# **SpringBoot ApplicationListener 监听执行顺序**



[maguangkun](https://blog.csdn.net/gaolengroujiamo" \t "https://blog.csdn.net/gaolengroujiamo/article/details/_blank) 2020-03-10 10:57:23 IMG_257 1797 IMG_258 收藏 1

文章标签： [spring boot](https://so.csdn.net/so/search/s.do?q=spring boot&t=blog&o=vip&s=&l=&f=&viparticle=" \t "https://blog.csdn.net/gaolengroujiamo/article/details/_blank) [spring](https://www.csdn.net/tags/MtTaEg0sMDg2NTAtYmxvZwO0O0OO0O0O.html" \t "https://blog.csdn.net/gaolengroujiamo/article/details/_blank)

版权

自己创建Listenert 实现 ApplicationListener

package com.mgk.demov1.listeners;

import com.mgk.demov1.annotation.Student;

import com.sun.org.apache.bcel.internal.generic.SWITCH;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.context.event.\*;

import org.springframework.context.ApplicationContext;

import org.springframework.context.ApplicationEvent;

import org.springframework.context.ApplicationListener;

import org.springframework.context.ConfigurableApplicationContext;

import org.springframework.context.event.ContextClosedEvent;

import org.springframework.context.event.ContextRefreshedEvent;

import org.springframework.context.event.ContextStartedEvent;

import org.springframework.context.event.ContextStoppedEvent;

public class MyListenery implements ApplicationListener {

@Override

public void onApplicationEvent(ApplicationEvent event) {

*// ApplicationStartingEvent//启动开始的时候执行的事件*

*// ApplicationEnvironmentPreparedEvent//上下文创建之前运行的事件*

*// ApplicationContextInitializedEvent//*

*// ApplicationPreparedEvent//上下文创建完成，注入的bean还没加载完成*

*// ContextRefreshedEvent//上下文刷新*

*// ServletWebServerInitializedEvent//web服务器初始化*

*// ApplicationStartedEvent//*

*// ApplicationReadyEvent//启动成功*

*// ApplicationFailedEvent//在启动Spring发生异常时触发*

switch (event.getClass().getSimpleName()){

case "ApplicationStartingEvent":

System.out.println("启动开始的时候执行的事件");

break;

case "ApplicationEnvironmentPreparedEvent":

System.out.println("上下文创建之前运行的事件");

break;

case "ApplicationContextInitializedEvent":

System.out.println("上下文初始化");

break;

case "ApplicationPreparedEvent":

System.out.println("上下文创建完成，注入的bean还没加载完成");

break;

case "ContextRefreshedEvent":

System.out.println("上下文刷新");

if( event instanceof ContextRefreshedEvent){

Object stu = ((ContextRefreshedEvent) event).getApplicationContext().getBean("stu");

System.out.println(stu);

}

break;

case "ApplicationStartedEvent":

System.out.println("ApplicationStartedEvent");

break;

case "ApplicationReadyEvent":

System.out.println("启动成功");

break;

case "ApplicationFailedEvent":

break;

}

}

}

package com.mgk.demov1.annotation;

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.Configuration;

@Configuration

public class Configurtion {

@Bean(name = "stu")

public Student getStudent(){

return new Student();

}

}

也可以单独去实现其中一个监听事件，来解决业务具体操作

package com.mgk.demov1.listeners;

import org.springframework.context.ApplicationContext;

import org.springframework.context.ApplicationListener;

import org.springframework.context.event.ContextRefreshedEvent;

public class MyListenerv2 implements ApplicationListener<ContextRefreshedEvent> {

