source=distribute.pc_relevant.none-task-blog-BlogCommendFromMachineLearnPai2-1.nonecase)

Spring核心组件只有Core、Context、Beans三个。core包侧重于帮助类，操作工具，beans包更侧重于bean实例的描述。context更侧重全局控制，功能衍生。

[Spring核心知识详细教程](https://blog.csdn.net/Song_JiangTao/article/details/79949633?utm_medium=distribute.pc_relevant.none-task-blog-BlogCommendFromMachineLearnPai2-1.nonecase&depth_1-utm_source=distribute.pc_relevant.none-task-blog-BlogCommendFromMachineLearnPai2-1.nonecase)

## 现在我们来回答什么是spring这个问题

spring是一个容器框架，它可以接管web层，业务层，dao层，持久层的各个组件，并且可以配置各种bean， 并可以维护bean与bean的关系，当我们需要使用某个bean的时候，我们可以直接getBean(id)，使用即可

## ioc是什么？

ioc（inverse of control）控制反转：所谓反转就是把创建对象（bean）和维护对象（bean）的关系的权利从程序转移到spring的容器（spring-config.xml）

## di是什么？

di（dependency injection）依赖注入：实际上di和ioc是同一个概念，spring的设计者，认为di更准确的表示spring的核心

## 获取Bean

* **ApplicationContext 应用上下文容器取**

ApplicationContext ac = new ClassPathXmlApplicationContext("spring-config.xml");

当这句代码被执行，spring-config.xml文件中配置的bean就会被实例化。（但要注意bean的生命周期要为singleton），也就是说，不管没有getBean()，使用上下文容器获取bean，就会实例化该bean

* **Bean工厂容器取**

BeanFactory factory = new XmlBeanFactory(new ClassPathResource("spring-config.xml"));

这句代码被执行，spring-config.xml文件中配置的bean不会被实例化，即光实例化容器，并不会实例化bean  
而是在执行以下代码时才会被实例化，即使用bean的时候：  
factory.getBean("beanId");

[Spring FrameWork的bean加载机制（长博文）](https://blog.csdn.net/linfujian1999/article/details/83309353)

品Spring：真没想到，三十步才能完成一个bean实例的创建

<https://www.cnblogs.com/lixinjie/p/taste-spring-016.html>

在容器启动快完成时，会把所有的单例bean进行实例化，也可以叫做预先实例化。

这样做的好处之一是，可以及早地发现问题，及早的抛出异常，及早地解决掉。

本文就来看下整个的实例化过程。其实还是比较繁琐的。

DLR--前序遍历（根在前，从左往右，一棵树的根永远在左子树前面，左子树又永远在右子树前面 ）

LDR--中序遍历（根在中，从左往右，一棵树的左子树永远在根前面，根永远在右子树前面）

LRD--后序遍历（根在后，从左往右，一棵树的左子树永远在右子树前面，右子树永远在根前面）

# 【4】

https://www.cnblogs.com/coprince/p/8603492.html

# 【5】

https://blog.csdn.net/Butterfly\_resting/article/details/89735792

## spring bean 的生命周期

1.Spring 容器根据配置中的 bean 定义中实例化 bean。

2. Spring 使用依赖注入填充所有属性，如 bean 中所定义的配置。

3. 如果 bean 实现 BeanNameAware 接口，则工厂通过传递 bean 的 ID 来调用 setBeanName()。

4. 如果 bean 实现 BeanFactoryAware 接口，工厂通过传递自身的实例来调用 setBeanFactory()。

5. 如果存在与 bean 关联的任何 BeanPostProcessors，则调用 postProcessBeforeInitialization() 方法。

6. 如果为 bean 指定了 init 方法（ 的 init-method 属性），那么将调用它。

7. 最后，如果存在与 bean 关联的任何 BeanPostProcessors，则将调用 postProcessAfterInitialization() 方法。

8. 如果 bean 实现 DisposableBean 接口，当 spring 容器关闭时，会调用 destory()。

9. 如果为 bean 指定了 destroy 方法（ 的 destroy-method 属性），那么将调用它。

<https://www.jianshu.com/p/9ea61d204559>

抛开一些细节处理和扩展功能，一个 Bean 的创建过程无非是：

获取完整定义 -> 实例化 -> 依赖注入 -> 初始化 -> 类型转换。

作为一个完善的框架，Spring 需要考虑到各种可能性，还需要考虑到接入的扩展性。

所以有了复杂的循环依赖的解决，复杂的有参数构造器的匹配过程，有了 BeanPostProcessor 来对实例化或初始化的 Bean 进行扩展修改。

先有个整体设计的思维，再逐步击破针对这些特殊场景的设计，整个 Bean 加载流程迎刃而解。

## IOC底层实现原理

IoC，控制反转 ，是一种设计思想，对于spring框架来说，就是由spring来负责控制对象的生命周期和对象间的关系。 是说创建对象的控制权进行转移，以前创建对象的主动权和创建时机是由自己把控的，而现在这种权力转移到第三方。它是通过反射机制+工厂模式实现的，在实例化一个类时，它通过反射调用类中set方法将事先保存在Map中的类属性注入到类中。

DI—Dependency Injection，即“依赖注入”由容器动态的将某个依赖关系注入到组件之中。

## AOP底层实现原理

面向方面编程，利用一种称为“横切”的技术，剖解开封装的对象内部。将那些与业务无关，却为业务模块所共同调用的逻辑或责任封装起来。便于减少系统的重复代码，降低模块间的耦合度。实现AOP的技术，主要分为两大类：一是采用动态代理技术，Spring默认使用Jdk动态代理，如果目标类不是接口选择cglib动态代理，二是采用静态织入的方式。

## Spring MVC 运行流程

1）. 用户发请求–>DispatcherServlet，前端控制器收到请求后自己不进行处理，而是委托给其他的解析器进行处理，作为统一访问点，进行全局的流程控制。

2）.DispatcherServlet–>HandlerMapping，HandlerMapping将会把请求映射为HandlerExecutionChain对象（包含一个Handler处理器,多个HandlerInterceptor拦截器)。

3）.DispatcherServlet–>HandlerAdapter,HandlerAdapter将会把处理器包装为适配器，从而支持多种类型的处理器。

4）.HandlerAdapter–>处理器功能处理方法的调用，HandlerAdapter将会根据适配的结果调用真正的处理器的功能处理方法，完成功能处理，并返回一个ModelAndView对象(包含模型数据，逻辑视图名)

5）.ModelAndView的逻辑视图名–>ViewResolver，ViewResoler将把逻辑视图名解析为具体的View。

6）.View–>渲染，View会根据传进来的Model模型数据进行渲染，此处的Model实际是一个Map数据结构

7）.返回控制权给DispatcherServlet，由DispatcherServlet返回响应给用户。

