**启动加载类：**

**二种选一种使用即可，混用@Order失效，无法控制顺序**

**接口CommandLineRunner：**

**接口ApplicationRunner：**

**共同点**：其一执行时机都是在容器启动完成的时候进行执行；其二这两个接口中都有一个run()方法；

**不同点**：ApplicationRunner中run方法的参数为ApplicationArguments 解析后的命令行参数，而CommandLineRunner接口中run方法的参数为String数组，原始未解析的命令行参数。

**想要更详细地获取命令行参数，那就使用CommandLineRunner接口**

如果在SpringBoot应用启动的时候需要执行特定的动作，可以利用CommandLineRunner。实现了CommandLineRunner接口的类@Component加入IOC容器，会在所有Spring Beans都初始化之后，SpringApplication.run()之前执行，非常适合在应用程序启动之初进行一些数据初始化的工作。

**CommandLineRunner用法如下:**

import com.first.study.service.service.first.service.RedisService;  
import org.springframework.beans.factory.annotation.**Autowired**;  
import org.springframework.boot.CommandLineRunner;  
import org.springframework.core.annotation.**Order**;  
import org.springframework.stereotype.**Component**;**@Component**//添加进IOC容器  
**@Order**(value = 1)//设置启动优先级，越小优先级越高  
public class TestCommandLineRunner implements CommandLineRunner {  
 **@Autowired** RedisService redisService;  
 **@Override** public void run(String... args) throws Exception {  
 redisService.getHRData();  
 }  
}

**ApplicationRunner用法如下：**

import com.first.study.service.service.first.service.RedisService;  
import org.springframework.beans.factory.annotation.**Autowired**;  
import org.springframework.boot.ApplicationArguments;  
import org.springframework.boot.ApplicationRunner;  
import org.springframework.core.annotation.**Order**;  
import org.springframework.stereotype.**Component**;**@Component**  
**@Order**(value = 4)  
public class TestApplicationRunner implements ApplicationRunner{  
 **@Autowired** RedisService redisService;  
 **@Override** public void run(ApplicationArguments args) throws Exception {  
 System.*err*.println(**"ApplicationRunner..."**);  
 redisService.getHRData();  
 }  
}