#### 一、热部署、热加载

更新代码后，不重启服务器编译/部署工程，不需重启服务器。

**热部署与热加载的区别**

部署方式：

热部署——在服务器运行时重新部署项目。不停服务器，直接替换工程。

热加载——在运行时重新加载class

实现原理：

热部署 ——运行时重新加载整个应用程序

热加载 ——运行时重新加载class。后台启动一个程序检查class变化，如果变化，重新加载。

使用场景：

热部署——生产环境使用

热加载——开发环境使用

**热部署原理：**

Java类加载过程

初始化jvm》产生启动类加载器》标准扩展类加载器

》系统类加载器》加载class文件

类加载的过程

加载》验证》准备》解析》初始化

加载：找到class字节码加载到jvm，形成运行时数据结构（方法区）

验证：检查字节码是否安全，保证不会对虚拟机产生威胁

准备：成员变量赋初值——初始默认值（比如：int类型的为0）

解析：符号引用变为直接引用

初始化：变量赋值，classinit

**初始化时机**

New、 getstatic、 invokstatic、 putstatic

final不会触发初始化。会把常量放入常量池

Java反射会初始化

子类调用，父类需要初始化

虚拟机启动，先初始化主类

动态语言Java.lang.invoke.method.handler

**Java类加载器特点**

1. 由AppClassLoader(系统类加载器)开始加载指定类
2. 类加载器将加载任务给其父类，如果其父类找不到，再由自己加载
3. BootstrapLoader(启动类加载器)是最顶级的类加载器

#### 二、类的热加载

继承java.lang.ClassLoader，重写findClass方法

原方法：

|  |
| --- |
| **protected** Class<?> findClass(String name) **throws** ClassNotFoundException {  **throw** **new** ClassNotFoundException(name);  } |

改写后的方法：

|  |
| --- |
| @Override  **protected** Class<?> findClass(String name) **throws** ClassNotFoundException {  System.***out***.println("加载类===" + name);  **byte**[] data = loadClassData(name);  **return** **this**.defineClass(name, data, 0, data.length);  } |

#### 三、通过tomcat配置实现热部署

1. 直接把项目web文件夹放在webapps里。

启动tomcat，将编译好的工程，copy到webapps就好了

1. 在tomcat\conf\server.xml中的<host></host>内部添加<context/>标签。

|  |
| --- |
| <Context debug=”0” docBase=”D:/Imooc/web” path=”/hot” privileged=”true” reloadable=”true”/> |

将编译好的工程 meta-inf、web-inf，copy到D:/Imooc/web就好了

1. 在%tomcat\_home%\conf\Catalina\localhost中添加一个xml——

hot.xml文件名作为webcontext，xml内容：

|  |
| --- |
| <?xml version=”1.0” encoding=”UTF-8”?>  <Context docBase=”D:/Imooc/web” reloadable=”true”/> |

