1. spring事务失效的场景？

事务不生效：

1. 方法访问权限，spring要求被代理的方法必须是public的
2. 方法是final（static）修饰的，事务是aop实现的，jdk代理或者cgilib代理无法重写final方法
3. 方法内部调用，service的方法中调用另个事务的方法，没有经过aop代理
4. 未被spring管理
5. 多线程调用，不同的线程获取的数据连接不一样

事务不回滚：

1. 错误使用事务的传播机制，如使用了NEVER传播机制
2. 自己吞了异常，在代码中使用了try……catch，spring事务要能正常回滚，必须抛出它能处理的异常
3. 手动抛了别的异常，spring事务默认情况下只会回滚runtimeexception和error异常，不会回滚exception普通异常
4. 自定义了回滚异常，报错的异常与自定义的异常不同
5. spring事务传播机制
6. required：如果当前上下文存在事务，那么加入该事务，若没有，则新建个事务，这是默认的传播机制
7. supports：如果当前上下文存在事务，则支持事务加入事务，若没有，则使用非事务的方式执行
8. mandatory：如果当前上下文中存在事务，则抛除异常
9. requires\_new：每次都会新建一个事务，并且同时将上下文中的事务挂起，执行当前新建事务完成后，上下文事务在恢复执行
10. not\_supported：如果当前上下文中存在事务，则挂起当前事务，然后新的方法在没有事务的环境中执行
11. never：若当前上下文中存在事务，则抛出异常，否则在无事务环境上执行代码
12. nested：如果当前上下文中存在事务，则嵌套事务执行；若没有，则新建事务
13. spring框架中用到了哪些设计模式？

工厂设计模式：Spring使用工厂模式通过BeanFactory和ApplicationContext创建bean对象

代理设计模式：Spring AOP功能的实现

单例设计模式：Spring中的bean默认都是单例的

模板方法模式：Spring中的jdbcTemplate、hibernateTemplate、redisTemplate等

观察者模式：Spring事件驱动模型就是观察者模式的应用

适配器模式：Spring AOP的增强或通知使用到了设配器模式，Spring MVC中也是用到了适配器模式适配Controller

4、Spring MVC的工作原理？

（1）用户发送请求至前端控制器dispatcherservlet

（2）dispatchersevlet收到请求调用处理器映射器handerMapping

（3）handerMapping根据请求url找到具体的处理器，生成handerExecutionChain（包括处理器对象和处理器拦截器）一并返回给dispatcherServlet

（4）dispatcherSevlet根据处理器hander获取处理器适配器handerAdapter执行handerAdapter处理一系列操作，如：参数封装、格式转换、数据验证等

（5）执行处理器hander（Controller，也叫页面控制器）

（6）hander执行完成后返回modelAndView

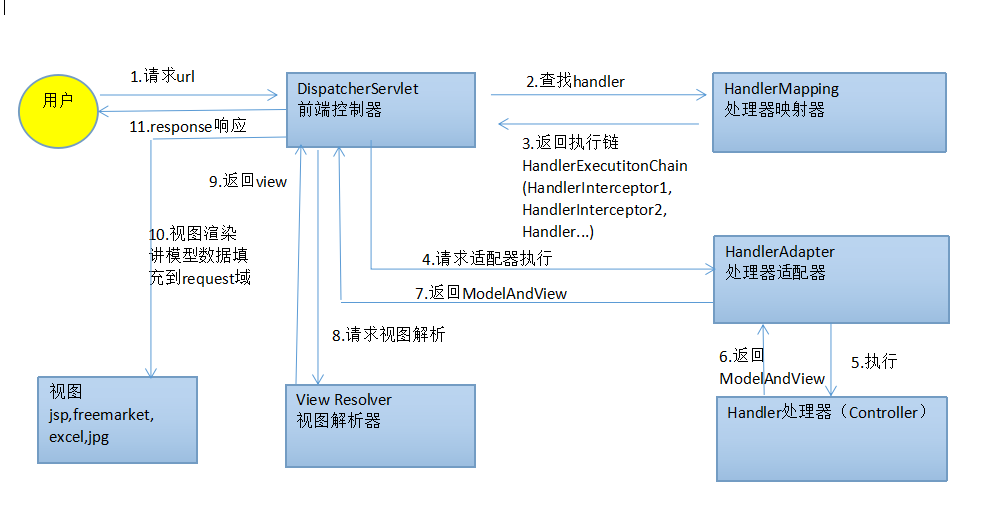
（7）handerAdapter将modelAndView返回到dispatcherServlet

（8）dispatcherServlet将modelAndView传给ViewReslover视图解析器

（9）viewReslover解析后返回具体view

（10）dispatcherServlet对view进行渲染（即将模型数据model填充到视图中）

（11）dispatcherServlet响应用户（把view返回给浏览器）



1. Spring中bena的生命周期

（1）Bean容器/BeanFactory 通过对象的构造器或工厂方法先实例化 Bean；

（2）再根据 Resource 中的信息再通过设定好的方法（典型的有setter，统称为BeanWrapper）对 Bean 设置属性值，得到 BeanDefintion 对象，然后 put 到 beanDefinitionMap 中，调用 getBean 的时候，从 beanDefinitionMap 里，拿出 Class 对象进行注入，同时，如果有依赖关系，将递归调用 getBean 方法，即依赖注入的过程。

（3）检查 xxxAware 相关接口，比如 BeanNameAware，BeanClassLoaderAware，ApplicationContextAware（ BeanFactoryAware）等等，如果有就调用相应的 setxxx 方法把所需要的xxx传入到 Bean 中。

**补充**：关于 Aware ，Aware 就是感知的意思， Aware 的目的是为了让Bean获得Spring容器的服务。 实现了这类接口的 bean 会存在“意识感”，从而让容器调用 setxxx 方法把所需要的 xxx 传到 Bean 中。

（4）此时检查是否存在有于 Bean 关联的任何 BeanPostProcessors， 执行 postProcessBeforeInitialization() 方法（前置处理器）。

（5）如果 Bean 实现了InitializingBean接口（正在初始化的 Bean），执行 afterPropertiesSet() 方法。

（6）检查是否配置了自定义的 init-method 方法，如果有就调用。

（7）此时检查是否存在有于 Bean 关联的任何 BeanPostProcessors， 执行 postProcessAfterInitialization() 方法（后置处理器）。返回 wrapperBean（包装后的 Bean）。

（8）这时就可以开始使用 Bean 了，当容器关闭时，会检查 Bean 是否实现了 DisposableBean 接口，如果有就调用 destory() 方法。

（9）如果 Bean 配置文件中的定义包含 destroy-method 属性，执行指定的方法。

1. BeanFactory和ApplicationContext有什么区别

两者都可以通过xml装载bean，前者是延迟加载，用到的时候再去实例化bean；后者是在启动的时候就把bean全都实例化好了等待使用。

1. ApplicationContext的通常实现是什么
2. FileSystemXmlAppiicationContext：此容器从一个XML文件中加载beans的定义，XML bean配置文件的全路径名必须提供给它的构造函数
3. ClassPathXmlApplicationContext：此容器也从一个XML文件中加载beans的定义，这里你需要正确设置classpath，因为这个容器将在classpath里找bean的配置
4. WebXmlApplicationContext：此容器加载一个XML文件，该文件中定义了一个web应用里的所有bean