1.集成druid时只需要制定datasource.type=DruidDataSource即可，其他数据库配置可使用druid的properties也可以使用spring原生的，效果一样

2.若要引入第三方的切面接口，只需要使用@Bean注解在方法中实例化切面类并返回，方法返回类型为Advisor接口，如此即可自动织入，已测试，暂不清楚使用Advisor的实现类或者继承接口是否可行

配置druidAdvisor方式如下：

@Bean("druidAdvisor")

public Advisor druidAdvisor(JdkRegexpMethodPointcut jrmp,DruidStatInterceptor dsi) {  
 Advisor adv = new DefaultPointcutAdvisor(jrmp, dsi);  
 return adv;  
}

3.自定义的切面程序，若有同样类型的切面方法指向同一个join point，那么其顺序是随机的，但可以通过人为设置其顺序。

1）实现 org.springframework.core.Ordered 接口，通过getOrder方法返回优先级数字，数字越小则优先级越高。

2）在类名上加入 @Order 注解，同样是数字越小优先级越高

但上述两种情况都是只适用于【不同的@Aspect 类内部有指向同一个join point的同类型（e.g. @Before , @After ）切面方法】，而你会发现，如果你在同一个类内部定义了两个@Before 方法，并且指向同一个point，那么即便在方法上声明 @Order 也是不起效果的，aop只会按照从上到下的顺序执行这两个方法

根据 spring-core reference 5.1.3.RELEASE 中 5.4.4. Declaring Advice → Advice Ordering 的描述，在这种情况下容器没有提供声明顺序的方法，开发者可以按照【从上到下】这个默认的顺序开发切面方法，也可以考虑每种类型的切面方法只定义一个，然后在内部使用责任链模式，来实现切面逻辑执行顺序的自定义

4.spring aop切面实例化模型。即可以自定义切面程序的实例化方式，默认为singleton单例，除此之外还有perthis 和 pertarget

1)perthis

例如

@Aspect("perthis(com.xyz.myapp.SystemArchitecture.businessService())")

以上例子表示，所有符合perthis括号内的表达式的join point都会实例化一个独立的aop对象。Aspect的实例化会在方法被执行的时候进行。应该也可以理解为所有的this切点都new一个aspect？（原文为：The aspect instance is created the first time that a method is invoked on the service object）

2)pertarget

pertarget用法与perthis类似，但对应于target的应用

这一块并不常用，而且reference说明也有点迷，留待以后有精力再深入。