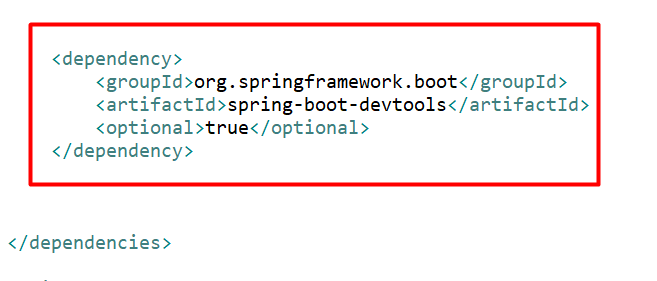
# devtools

Spring Boot中提供了一组开发工具spring-boot-devtools，可以提高开发者的工作效率，开发者可以将该模块包含在任何项目中，spring-boot-devtools最方便的地方莫过于热部署了。

## 基本使用

在项目中引入spring-boot-devtools依赖，如下：



在这里多了一个<optional>选项，是为了防止将devtools依赖传递到其他模块中。当开发者将应用打包运行后，devtools会被禁用。

开发者引入spring-boot-devtools依赖后，只要classpath路径下的文件发送了变化，项目就会自动重启，这极大的提供了开发效率。

如果开发者使用的是Eclipse，那么在修改完代码并保存后，项目将自动编译并触发重启，如果开发者使用的是IDEA，默认情况下需要手动编译才会触发重启，当然也可以配置自动重启。

## 自定义监控

默认情况下，classpath下的静态资源路径以及存放视图文件的templates目录变化后不会触发重启，如果开发者也想变化后使得项目重启，则可以在application.properties中配置，如下：



表示从默认的不触发重启的目录中去除static目录，即classpath:static目录下的资源发送变化时项目也会重启。开发者也可以反向配置，如下：



表示当classpath:static目录发生变化后会触发重启。

## 修改重启触发条件

由于编码是一个连续的过程，如果每修改一行代码就触发项目重启，这样不仅浪费电脑性能，也没有实际意义。鉴于这种情况，开发者可以使用触发文件。

在resource目录下创建一个名为.trigger-file的文件，然后在application.properties中配置触发文件，如下：



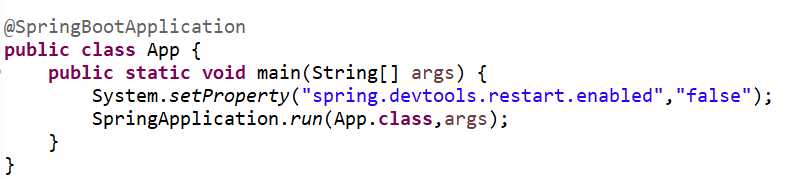
此时，当开发者修改代码时，项目将不会默认重启，如果需要项目重启，开发者只需修改.trigger-file文件触发重启。

但是需要注意的是，如果项目没有改变，只是单纯的改变了.trigger-file文件，那么项目不会重启。

## 禁用自动重启

如果开发者添加spring-boot-devtools依赖但是不想使用自动重启特性，那么可以关闭它，如下：



 或者在项目启动类中关闭它，如下：

# 单元测试

通过单元测试可以实现对每一个环节的代码进行测试，如控制层、业务层、持久层等。Spring Boot中的单元测试与Spring单元测试一脉相承，但有做了大量的简化，只需要少量的代码就能搭建一个测试环境，进而实现对Controller、Service或者Dao层的代码进行测试。

## 基本用法

## Service测试

## Controller测试

## JSON测试