Spring IOC

1. 相关类/接口

1.0 ApplicationContext

> ApplicationContext 继承自 BeanFactory，但是它不应该被理解为 BeanFactory 的实现类，而是说其内部持有一个实例化的 BeanFactory（DefaultListableBeanFactory）。以后所有的 BeanFactory 相关的操作其实是委托给这个实例来处理的。

1.1 BeanPostProcessor

参考：Spring BeanPostProcessor接口使用

1.2 ApplicationContextAwareProcessor

默认的Aware的子接口都在这里触发执行的（invokeAwareInterfaces方法）

1.3 ConversionService

参考：在文中搜索“ConversionService”

1.4 SmartInitializingSingleton

参考：SmartInitializingSingleton的作用和原理
源码：`DefaultListableBeanFactory$preInstantiateSingletons` 中

1.5 MergedBeanDefinitionPostProcessor

执行位置：`AbstractAutowireCapableBeanFactory$doCreateBean()` 中
`applyMergedBeanDefinitionPostProcessors(mbd, beanType, beanName);`

1.6 AbstractApplicationContext

1.6.1 父

类：
接口：

1.6.2 作用

- 关键方法 `refresh()`（`ClassPathXmlApplicationContext` 类）：

```
@Override
public void refresh() throws BeansException, IllegalStateException {
    synchronized (this.startupShutdownMonitor) {
        // 设置启动时间；
        // 初始化PropertySources，校验必填（非空）的property
        prepareRefresh();
```

```
        // 刷新（删掉旧的，创建个新的）BeanFactory（继承关系不变）；
        // 设置allowBeanDefinitionOverriding、allowCircularReferences；
        // 加载beanDefinitions（重要，加载解析xml配置文件）
        ConfigurableListableBeanFactory beanFactory = obtainFreshBeanFactory();

        // Prepare the bean factory for use in this context.
        prepareBeanFactory(beanFactory);

        try {
            // Allows post-processing of the bean factory in context subclasses.
            // 子类可重写此方法：此时beanDefinitions已经加载，但还没有创建beans，可对BeanFactory进行修改；
            postProcessBeanFactory(beanFactory);

            // Invoke factory processors registered as beans in the context.
            // 执行各种BeanDefinitionRegistryPostProcessor（包括根据Ordered对BeanFactoryPostProcessor进行排序）
            // 执行BeanFactoryPostProcessor
            invokeBeanFactoryPostProcessors(beanFactory);

            // 实例化BeanPostProcessor
            // 自动注入了ApplicationListenerDetector（检测ApplicationListener，并将ApplicationListener添加到applic
            registerBeanPostProcessors(beanFactory);

            // Initialize message source for this context.
            // 默认DelegatingMessageSource；
            initMessageSource();

            // 仅仅是Initialize event multicaster for this context.
            initApplicationEventMulticaster();

            // Initialize other special beans in specific context subclasses.
            // 默认实现为空，子类可实现自己的逻辑
            onRefresh();

            // Check for listener beans and register them.
            // 注册监听器，出发一部分早期事件（earlyApplicationEvents）的监听器
            registerListeners();

            // Instantiate all remaining (non-lazy-init) singletons.
            finishBeanFactoryInitialization(beanFactory);

            // Last step: publish corresponding event.
            finishRefresh();
        } catch (BeansException ex) {
            if (logger.isWarnEnabled()) {
                logger.warn("Exception encountered during context initialization - " +
                        "cancelling refresh attempt: " + ex);
            }

            // Destroy already created singletons to avoid dangling resources.
            destroyBeans();

            // Reset 'active' flag.
            cancelRefresh(ex);

            // Propagate exception to caller.
            throw ex;
        } finally {
            // Reset common introspection caches in Spring's core, since we
            // might not ever need metadata for singleton beans anymore...
            resetCommonCaches();
        }
    }
```

```
}
```

1.7 ServletContext

定义一系列的方法，供 `servlet` 连接到 `servlet container` 使用

- `The ServletContext object is contained within the ServletConfig object`

1.8 ServletConfig

1.9 RequestDispatcher

可能需要更大的篇幅

1.10 ResourceBundleMessageSource

默认的messageSource

1.10.1

- 和 `ResourceBundle` 类似，默认读取 `classpath` 下的文件；
- 如果message中没有参数，默认调用 `AbstractResourceBasedMessageSource` 中的逻辑进行消息解析；否则，使用JDK的 `MessageFormat` 解析message；
- resources的编码可以配置，默认是 `ISO-8859-1`；
- 关于 `JDK9+`，自行阅读此类的javadoc；

1.11 BeanDefinition

作用：定义了各种字段，用于组装bean实例；

1.11.1 在spring中的保存

- 保存地方： `BeanFactory`
- 保存方式： `BeanFactory` 的一个Map类型的字段，key= `beanName`，value=BeanDefinition（包含 `RootBeanDefinition` / `ChildBeanDefinition`）；
- `AbstractBeanFactory.getMergedBeanDefinition()` 方法

更多

参考：Spring IOC 容器源码分析、spring源码阅读4——BeanDefinition 、Spring依赖注入的三种方式（好的 坏的和丑的）、【Spring】浅谈spring为什么推荐使用构造器注入