Spring DevTools 介绍

> 本文地址：http://blog.csdn.net/isea533/article/details/70495714

Spring Boot包括一组额外的工具，可以使应用程序开发体验更加愉快。 `spring-boot-devtools`模块可以包含在任何项目中，它可以节省大量的时间。 想要使用devtools支持，只需将模块依赖关系添加到你的构建中：

Maven.

```
<dependencies>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-devtools</artifactId>
        <optional>true</optional>
    </dependency>
</dependencies>
```

Gradle.

```
dependencies {
    compile("org.springframework.boot:spring-boot-devtools")
}
```

> 运行打包的应用程序时，开发人员工具会自动禁用。如果你通过 `java -jar`或者其他特殊的类加载器进行启动时，都会被认为是“生产环境的应用”。
>
> 将依赖标记为`optional`可选是一种最佳做法，可以防止将devtools依赖传递到其他模块中。Gradle 不支持开箱即用的`optional`依赖项，你可以参考`propdeps-plugin`。

一、属性默认值

Spring Boot 支持的一些库中会使用缓存来提高性能。例如模版引擎将缓存编译后的模板，以避免重复解析模板文件。 此外，Spring MVC可以在服务静态资源时向响应中添加HTTP缓存头。

虽然缓存在生产中非常有益，但它在开发过程中可能会产生反效果，它会阻止你看到刚刚在应用程序中进行的更改。 因此，spring-boot-devtools将默认禁用这些缓存选项。

缓存选项通常在`application.properties`文件中配置。 例如，Thymeleaf提供了`spring.thymeleaf.cache`属性。 `spring-boot-devtools`模块不需要手动设置这些属性，而是自动应用合理的开发时配置。

二、自动重启

`spring-boot-devtools`会在类路径上的文件发生更改时自动重启。 这在IDE中工作时可能是一个有用的功能，因为它为代码更改提供了非常快的反馈循环。 默认情况下会监视类路径上的所有变动，但请注意，某些资源（如静态资源和视图模板）不需要重启应用程序。

> 触发重启

> 当DevTools监视类路径资源时，触发重启的唯一方法是更新类路径。 导致类路径更新的方式取决于你正在使用的IDE。在Eclipse中，保存修改的文件将导致类路径被更新并触发重启。 在IntelliJ IDEA中，构建项目（`Build -> Make Project`）将具有相同的效果。

> 重新启动和重新加载
>
> Spring Boot提供的重新启动技术使用了两个类加载器。 不改变的类（例如，来自第三方jar的）被加载到 *base* 类加载器中。 你正在开发的类被加载到 *restart* 类加载器中。 当应用程序*重启*时， *restart*加载器将被丢弃，并创建一个新的类加载器。 这种方法意味着应用程序重启通常比“冷启动”快得多，因为 *base* 加载器已经已加载并且可用。

1. 排除资源

某些资源在更改时不一定需要触发重启。 例如，可以直接编辑Thymeleaf模板。 默认情况下，更改`/META-INF/maven`， `/META-INF/resources`， `/resources`， `/static`， `/public`或`/templates`中的资源不会触发重启，但会触发实时重新加载。 如果要自定义这些排除项，可以使用`spring.devtools.restart.exclude`属性。 例如，要仅排除`/static`和`/public`你将设置以下内容：

```
spring.devtools.restart.exclude = static / **, public / **
```

> 如果你想保留上面的默认（情况下的）值并添加其他的排除项，你可以使用
>
> `spring.devtools.restart.additional-exclude`属性。

2. 监控额外的路径

当你对不在类路径中的文件进行更改时，可能需要重启或重新加载应用程序。 为此，请使用`spring.devtools.restart.additional-paths`属性来配置监视其他路径的更改。 你可以使用上述的`spring.devtools.restart.exclude`属性来控制附加路径下的更改是否会触发完全重启或只是实时重新加载 。

3. 禁用重启

如果不想使用重启功能，可以使用`spring.devtools.restart.enabled`属性来禁用它。 在大多数情况下，你可以在`application.properties`中设置此项（这仍将初始化重启类加载器，但不会监视文件更改）。