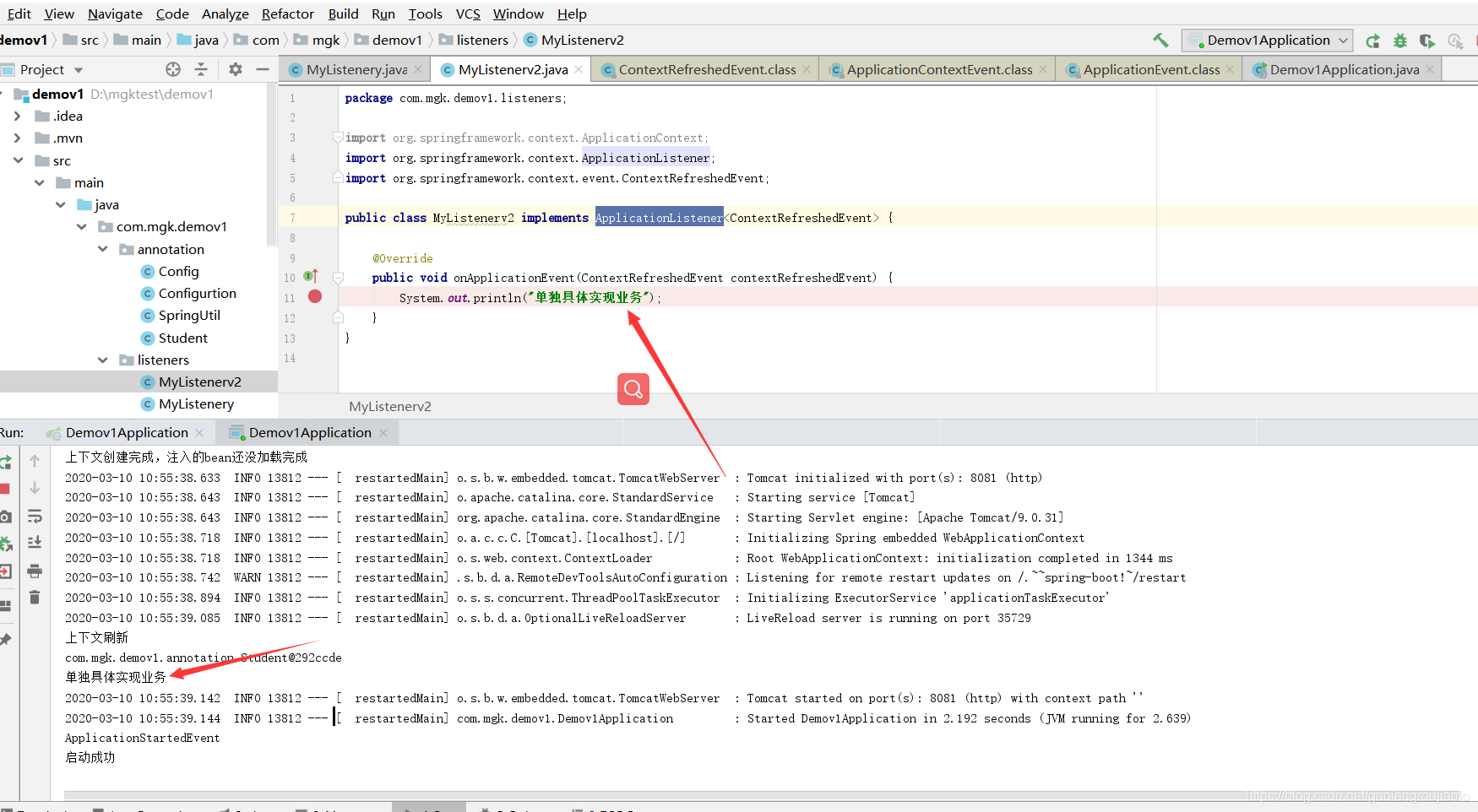
@Override

public void onApplicationEvent(ContextRefreshedEvent contextRefreshedEvent) {

System.out.println(contextRefreshedEvent);

}

}



主程序这边可以添加多个自定义监听顺序自上而下

SpringApplication app = new SpringApplication(Demov1Application.class);

app.addListeners(new MyListenery());

app.addListeners(new MyListenerv2());

app.run(args);