纯xml配置：

JavaSE应用：

ApplicationContext applicationContext = new ClassPathXmlApplication(“bean.xml”);

BeanFactory与ApplicationContext的区别：

BeanFactory是SPringle框架中IoC容器定义的顶层接口，只用来定义一些基础功能和一些基础的规范，通常称BeanFactory是SpringIoC的基础容器；

ApplicationContext继承了BeanFactory，包含了BeanFactory中的所有功能，通常称之为springIoC是容器的高级接口，拥有更多的功能；

实例化Bean的三种方式：

1. 使用反射技术调用无参构造实例化Bean对象
2. 通过静态方法实例化对象：先创建一个工厂，在工厂中添加一个静态方法实例化 Bean对象
3. 通过实例化方法实例化对象：先在方法二中创建的工厂对象里添加获取Bean对象的 方法，实例化该工厂，然后调用工厂中实例化Bean对象的方法创建Bean对象

方法二、三一般用于当创建对象比较复杂不适合用方法一时使用。

Bean的生命周期：

Singletion(单例)：单例模式Bean的生命周期与容器相同

创建：当创建容器时，对象就被创建了；

存在：容器存在，对象就存在；

销毁：当容器销毁时，对象就被销毁了；

Prototype(原型)：原型模式Bean对象在容器创建时被创建，销毁时由GC回收

创建：当容器创建时，对象就被创建；

存在：对象被使用时，就一直存在；

销毁：当对象长时间不使用时，对象就被JAVA的GC回收了；

Xml和注解混合使用：

使用Xml配置bean：一般用于配置第三方jar包中的bean

使用注解配置bean：一般用于配置自定义的bean

@Autowirde 按类型注入，若无法锁定唯一对象（即当前接口有多个实现类时），则需搭配@Qualifier(“”)指定id使用。

@Resource 默认按照name注入（即按照id注入），可配置type属性，于JDK11中已移除。

纯注解模式：

@ConfiguRation:表示该类是一个配置类

@ComponentScan:开启扫描注解，并配置需要扫描的包

@Component:将pojo实例化到spring IoC容器中

@Controller：控制层pojo注入容器

@Service：服务层pojo注入容器

@Repository：持久层pojo注入容器

@Scope：设置pojo的作用域（singleton、prototype、request、session、globalsession ）

@PropertySource：引入外部资源文件

@Value：获取配置文件中的属性值

@Import:引入其他配置类，将当前类作为一个入口

@Bean：将方法返回的对象加入到SpringIoC容器中