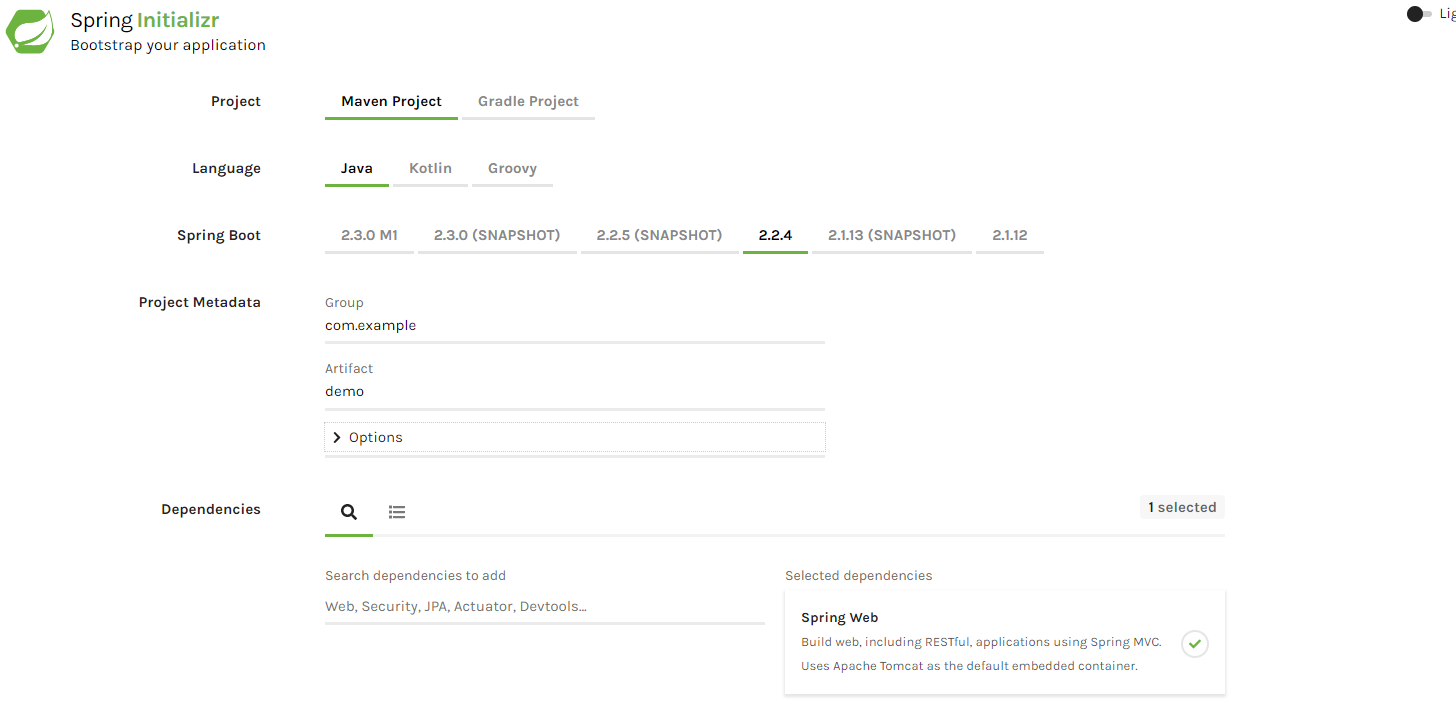
构建springboot应用:https://start.spring.io 勾选web应用



value注解拿到配置文件中的值

MyController.java

@Value(**"${myConfig.myObject.myName}"**)  
 **private** String **myName**;

**application.yml**

**myConfig**:  
 **myObject**:  
 **myName**: zhangsan  
 **myAge**: 20

更优雅的写法不是这样 而是下面这样

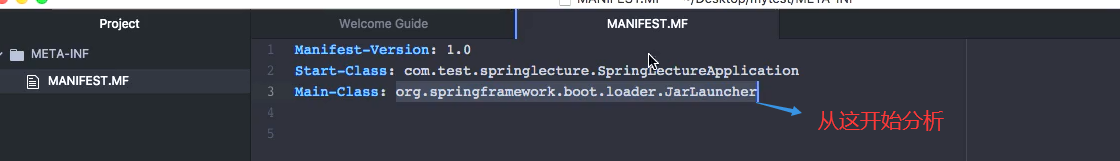
通过配置方式

|  |
| --- |
| @Configuration **public class** MyConfig {   @Bean *//这个注解必须有 代表当前对象是spring管辖的bean* **public** MyConfigBean myConfigBean(){  **return new** MyConfigBean();  }  }  @Data **public class** MyConfigBean {  @Value(**"${myConfig.myObject.myName}"**)  **private** String **myName**;  @Value(**"${myConfig.myObject.myAge}"**)  **private** Integer **myAge**;  }  具体使用的时候通过  @Autowired **private** MyConfigBean **myConfigBean**; |

