- 在启动Spring时，需要传入一个AppConfig.class给ApplicationContext，ApplicationContext会根据AppConfig类封装为一个BeanDefinition，这种BeanDefinition我们把它称为配置类BeanDefinition。
- ConfigurationClassPostProcessor中会把配置类BeanDefinition取出来
- 构造一个ConfigurationClassParser用来解析配置类BeanDefinition，并且会生成一个配置类对象ConfigurationClass
- 如果配置类上存在@Component注解，那么解析配置类中的内部类
- 如果配置类上存在@PropertySource注解，那么则解析该注解，并得到PropertySource对象，并添加到environment中去
- 如果配置类上存在@ComponentScan注解，那么则解析该注解，进行扫描，扫描得到一系列的BeanDefinition对象，然后判断这些BeanDefinition是不是也是配置类BeanDefinition，并且会生成对应的ConfigurationClass
- 如果配置类上存在@Import注解，那么则判断Import的类的类型：
    - 如果是ImportSelector，那么调用执行selectImports方法得到类名，然后在把这个类当做配置类进行解析
    - 如果是ImportBeanDefinitionRegistrar，那么则生成一个ImportBeanDefinitionRegistrar实例对象，并添加到配置类对象中（ConfigurationClass）的importBeanDefinitionRegistrars属性中。
- 如果配置类上存在@ImportResource注解，那么则把导入进来的资源路径存在配置类对象中的importedResources属性中。
- 如果配置类中存在@Bean的方法，那么则把这些方法封装为BeanMethod对象，并添加到配置类对象中的**beanMethods**属性中。
- 如果配置类实现了某些接口，则看这些接口内是否定义了@Bean的默认方法
- 如果配置类有父类，则把父类当做配置类进行解析
- AppConfig这个配置类会对应一个ConfigurationClass，同时在解析的过程中也会生成另外的一些ConfigurationClass，接下来就利用reader来进一步解析ConfigurationClass
    - 如果ConfigurationClass是通过@Import注解导入进来的，则把这个类生成一个BeanDefinition，同时解析这个类上@Scope,@Lazy等注解信息，并注册BeanDefinition
    - 如果ConfigurationClass中存在一些BeanMethod，也就是定义了一些@Bean，那么则解析这些@Bean，并生成对应的BeanDefinition，并注册
    - 如果ConfigurationClass中导入了一些资源文件，比如xx.xml，那么则解析这些xx.xml文件，得到并注册BeanDefinition
    - 如果ConfigurationClass中导入了一些ImportBeanDefinitionRegistrar，那么则执行对应的registerBeanDefinitions进行BeanDefinition的注册