spring.factories　启动加载原理

1，SpringApplication 构造器调用 setInitializers

在 SpringApplication.run(DemoApplication.class, args) 方法里面 构造器里面 调用了

this.setInitializers(this.getSpringFactoriesInstances(ApplicationContextInitializer.class));

构造器 代码如下：

```
public SpringApplication(ResourceLoader resourceLoader, Class... primarySources) {
    this.sources = new LinkedHashSet();
    this.bannerMode = Mode.CONSOLE;
    this.logStartupInfo = true;
    this.addCommandLineProperties = true;
    this.addConversionService = true;
    this.headless = true;
    this.registerShutdownHook = true;
    this.additionalProfiles = new HashSet();
    this.isCustomEnvironment = false;
    this.resourceLoader = resourceLoader;
    Assert.notNull(primarySources, "PrimarySources must not be null");
    this.primarySources = new LinkedHashSet(Arrays.asList(primarySources));
    this.webApplicationType = WebApplicationType.deduceFromClasspath();
    this.setInitializers(this.getSpringFactoriesInstances(ApplicationContextInitializer.class));
    this.setListeners(this.getSpringFactoriesInstances(ApplicationListener.class));
    this.mainApplicationClass = this.deduceMainApplicationClass();
}
```

2，getSpringFactoriesInstances 调用分析

在这个里面调用了 getSpringFactoriesInstances

```
private <T> Collection<T> getSpringFactoriesInstances(Class<T> type) {
    return this.getSpringFactoriesInstances(type, new Class[0]);
}
```

继续调用： createSpringFactoriesInstances 这个方法，如下 ： 调用了 loadFactoryNames

```
private <T> Collection<T> getSpringFactoriesInstances(Class<T> type, Class<?>[] parameterTypes, Object... args)
{
    ClassLoader classLoader = this.getClassLoader();
    Set<String> names = new LinkedHashSet(SpringFactoriesLoader.loadFactoryNames(type, classLoader));
    List<T> instances = this.createSpringFactoriesInstances(type, parameterTypes, classLoader, args, names);
    AnnotationAwareOrderComparator.sort(instances);
    return instances;
}
```

3，loadFactoryNames 里面 执行调用 分析：调用了 loadSpringFactories

```
public static List<String> loadFactoryNames(Class<?> factoryClass, @Nullable ClassLoader classLoader) {
    String factoryClassName = factoryClass.getName();
    return (List)loadSpringFactories(classLoader).getOrDefault(factoryClassName, Collections.emptyList());
}
```

4, loadSpringFactories 正式读取了 META-INF/spring.factories 文件中的内容

```
    Enumeration<URL> urls = classLoader != null ? classLoader.getResources("META-INF/spring.factories") :
    ClassLoader.getSystemResources("META-INF/spring.factories");
    LinkedMultiValueMap result = new LinkedMultiValueMap();
```

最后将 spring.factories 里面的（factoryClassName，factoryName）读取到 LinkedMultiValueMap 里面，
并且通过 cache.put(classLoader, result); 存放在 SpringFactoriesLoader.cache 里面

5，创建 factoryName 实例对象

在 createSpringFactoriesInstances 通过获取特定的构造函数 创建 factoryName 实例对象

```
    Constructor<?> constructor = instanceClass.getDeclaredConstructor(parameterTypes);
    T instance = BeanUtils.instantiateClass(constructor, args);
```