#### 四、Springboot热部署

**1、Spring-boot-devtools实现热部署**

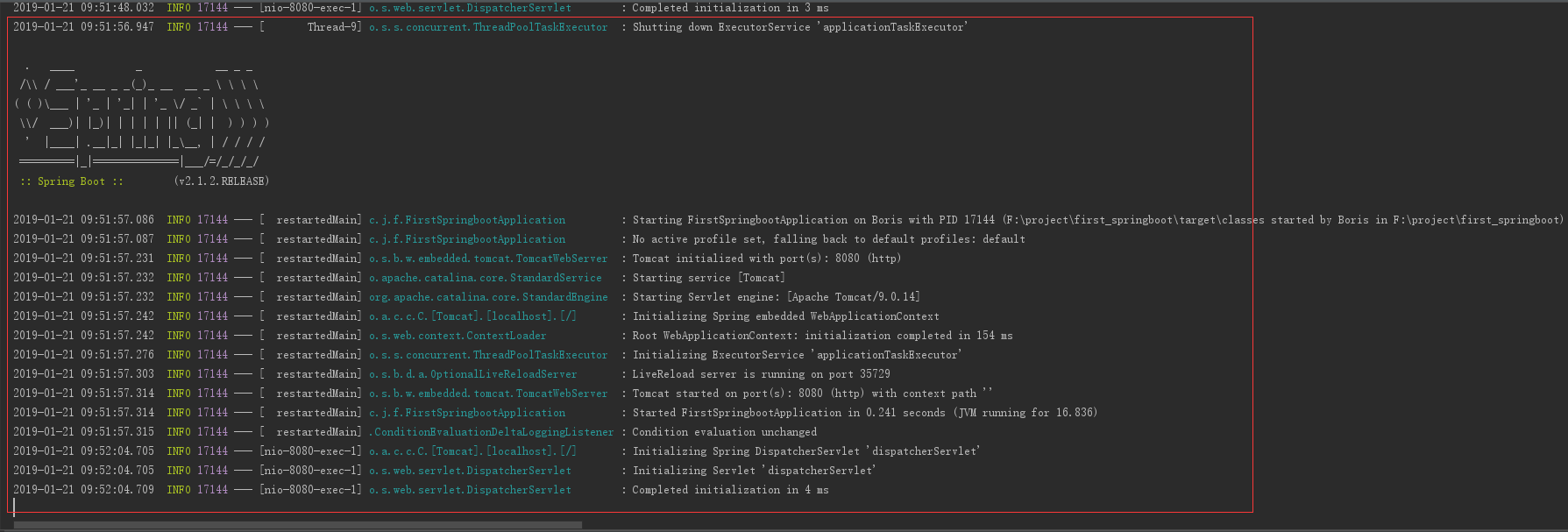
|  |
| --- |
| <!-- https://mvnrepository.com/artifact/org.springframework.boot/spring-boot-devtools -->  <dependency>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-devtools</artifactId>  <version>2.1.2.RELEASE</version>  </dependency> |

