

# 第二十章：Spring 应用上下文生命周期

小马哥 • mercyblitz

# Spring 应用上下文生命周期

---

1. Spring 应用上下文启动准备阶段
2. BeanFactory 创建阶段
3. BeanFactory 准备阶段
4. BeanFactory 后置处理阶段
5. BeanFactory 注册 BeanPostProcessor 阶段
6. 初始化内建 Bean: MessageSource
7. 初始化内建 Bean: Spring 事件广播器
8. Spring 应用上下文刷新阶段
9. Spring 事件监听器注册阶段
10. BeanFactory 初始化完成阶段



# Spring 应用上下文生命周期

---

- 11. Spring 应用上下刷新完成阶段
- 12. Spring 应用上下文启动阶段
- 13. Spring 应用上下文停止阶段
- 14. Spring 应用上下文关闭阶段
- 15. 面试题精选
- 16. 结束语



# Spring 应用上下文启动准备阶段

- `AbstractApplicationContext#prepareRefresh()` 方法
  - 启动时间 - `startupDate`
  - 状态标识 - `closed(false)`、`active(true)`
  - 初始化 `PropertySources` - `initPropertySources()`
  - 检验 `Environment` 中必须属性
  - 初始化事件监听器集合
  - 初始化早期 Spring 事件集合

# BeanFactory 创建阶段

- `AbstractApplicationContext#obtainFreshBeanFactory()` 方法
  - 刷新 Spring 应用上下文底层 BeanFactory - `refreshBeanFactory()`
    - 销毁或关闭 BeanFactory, 如果已存在的话
    - 创建 BeanFactory - `createBeanFactory()`
    - 设置 BeanFactory Id
    - 设置 “是否允许 BeanDefinition 重复定义” - `customizeBeanFactory(DefaultListableBeanFactory)`
    - 设置 “是否允许循环引用 (依赖)” - `customizeBeanFactory(DefaultListableBeanFactory)`
    - 加载 BeanDefinition - `loadBeanDefinitions(DefaultListableBeanFactory)` 方法
    - 关联新建 BeanFactory 到 Spring 应用上下文
- 返回 Spring 应用上下文底层 BeanFactory - `getBeanFactory()`

# BeanFactory 准备阶段

- `AbstractApplicationContext#prepareBeanFactory(ConfigurableListableBeanFactory)` 方法
  - 关联 `ClassLoader`
  - 设置 `Bean` 表达式处理器
  - 添加 `PropertyEditorRegistrar` 实现 - `ResourceEditorRegistrar`
  - 添加 `Aware` 回调接口 `BeanPostProcessor` 实现 - `ApplicationContextAwareProcessor`
  - 忽略 `Aware` 回调接口作为依赖注入接口
  - 注册 `ResolvableDependency` 对象 - `BeanFactory`、`ResourceLoader`、`ApplicationEventPublisher` 以及 `ApplicationContext`
  - 注册 `ApplicationListenerDetector` 对象
  - 注册 `LoadTimeWeaverAwareProcessor` 对象
  - 注册单例对象 - `Environment`、`Java System Properties` 以及 `OS 环境变量`

## BeanFactory 后置处理阶段

- `AbstractApplicationContext#postProcessBeanFactory(ConfigurableListableBeanFactory)` 方法
  - 由子类覆盖该方法
- `AbstractApplicationContext#invokeBeanFactoryPostProcessors(ConfigurableListableBeanFactory)` 方法
  - 调用 `BeanFactoryPostProcessor` 或 `BeanDefinitionRegistry` 后置处理方法
  - 注册 `LoadTimeWeaverAwareProcessor` 对象

# BeanFactory 注册 BeanPostProcessor 阶段

- `AbstractApplicationContext#registerBeanPostProcessors(ConfigurableListableBeanFactory)` 方法
  - 注册 `PriorityOrdered` 类型的 `BeanPostProcessor` Beans
  - 注册 `Ordered` 类型的 `BeanPostProcessor` Beans
  - 注册普通 `BeanPostProcessor` Beans
  - 注册 `MergedBeanDefinitionPostProcessor` Beans
  - 注册 `ApplicationListenerDetector` 对象



