笔记

2. 大纲
- 主要观点
- 实例
- 详情

3. 技巧
- 写下关键事实
- 切勿过度，量力而行
- 解答

分支主题 4

1 标记好@SpringBootApplication 注解

```
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.context.ConfigurableApplicationContext;

@SpringBootApplication   1
public class BeanLifeCycleApplication {
    public static void main(String[] args) {
        ConfigurableApplicationContext context = SpringApplication.run(BeanLifeCycleApplication.class, args);   3  2
        context.close();
    }
}
```

2 把启动类.class作为参数传入方法中

3 SpringApplication.run(BeanLifeCycleApplication.class, args)

启动run方法

(3.1) new SpringApplication(primarySources)

primarySources参数 就是启动类.class

(3.2) run(args)

```
public SpringApplication(ResourceLoader resourceLoader, Class<?>... primarySources) {   resourceLoader: null   primarySources: [class BeanLifeCycleApplication]
    this.resourceLoader = resourceLoader;   resourceLoader: null   resourceLoader: null
    Assert.notNull(primarySources, message: "PrimarySources must not be null");
    this.primarySources = new LinkedHashSet<>(Arrays.asList(primarySources));   primarySources: [class BeanLifeCycleApplication]
    this.webApplicationType = WebApplicationType.deduceFromClasspath();   webApplicationType: "SERVLET"
    this.bootstrapRegistryInitializers = new ArrayList<>(getSpringFactoriesInstances(BootstrapRegistryInitializer.class));   bootstrapRegistryInitializers: null
    setInitializers((Collection) getSpringFactoriesInstances(ApplicationContextInitializer.class));
    setListeners((Collection) getSpringFactoriesInstances(ApplicationListener.class));
    this.mainApplicationClass = deduceMainApplicationClass();
}
```

3.1.1

子主题 2

```
private <T> Collection<T> getSpringFactoriesInstances(Class<T> type, Class<?>[] parameterTypes, Object... args) {   type: "interface org.springframework.boot.SpringApplicationRunListener"   parameterTypes: [class org.springframework.boot.SpringApplication, class [Ljava.lang.String;]   args: [Ljava.lang.String;@1532]
    ClassLoader classLoader = getClassLoader();   classLoader: LauncherHelper$AppClassLoader@1532
    // Use names and ensure unique to protect against duplicates
    Set<String> names = new LinkedHashSet<>(SpringFactoriesLoader.loadFactoryNames(type, classLoader));   names: size = 1
    List<T> instances = createSpringFactoriesInstances(type, parameterTypes, classLoader, args, names);   type: "interface org.springframework.boot.SpringApplicationRunListener"   parameterTypes: [class org.springframework.boot.SpringApplication, class [Ljava.lang.String;]   classLoader: LauncherHelper$AppClassLoader@1532   args: [Ljava.lang.String;@1532   names: size = 1
    AnnotationAwareOrderComparator.sort(instances);
    return instances;   instances: size = 1
}
```

3.1.2

```
private static Map<String, List<String>> loadSpringFactories(ClassLoader classLoader) {
    Map<String, List<String>> result = cache.get(classLoader);
    if (result != null) {
        return result;
    }

    result = new HashMap<>();
    try {
        Enumeration<URL> urls = classLoader.getResources(FACTORIES_RESOURCE_LOCATION);
        while (urls.hasMoreElements()) {
            URL url = urls.nextElement();
            UrlResource resource = new UrlResource(url);
            Properties properties = PropertiesLoaderUtils.loadProperties(resource);
            for (Map.Entry<?, ?> entry : properties.entrySet()) {
                String factoryTypeName = ((String) entry.getKey()).trim();
                String[] factoryImplementationNames =
                        StringUtils.commaDelimitedListToStringArray((String) entry.getValue());
                for (String factoryImplementationName : factoryImplementationNames) {
                    result.computeIfAbsent(factoryTypeName, key -> new ArrayList<>())
                            .add(factoryImplementationName.trim());
                }
            }
        }

        // Replace all lists with unmodifiable lists containing unique elements
        result.replaceAll((factoryType, implementations) -> implementations.stream().distinct()
                .collect(Collectors.collectingAndThen(Collectors.toList(), Collections::unmodifiableList)));
        cache.put(classLoader, result);
    }
```

3.1.2

1）会遍历ClassLoader中所有jar包下的spring.factories文件中的类名，包括我们项目里自定义的spring.factories文件，将这些classname 放入缓存的map中

```
private <T> List<T> createSpringFactoriesInstances(Class<T> type, Class<?>[] parameterTypes,
        ClassLoader classLoader, Object[] args, Set<String> names) {
    List<T> instances = new ArrayList<>(names.size());
    for (String name : names) {
        try {
            Class<?> instanceClass = ClassUtils.forName(name, classLoader);
            Assert.isAssignable(type, instanceClass);
            Constructor<?> constructor = instanceClass.getDeclaredConstructor(parameterTypes);
            T instance = (T) BeanUtils.instantiateClass(constructor, args);
            instances.add(instance);
        }
        catch (Throwable ex) {
            throw new IllegalArgumentException("Cannot instantiate " + type + " : " + name, ex);
        }
    }
    return instances;
}
```

2）实例化这些类