研究springboot的启动过程

|  |
| --- |
| *<!--研究源码用 研究springboot的启动过程-->* <**dependency**>  <**groupId**>org.springframework.boot</**groupId**>  <**artifactId**>spring-boot-loader</**artifactId**> </**dependency**> |

从哪里分析呢？从打好的jar包Main-Class找起来



为什么从打好的jar包里面的MANIFEST.MF的Main-Class开始分析呢？因为我们的项目都是最终要打成jar包，部署到服务器上的.

JDWP远程调试 java debug wire protocol java调试协议

cmd输入java后，出来很多选项，其中有一个是用于调试的

|  |
| --- |
| -agentlib:<libname>[=<选项>]  加载本机代理库 <libname>, 例如 -agentlib:hprof  另请参阅 -agentlib:jdwp=help 和 -agentlib:hprof=help |

按照提示输入

|  |
| --- |
| C:\Users\Administrator>java -agentlib:jdwp=help  Java Debugger JDWP Agent Library  --------------------------------  (see http://java.sun.com/products/jpda for more information)  jdwp usage: java -agentlib:jdwp=[help]|[<option>=<value>, ...]  Option Name and Value Description Default  --------------------- ----------- -------  **suspend=y|n wait on startup? y**  **transport=<name> transport spec none**  **address=<listen/attach address> transport spec ""**  **server=y|n listen for debugger? n**  launch=<command line> run debugger on event none  onthrow=<exception name> debug on throw none  onuncaught=y|n debug on any uncaught? n  timeout=<timeout value> for listen/attach in milliseconds n  mutf8=y|n output modified utf-8 n  quiet=y|n |

suspend都要设置成y(默认就是y)

transport=dt\_socket是固定的

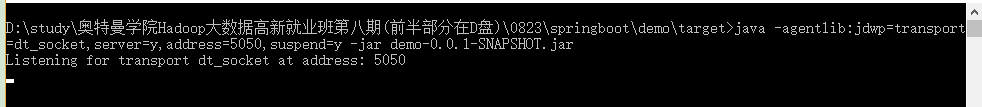
address=localhost:8080 要调试的程序的位置,如果是服务端只需要指定客户端需要连接的端口即可比address=5050,代表服务端要在5050端口进行监听。

server=y|n y代表服务端（要被监视的程序） n代表客户端(要去监视服务端的程序 默认就是)

