（Spring负责管理项目中的所有对象）

容器

一站式框架---{框架性质是属于容器性质的，容器中装什么对象就有什么功能}

Spring核心：控制反转（IoC）和面向切面编程（AOP）

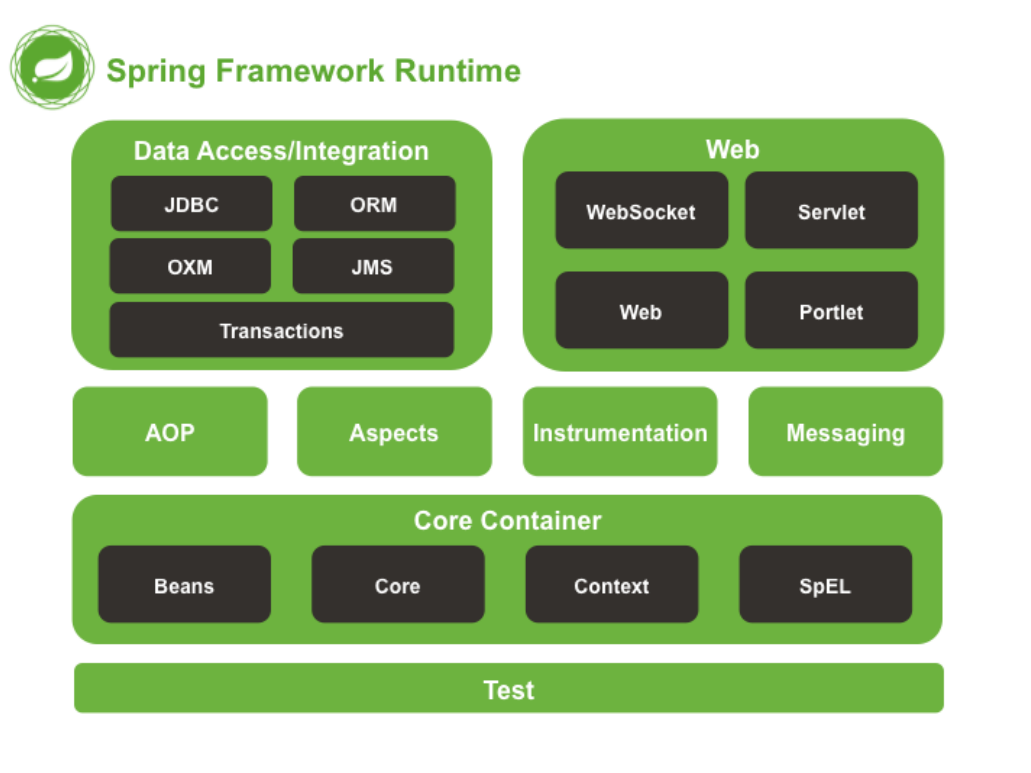
Spring的主要作用就是为代码“解耦”，降低代码间的耦合度。

根据功能的不同，可以将一个系统中的代码分为主业务逻辑与系统级业务逻辑两类。它们各自具有鲜明的特点：主业务代码间逻辑联系紧密，有具体的专业业务应用场景，复用性相对较低；系统级业务相对功能独立，没有具体的专业业务应用场景，主要是为主业务提供系统级服务，如日志、安全、事务等，复用性强。

Spring根据代码的功能特点，将降低耦合度的方式分为了两类：IoC与AOP。

IoC使得主业务在相互调用过程中，不用再自己维护关系了，即不用再自己创建要使用的对象了。而是由Spring容器统一管理，自动“注入”。而AOP使得系统级服务得到了最大复用，且不用再由程序员手工将系统级服务“混杂”到主业务逻辑中了，而是由Spring容器统一完成“织入”。

###### Spring体系结构



###### Spring的特点

1. **非侵入式：**Spring框架的API不会从业务逻辑上出现，即业务逻辑是POJO（Plain Old Java Object）。由于业务逻辑中没有Spring的API，所以业务逻辑可以从Spring框架快速的移植到其他框架，即与环境无关。
2. **容器：**Spring作为一个容器，可以管理对象的生命周期、对象与对象之间的依赖关系。可以通过配置文件，来定义对象，以及设置与其他对象的依赖关系。
3. **IoC：**控制反转（Inversion of Control），即创建被调用者的实例不是由调用者完成，而是由Spring容器完成，并注入调用者。

当应用了IoC，一个对象依赖的其它对象会通过被动的方式传递进来，而不是这个对象自己创建或者查找依赖对象。即，不是对象从容器中查找依赖，而是容器在对象初始化时不等对象请求就主动将依赖传递给它。

1. **AOP：**面向切面编程（AOP，Aspect Orient Programming），是一种编程思想，是面向对象编程OOP 的补充。很多框架都实现了对 AOP 编程思想的实现。Spring 也提供了面向切面编程的丰富支持，允许通过分离应用的业务逻辑与系统级服务（例如日志和事务管理）进行开发。应用对象只实现它们应该做的——完成业务逻辑——仅此而已。它们并不负责其它的系统级关注点，例如日志或事务支持。

我们可以把日志、安全、事务管理等服务理解成一个“切面”，那么以前这些服务一直是直接写在业务逻辑的代码当中的，这有两点不好：首先业务逻辑不纯净；其次这些服务被很多业务逻辑反复使用，完全可以剥离出来做到复用。那么AOP就是这些问题的解决方案，可以把这些服务剥离出来形成一个“切面”，以期复用，然后将“切面”动态的“织入”到业务逻辑中，让业务逻辑能够享受到此“切面”的服务。