1. 执行refresh()方法。

2.1 refreshBeanFacotory():创建BeanFactory并根据路径数组使用DefinitionReader(一般是XmlDefinitionReader)来加载BeanDefinition。

2.2 根据类型创建BeanFactoryPostProcessor的Bean，并使用这些处理器来对BeanFacotory进行后期处理，这些处理主要是修改BeanDefinition。

2.3 根据类型创建BeanPostProcessor(初始化前置与后置处理器)，并将它们注册到BeanFactory的BeanPostProcessorMap中。ApplicationContextAwareProcessor的创建方式不是通过getBean，因为它需要ApplicationContext对象，而application对象的感知又只能通过ApplicationContextAwareProcessor本身去处理，因此只能通过new创建的方式为ApplicationContextAwareProcessor注入ApplicationContext。

2.4 初始化事件传播器EventMulticaster，并向EventMulticaster加入创建好了的监听器。

2.5 提前初始化不需要延迟初始化的Bean。

2.6 完成刷新，并发布ContextRefresh刷新完成事件。

2.7 可选择向JVM注册销毁的构造函数。销毁时会先发布ContextClose容器关闭事件，最后会从DisposableBeanMap中取出DisposableBean并执行它们的destroy方法。

3. 关于GetBean

3.1 首先尝试从singletonObjects缓存中取出Bean

3.2 若缓存中不存在，则会通过BeanDefinition创建Bean。首先会进行实例化前预解析(resolveBeforeInstantiation)，即使用InstantiationAwareBeanPostProcessor进行处理，若产生了Bean则会直接返回并执行初始化后置处理。AOP通过InstantiationAwareBeanPostProcessor实现代理并返回对象。

3.3 若预解析过程中没有产生代理Bean，则会走常规创建。常规创建根据入参找到构造器，并传给实例化策略，最后实例化策略会使用该构造器进行实例化。实例化策略有Cglib和JDK这两种。

3.4 属性填充会使用InstantiationAwareBeanPostProcessor来做预处理BeanDefinition的PropertyValue，这些处理包括占位符替换之类的。最后会进行属性填充，BenRefrence填充时会从BeanFactory中根据名字获取该Bean。

3.5 初始化时会先让感知类Aware的Bean感知到对应的属性，最后分别执行初始化前置处理、**初始化方法调用**、初始化后置处理。

3.6 如果单例Bean为DisposableBean且未注册进DisposableBeans时，会将该单例Bean注册进DisposableBeans。

3.7 初始化完成后，单例Bean会被加进singletonObjects的缓存中。

3.8 如果创建的Bean为FactoryBean，则首先会尝试从FactoryBeanObjectCache中获取Bean。如果缓存中不存在产品Bean，则使用FactoryBean的getObject()方法生产Bean。最后，如果FactoryBean生产的Bean为单例(FactoryBean.isSingleton()=true，默认为true)，则会将该产品Bean放入FactoryBeanObjectCache这一缓存中。