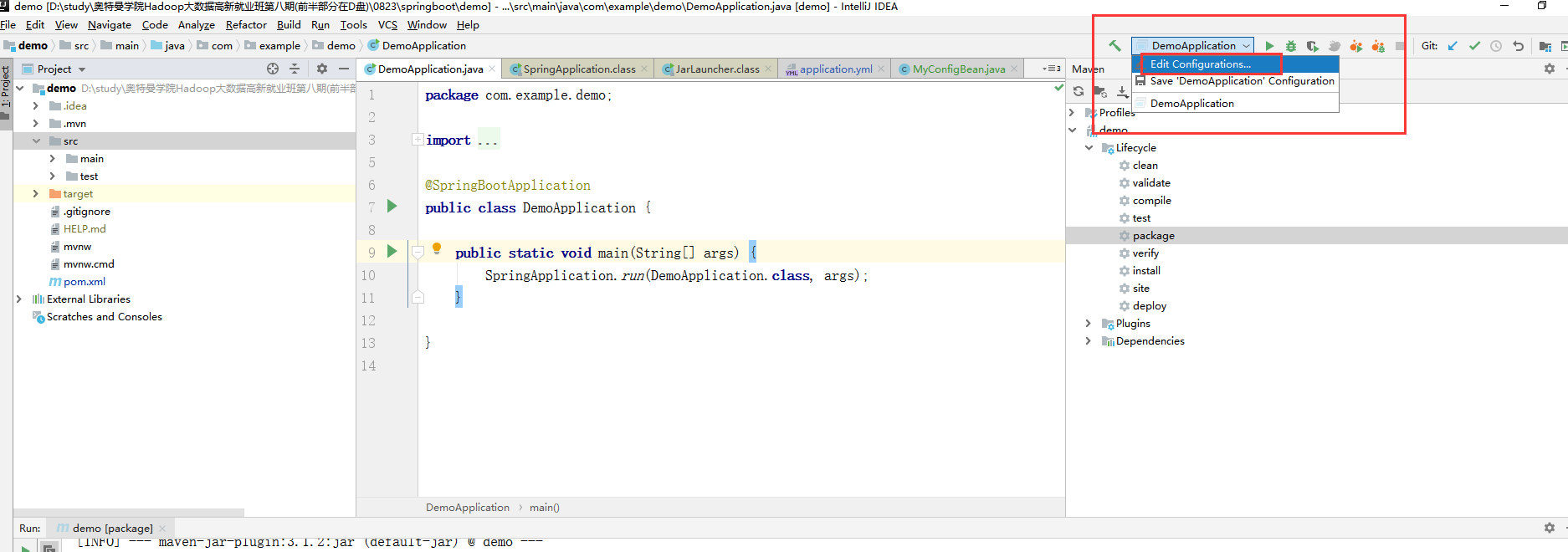
**一般情况是idea是客户端 jar包是服务端 我们是为了发现线上的jar包的问题，也就是一般说的本地没问题，上线就有问题的解决办法**

java -agentlib:jdwp=transport=dt\_socket,server=y,address=5050,suspend=y -jar test.jar

