1. 全注解的 Spring Ioc

控制反转IoC（Inversion of Control)

ApplicationContext是BeanFactory的子接口之一

@Configuration代表这是一个Java配置文件

@Bean用在方法上，表示将方法返回的java对象装配到IoC容器中，属性name定义这个bean的名称，没有配置就将方法名作为Bean名

通过扫描装配Bean：

@Component是标明哪个类被扫描进入Spring IoC容器

@ComponentScan标明采用何种策略去扫描装配

（basePackages="包名"，lazyInit=boolean（懒加载，默认false））

依赖注入（Dependency Injection)

@Autowired：可以标注属性、方法

首先根据类型找到对应的Bean，对应的类型不是唯一的那么它会根据其属性名和Bean的名称进行匹配，如果还是无法匹配会抛出异常

@Autowired（required = false）：不必须找到对应的Bean的注解

@Primary告诉Spring IoC容器，当发现有多个同样类型的Bean时，优先使用这个

使用属性文件

@PropertySource：定义对应的属性文件

Bean的作用域：

scope="singleton" 单例

scope="prototype" 多例

2. Spring AOP

在JDK中，提供了类Proxy的静态方法--newProxyInstance（）

```
public static Object newProxyInstance(ClassLoader loader,
                                       Class<?>[] interfaces,
                                       InvocationHandler h)
```

参数：类加载器，绑定的接口，绑定代理对象逻辑实现

AOP最典型的应用实际就是数据库事务的管控

@Transactional注解（基于Spring的动态代理机制）：实质是使用了JDBC的事务来进行事务控制，作用于接口实现类或者接口实现方法上

AOP开发详解
开发切面：切面类使用@Aspect来标注

正则表达式："execution（）"
切点：@Pointcut（正则表达式）标注方法，排除冗余的正则表达式，后面的通知注解可以使用其标注的方法表示

通知：@Before、@After、@AfterReturning、@AfterThrowing
无论是否发生异常，后置通知（After）都会被流程执行；发生异常时，异常通知（AfterThrowing）会被触发，返回通知（AfterReturning）不会被触发
环绕通知（最为强大）：@Around

Spring采用的动态代理是JDK与CGLIB：默认情况下，当你需要使用AOP的类拥有接口时，它会以JDK动态代理运行，否则以CGLIB运行

多个切面：先before的一定后after，可以使用@Order（数字等级比如1）注解来确定执行顺序

3. 访问数据库