例如，如果你需要完全禁用重启支持，因为它不适用于特定库，则需要在调用`SpringApplication.run(…)`之前设置`System`属性。 例如：

```
public static void main(String[] args) {
    System.setProperty("spring.devtools.restart.enabled", "false");
    SpringApplication.run(MyApp.class, args);
}
```

4. 使用触发文件

如果你使用自动编译已更改文件的IDE，则可能希望仅在特定时间触发重启。 为此，你可以使用“触发文件”，这是一个特殊文件，当你要实际触发重启检查时，必须修改它。 更改文件只会触发检查，只有在Devtools检测到它必须执行某些操作时才会重启。 触发文件可以手动更新，也可以通过IDE插件更新。

要使用触发器文件，请使用`spring.devtools.restart.trigger-file`属性。

> 如果你希望将`spring.devtools.restart.trigger-file`设置为全局配置，可以参考下面第四小节。

5. 自定义重启类加载器

如上面**重新启动和重新加载**部分所述，重启功能是通过使用两个类加载器实现的。 对于大多数应用程序，此方法运行良好，但有时可能会导致类加载问题。

默认情况下，IDE中的任何打开的项目都会使用“restart”类加载器加载，任何常规`.jar`文件将使用“base”类加载器加载。 如果你在多模块项目上工作，但不是每个模块都导入到IDE中，则可能需要自定义配置。 为此，你可以创建一个`META-INF/spring-devtools.properties`文件。

`spring-devtools.properties`文件可以包含`restart.exclude.`和`restart.include.`前缀的属性。 `include`元素是应该被放入“restart”类加载器的项目， `exclude`元素是应该放入“base”类加载器的项目。 属性的值是应用于类路径下的正则表达式。

例如：

```
restart.exclude.companycommonlibs=/mycorp-common-[\\w-]+\.jar
restart.include.projectcommon=/mycorp-myproj-[\\w-]+\.jar
```

> 针对通用Mapper，可以做如下配置：
>
> `restart.include.mapper=/mapper-[\\w-\\.]+jar`

> 所有属性的键值（名字，companycommonlibs 部分）必须是唯一的，只有`restart.exclude.`和`restart.include.`开头的属性有效。

> 所有类路径下面的`META-INF/spring-devtools.properties`配置文件都会生效，所以你可以把该配置打包到每个模块中。
>
> 注：新版本的Mapper(3.4.1+)会默认增加该配置。

6. 已知限制

重启功能对使用标准`ObjectInputStream`对象序列化的对象不是很好 。如果需要反序列化数据，可能需要使用Spring的`ConfigurableObjectInputStream`配合`Thread.currentThread().getContextClassLoader()`使用。

不幸的是，一些第三方库都不考虑在使用上下文类加载器的情况下反序列化。 如果你发现这样的问题，你需要向原作者请求修复。

三、实时加载

`spring-boot-devtools`模块包含嵌入式LiveReload服务器，可以在资源更改时用于触发浏览器刷新。 LiveReload浏览器扩展程序支持Chrome，Firefox和Safari，你可以从livereload.com免费下载。

如果你不想在应用程序运行时启动LiveReload服务器，则可以将`spring.devtools.livereload.enabled`属性设置为`false` 。

> 同一时间只能运行一个LiveReload服务器。 开始应用程序之前，请确保没有其他LiveReload服务器正在运行。如果从IDE启动多个应用程序，则只有第一个应用程序将支持LiveReload。

四、全局设置

你可以通过向`$HOME`文件夹添加名为`.spring-boot-devtools.properties`的文件来配置全局devtools设置（请注意，文件名以“.”开头）。 添加到此文件的任何属性将适用于你的计算机上使用devtools的*所有* Spring Boot应用程序。 例如，要配置重启始终使用触发器文件 ，你可以添加以下内容：

~/ .spring-boot-devtools.properties。

```
spring.devtools.reload.trigger-file=.reloadtrigger
```

Spring Boot 系列

由于我博客Spring Boot 系列文章还不够多，所以暂时不打算创建专栏，如果再多几篇我就建专栏。

1. Spring Boot 入门
2. Spring Boot 属性配置和使用
3. Spring Boot 集成MyBatis
4. Spring Boot 静态资源处理
5. Spring Boot - 配置排序依赖技巧
6. Spring Boot - DevTools 介绍