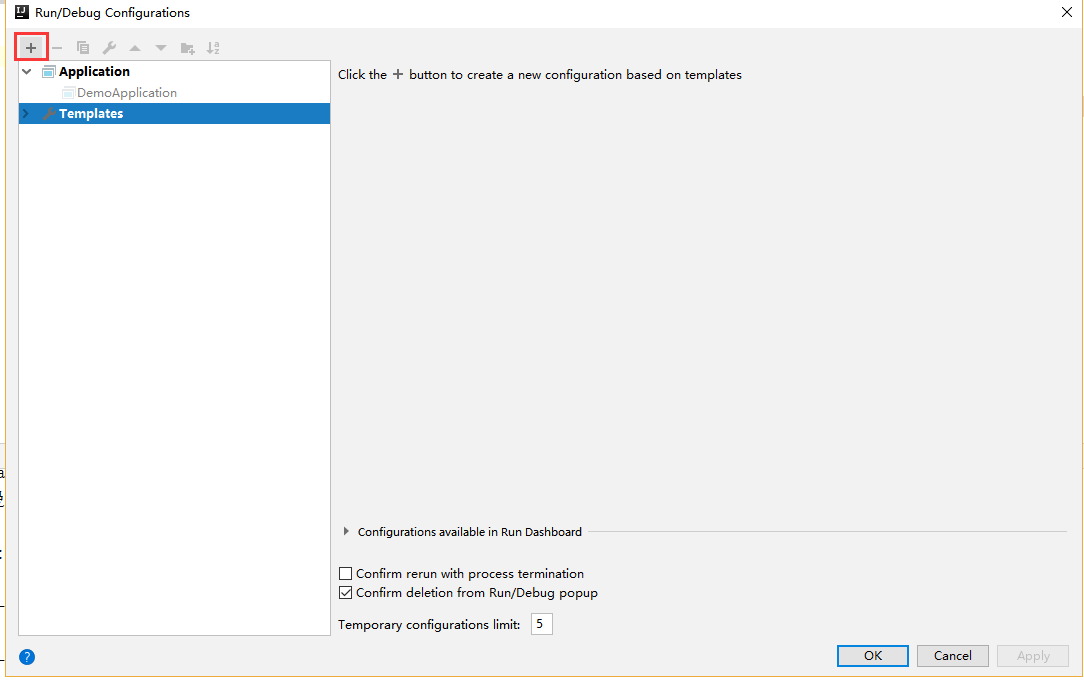
**注意执行完上面的命令后，jar包处于监听状态，不会启动起来项目**



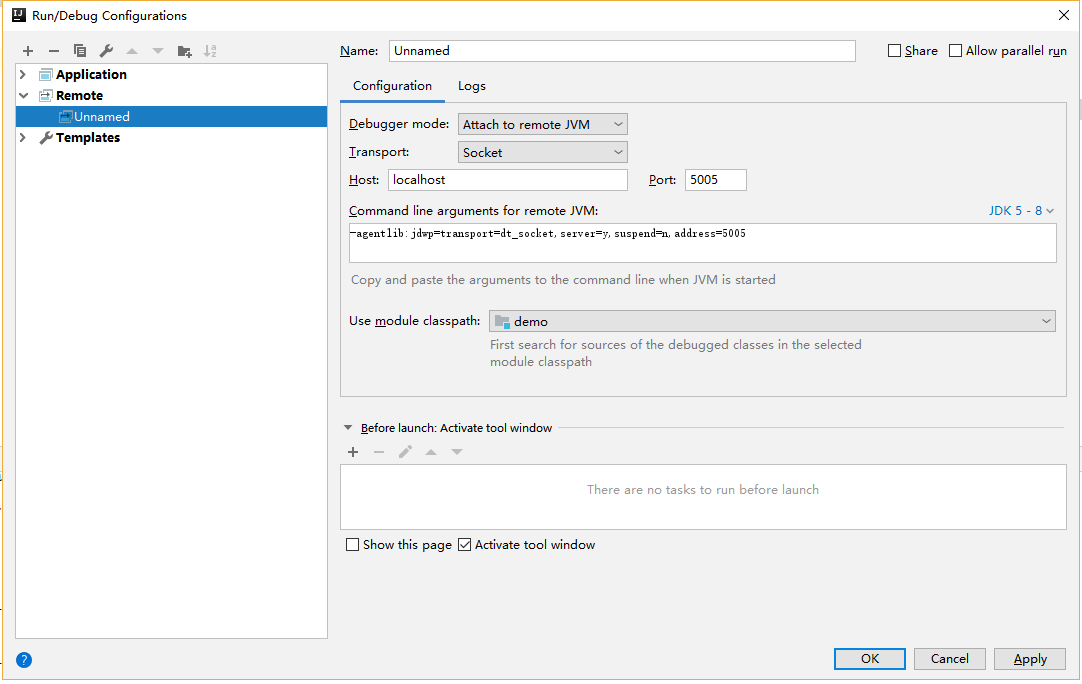
下面设置客户端

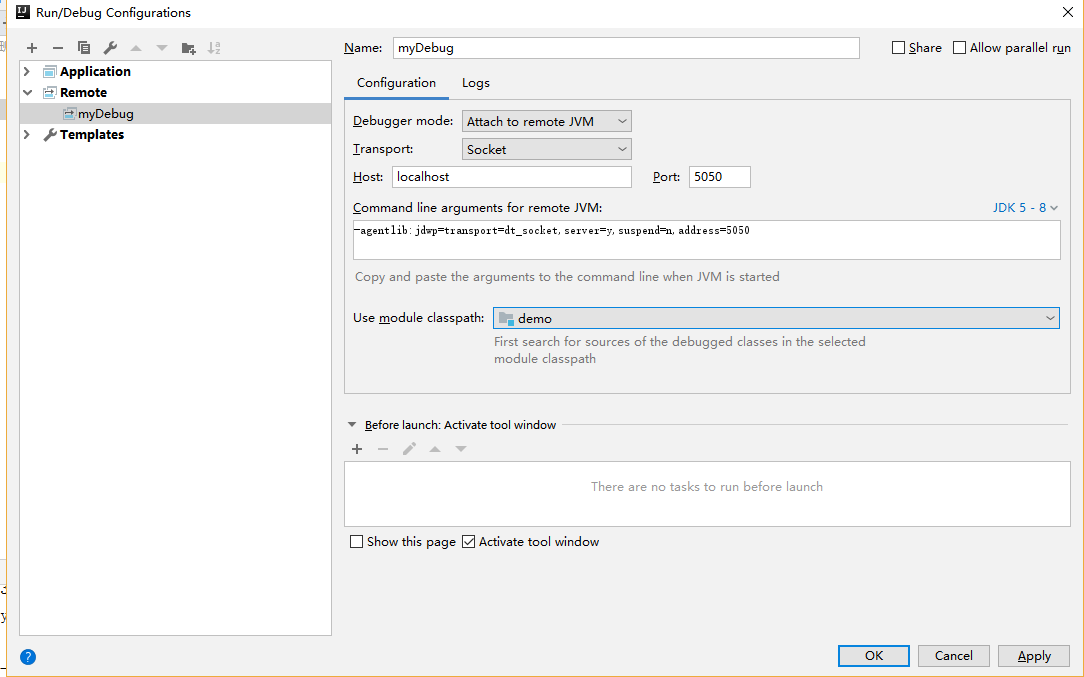


点击加号

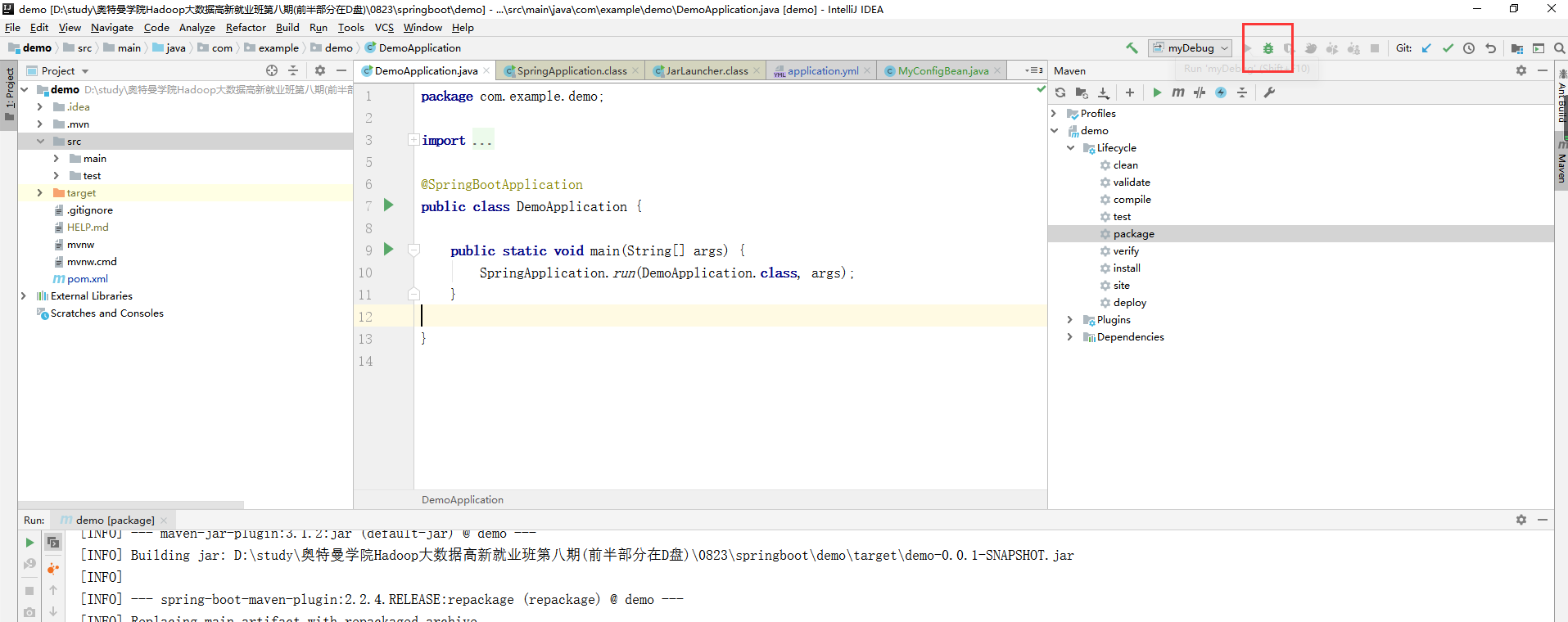


选择remote

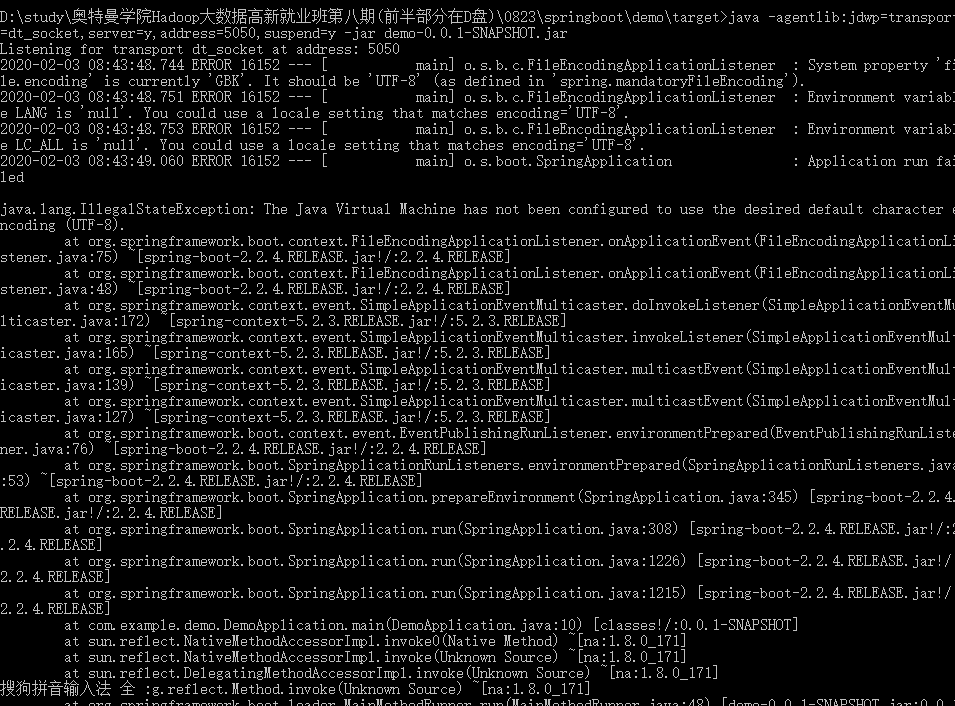




点击debug



发现服务器端也开始跑了，成功



但是跑的时候报错了，不要紧，那是另一个问题，不太重要

下一个话题 springboot日志

开发环境和测试环境的切换

|  |
| --- |
| application.yml  **spring**:  **application**:  **name**: mytest  **mandatory-file-encoding**: UTF-8  **http**:  **encoding**:  **enabled**: true  **charset**: UTF-8  **profiles**:  **active**: prod |
| application-dev.yml  **spring**:  **application**:  **name**: mytest  **mandatory-file-encoding**: UTF-8  **http**:  **encoding**:  **enabled**: true  **charset**: UTF-8 **server**:  **port**: 9090  **myConfig**:  **myObject**:  **myName**: lisi  **myAge**: 20 |