可以用maven的方式启动，maven命令：mvn spring-boot:run

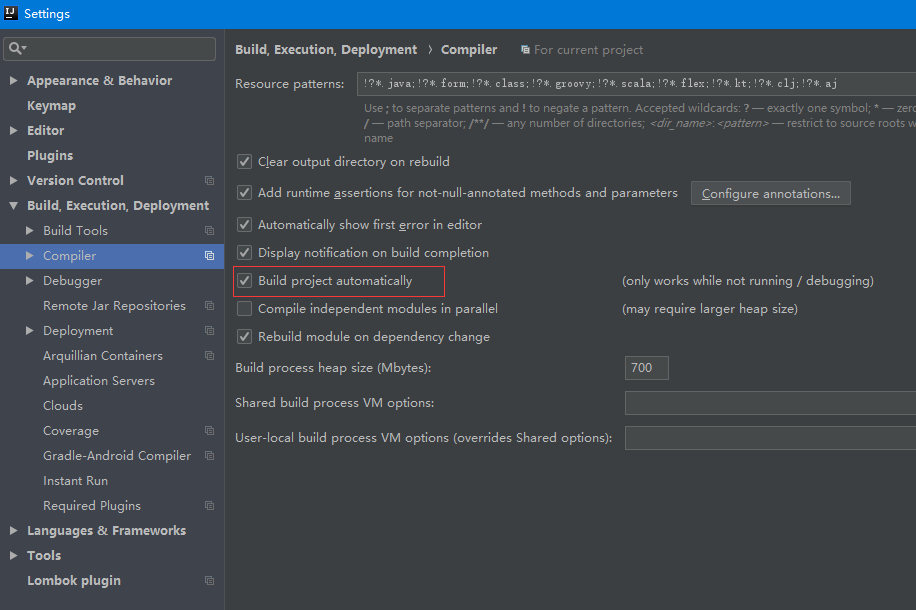
这种方式，程序自动进行了重启。

也可以用run as 或者debug as来启动spring-boot程序

spring-boot-devtools实现热部署，程序重新启动了。



使用idea开发，需要设置



**2、Springloaded方式实现热部署。**

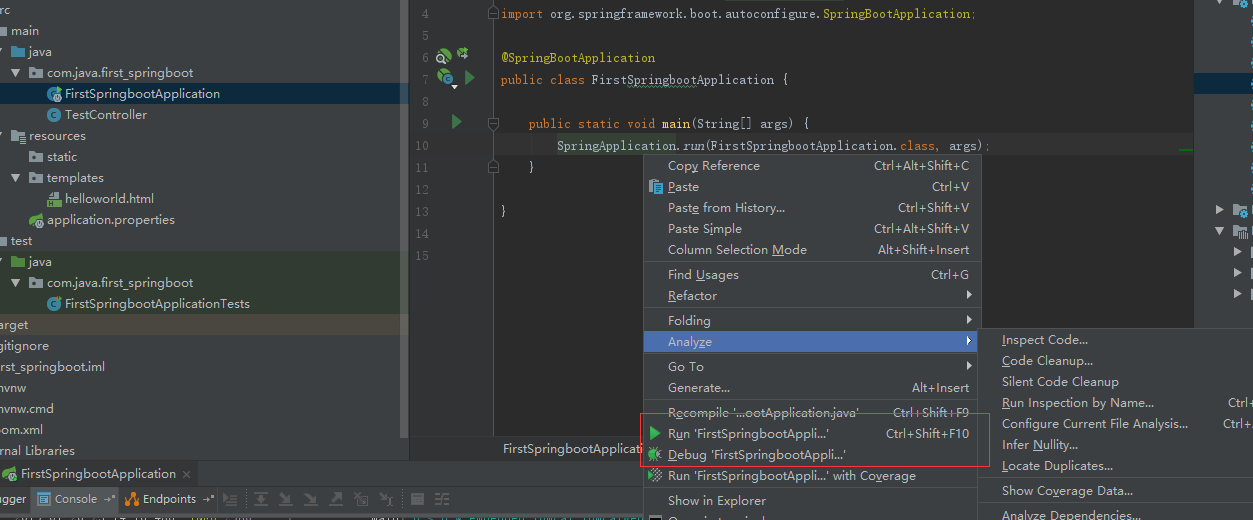
不需要添加依赖，只需要下载jar包就好了。

<dependency>  
 <groupId>org.springframework</groupId>  
 <artifactId>springloaded</artifactId>  
 <version>1.2.8.RELEASE</version>  
</dependency>

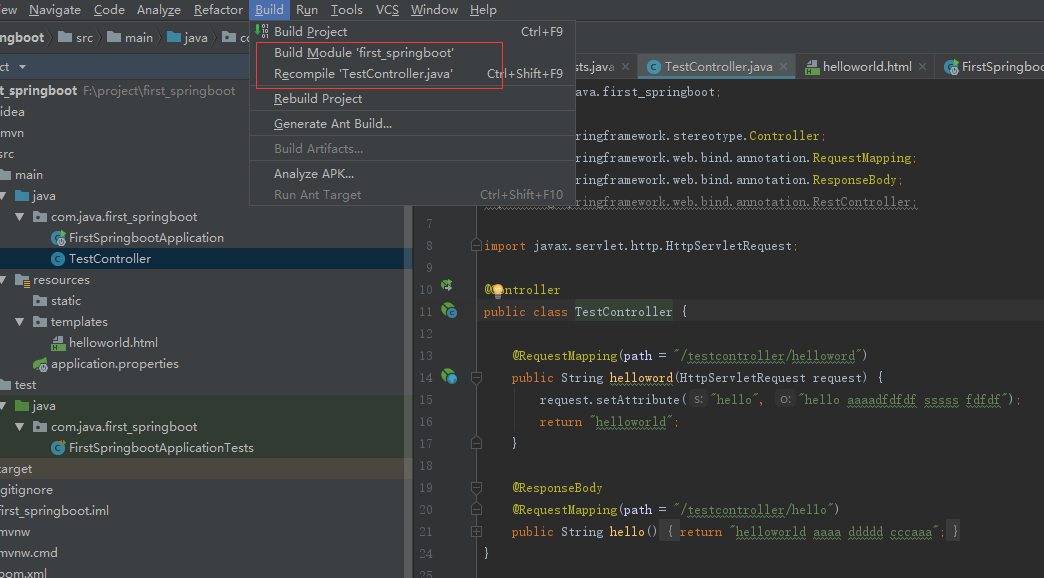
需要配置vm参数：

|  |
| --- |
| -javaagent:F:\maven-repository\org\springframework\springloaded\1.2.8.RELEASE\springloaded-1.2.8.RELEASE.jar -noverify |

