## 创建AnnotationConfigApplicationContex

### 父类创建容器工具类

包含：DefaultListableBeanFactory、PathMatchingResourcePatternResolver

### 创建AnnotatedBeanDefinitionReader

期间会注册以下beandefinition进容器：

**internalConfigurationAnnotationProcessor(ConfigurationClassPostProcessor)**

**internalAutowiredAnnotationProcessor（AutowiredAnnotationBeanPostProcessor）**

**internalCommonAnnotationProcessor(CommonAnnotationBeanPostProcessor)**

internalEventListenerProcessor(EventListenerMethodProcessor) -用来处理@EventListener注解的

internalEventListenerFactory(DefaultEventListenerFactory) -用来处理@EventListener注解的

### 创建ClassPathBeanDefinitionScanner

期间会注册过滤器registerDefaultFilters，指定扫描已添加Component.class注解的类

设置environment

设置resourceLoader

## 调用register(config.class)

解析config成beandefinitioin,并注册进入容器

## 调用refresh()

### prepareRefresh()

#### initPropertySources()

具体看子类实现，例如初始化web应用中web.xml的配置

#### **getEnvironment().validateRequiredProperties()**

验证配置必填的属性

#### **添加容器监听器applicationListeners**

### obtainFreshBeanFactory()

判断是否可刷新容器，是的话执行销毁bean然后重新生成beanFactory;

### prepareBeanFactory(beanFactory)

1. 设置beanClassLoader
2. 设置beanExpressionResolver(el表达式解析器)
3. 添加一个propertyEditorRegistrar(ResourceEditorRegistrar)

备注：PropertyEditorRegistry 主要就是为了添加和查找PropertyEditor

1. 添加一个**BeanPostProcessor(ApplicationContextAwareProcessor)**

备注:ApplicationContextAwareProcessor用来处理EnvironmentAware、EmbeddedValueResolverAware等回调

1. 添加七个ignoreDependencyInterface

包含： EnvironmentAware.class

EmbeddedValueResolverAware.class

ResourceLoaderAware.class

ApplicationEventPublisherAware.class

MessageSourceAware.class

ApplicationContextAware.class

ApplicationStartupAware.class

意思是：实现类以上接口的bean且通过@Bean(byType、byName)注册，该bean的所有用于注入的set方法均无效

1. 6，执行四次registerResolvableDependency(M,N)

其中M包括BeanFactory.class、ResourceLoader.class、ApplicationEventPublisher.class、ApplicationContext.class

(意思是M在容器中注册多个不同子类，但是spring 只会atuowried出N)

1. 7,添加一个**BeanPostProcessor(ApplicationListenerDetector)**

ApplicationListenerDetector负责把ApplicantsListener类型的Bean注册到ApplicationContext中

1. 8,添加一个**BeanPostProcessor(LoadTimeWeaverAwareProcessor)**

AOP再讲,Aspectj本身是通过编译期进行代理的，在Spring中就跟LoadTimeWeaver有关;

设置TempClassLoader(ContextTypeMatchClassLoader)

9,注册四个单例bean进容器，包含：

**environment、systemProperties、systemEnvironment、applicationStartup**

### postProcessBeanFactory(beanFactory)

子类来设置一下BeanFactory

### **invokeBeanFactoryPostProcessors(beanFactory)重点**

**只要是执行：**

**ConfigurationClassPostProcessor.postProcessBeanDefinitionRegistry(registry);**

**ConfigurationClassPostProcessorpostProcessBeanFactory(beanFactory);**

进行@Component的扫描，扫描得到BeanDefinition，并注册到beanFactory中

### **registerBeanPostProcessors(beanFactory)重点**

将扫描到的BeanPostProcessors实例化并排序，并添加到BeanFactory的beanPostProcessors属性中去

### **initMessageSource();**

初始化ApplicationContext的MessageSource，要么是用户设置的，要么是DelegatingMessageSource

### **initApplicationEventMulticaster();**

ApplicationEventMulticaster是spring中事件广播器接口，负责事件的广播发布

### **onRefresh();**

初始化springMVC的主体工具接口ThemeSource的实现类

### **registerListeners();**

将容器中的监听器到事件广播器；

注册beanDefinition中的监听器到事件广播器；

如果存在earlyApplicationEvents则使用事件广播器发布

### **finishBeanFactoryInitialization(beanFactory);**

1，设置beanFactory的conversionService(如果容器已经存在)

2，设置默认的占位符解析器EmbeddedValueResolver

3，尽早初始化LoadTimeWeaverAware bean，以便尽早注册其转换器

4，TempClassLoader属性置为空(停止使用临时类加载器进行类型匹配)

5，freezeConfiguration()；允许缓存所有bean定义元数据，不需要进一步更改

6，preInstantiateSingletons()实例化非懒加载的单例Bean；

### **finishRefresh()；**

1，清空resourceCaches；

2，设置lifecycleProcessor；

3，调用LifecycleBean的start()；

4，发布ContextRefreshedEvent事件；

5，将 Spring容器注册到 LiveBeansView