| logback-spring.xml  *<?***xml version="1.0" encoding="UTF-8"** *?>* <**configuration**>  <**include resource="org/springframework/boot/logging/logback/base.xml"**/>  *<!--会打印 debug info warn error级别日志 但是trace级别不打印 后面的具体配置会替代这个总的配置-->* <**logger name="com.example.demo" level="DEBUG"**/>  *<!--dev环境 会覆盖上面总的那个配置-->* <**springProfile name="dev"**>  <**logger name="com.example.demo" level="WARN"**/>  </**springProfile**>  <**springProfile name="prod"**>  <**logger name="com.example.demo" level="ERROR"**/>  </**springProfile**> </**configuration**> |
| 启动类DemoApplication  **package** com.example.demo;  **import** org.slf4j.Logger; **import** org.slf4j.LoggerFactory; **import** org.springframework.boot.SpringApplication; **import** org.springframework.boot.autoconfigure.SpringBootApplication;  **import** javax.annotation.PostConstruct;  @SpringBootApplication **public class** DemoApplication {  *//定义日志器* **private static final** Logger ***logger*** = LoggerFactory.*getLogger*(DemoApplication.**class**);   **public static void** main(String[] args) {  SpringApplication.*run*(DemoApplication.**class**, args);  }   @PostConstruct  **public void** myLog(){  ***logger***.trace(**"trace message"**);  ***logger***.debug(**"debug message"**);  ***logger***.info(**"info message"**);  ***logger***.warn(**"warn message"**);  ***logger***.error(**"error message"**);  }  } |
| 启动结果：  只打印error级别日志 因为application.yml配置了使用active为prod配置，这样logback-spring.xml就会找prod的日志级别. |