# 初始化内建 Bean: MessageSource

- AbstractApplicationContext#initMessageSource() 方法
  - 回顾章节 - 第十二章 Spring 国际化 - MessageSource内建依赖

# 初始化内建 Bean: Spring 事件广播器

- `AbstractApplicationContext#initApplicationEventMulticaster()` 方法
  - 回顾章节 - 第十七章 Spring 事件 - `ApplicationEventPublisher` 底层实现

# Spring 应用上下文刷新阶段

- `AbstractApplicationContext#onRefresh()` 方法
  - 子类覆盖该方法
    - `org.springframework.web.context.support.AbstractRefreshableWebApplicationContext#onRefresh()`
    - `org.springframework.web.context.support.GenericWebApplicationContext#onRefresh()`
    - `org.springframework.boot.web.reactive.context.ReactiveWebServerApplicationContext#onRefresh()`
    - `org.springframework.boot.web.servlet.context.ServletWebServerApplicationContext#onRefresh()`
    - `org.springframework.web.context.support.StaticWebApplicationContext#onRefresh()`

# Spring 事件监听器注册阶段

- `AbstractApplicationContext#registerListeners()` 方法
  - 添加当前应用上下文所关联的 `ApplicationListener` 对象（集合）
  - 添加 `BeanFactory` 所注册 `ApplicationListener` Beans
  - 广播早期 Spring 事件

# BeanFactory 初始化完成阶段

- AbstractApplicationContext#finishBeanFactoryInitialization(ConfigurableListableBeanFactory) 方法
  - BeanFactory 关联 ConversionService Bean, 如果存在
  - 添加 StringValueResolver 对象
  - 依赖查找 LoadTimeWeaverAware Bean
  - BeanFactory 临时 ClassLoader 置为 null
  - BeanFactory 冻结配置
  - BeanFactory 初始化非延迟单例 Beans

# Spring 应用上下刷新完成阶段

- `AbstractApplicationContext#finishRefresh()` 方法
  - 清除 `ResourceLoader` 缓存 - `clearResourceCaches()` @since 5.0
  - 初始化 `LifecycleProcessor` 对象 - `initLifecycleProcessor()`
  - 调用 `LifecycleProcessor#onRefresh()` 方法
  - 发布 Spring 应用上下文已刷新事件 - `ContextRefreshedEvent`
  - 向 `MBeanServer` 托管 Live Beans

# Spring 应用上下文启动阶段

- `AbstractApplicationContext#start()` 方法
  - 启动 `LifecycleProcessor`
    - 依赖查找 `Lifecycle Beans`
    - 启动 `Lifecycle Beans`
  - 发布 Spring 应用上下文已启动事件 - `ContextStartedEvent`

# Spring 应用上下文停止阶段

- `AbstractApplicationContext#stop()` 方法
  - 停止 `LifecycleProcessor`
    - 依赖查找 `Lifecycle Beans`
    - 停止 `Lifecycle Beans`
  - 发布 Spring 应用上下文已停止事件 - `ContextStoppedEvent`



# Spring 应用上下文关闭阶段

- `AbstractApplicationContext#close()` 方法
  - 状态标识: `active(false)`、`closed(true)`
  - Live Beans JMX 撤销托管
    - `LiveBeansView.unregisterApplicationContext(ConfigurableApplicationContext)`
  - 发布 Spring 应用上下文已关闭事件 - `ContextClosedEvent`
  - 关闭 `LifecycleProcessor`
    - 依赖查找 `Lifecycle Beans`
    - 停止 `Lifecycle Beans`
  - 销毁 Spring Beans
  - 关闭 `BeanFactory`
  - 回调 `onClose()`
  - 注册 `Shutdown Hook` 线程（如果曾注册）

# 面试题

沙雕面试题 - Spring 应用上下文生命周期有哪些阶段?



答:

- 刷新阶段 - `ConfigurableApplicationContext#refresh()`
- 启动阶段 - `ConfigurableApplicationContext#start()`
- 停止阶段 - `ConfigurableApplicationContext#stop()`
- 关闭阶段 - `ConfigurableApplicationContext#close()`

## 面试题

996 面试题 - Environment 完整的生命周期是怎样的？



答：代码演示

# 面试题

**劝退面试题** - Spring 应用上下文生命周期执行动作？

答：本章整体回顾