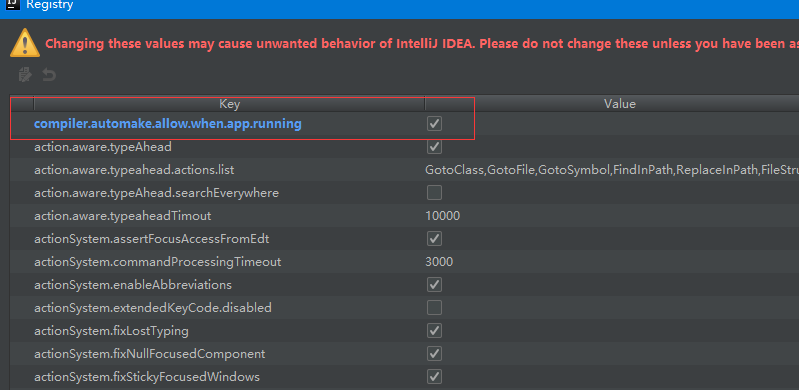
启动程序。



修改了工程文件之后，需要手动build project，或者recompile 文件



如果不想手动build和recompile，需要做如下配置：compiler.automake.allow.when.app.running



新版的idea找不到 compiler.automake.allow.when.app.running 。可以在settings中搜索registry，添加一个registry的keymap快捷键。然后用快捷键可以打开上面的界面，就可以找到这个参数，然后设置。

这种方式是重新编译编译，build了程序，没有重启服务器。

如果是纯tomcat中配置,修改启动脚本catalina.bat(linux中为catalina.sh):

#windows

#set JAVA\_OPTS=-javaagent:F:\maven-repository\org\springframework\springloaded\1.2.8.RELEASE\springloaded-1.2.8.RELEASE.jar -noverify

#linux

export JAVA\_OPTS="-javaagent:F:\maven-repository\org\springframework\springloaded\1.2.8.RELEASE\springloaded-1.2.8.RELEASE.jar -noverify"

下面这种方式，在idea中使用springloaded，然后用mvn spring-boot:run命令来启动程序实现热部署，不行。

<build>  
 <plugins>  
 <plugin>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-maven-plugin</artifactId>  
 <dependencies>  
 <!-- https://mvnrepository.com/artifact/org.springframework/springloaded -->  
 <dependency>  
 <groupId>org.springframework</groupId>  
 <artifactId>springloaded</artifactId>  
 <version>1.2.8.RELEASE</version>  
 </dependency>  
 </dependencies>  
  
 </plugin>  
 </plugins>  
</build>

#### 五、Springboot发布方式

1、构建jar，命令行运行springboot程序

修改打包方式为<packaging>jar</packaging>

用java -jar first\_springboot-0.0.1-SNAPSHOT.jar

2、构建war包，发布到tomcat

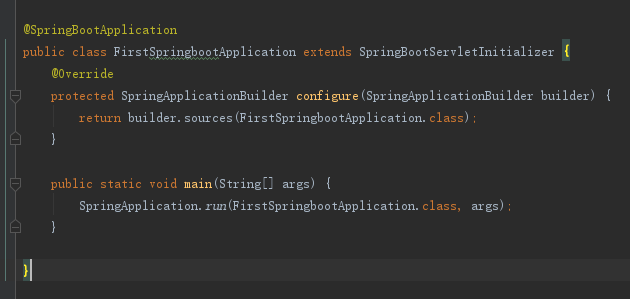
修改打包方式为：

<packaging>war</packaging>

增加tomcatstarter依赖

<dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-tomcat</artifactId>  
</dependency>

修改启动类



然后打包成war，copy到tomcat的webapp路径下面，启动tomcat，就ok了。

访问的时候，需要添加webapp-context，就是tomcat中webapp下面解压的文件夹名称。

http://localhost:8080/testcontroller/helloword

http://localhost:8080/first\_springboot-0.0.1-SNAPSHOT/testcontroller/helloword