<https://www.cnblogs.com/jpfss/p/9144433.html>

# Spring Boot静态资源处理

# 8.8 Spring Boot静态资源处理

当使用Spring Boot来开发一个完整的系统时，我们往往需要用到前端页面，这就不可或缺地需要访问到静态资源，比如图片、css、js等文件。

Spring Boot使用 WebMvcAutoConfiguration 中的配置各种属性, 默认为我们提供了静态资源处理。如果需要特殊处理的再通过配置进行修改。

我们来看一下WebMvcAutoConfiguration类里面的默认配置。

## 静态资源默认配置WebMvcAutoConfiguration

在WebMvcAutoConfiguration类里面有个addResourceHandlers方法：

@Override

public void addResourceHandlers(ResourceHandlerRegistry registry) {

if (!this.resourceProperties.isAddMappings()) {

logger.debug("Default resource handling disabled");

return;

}

Integer cachePeriod = this.resourceProperties.getCachePeriod();

if (!registry.hasMappingForPattern("/webjars/\*\*")) {

customizeResourceHandlerRegistration(

registry.addResourceHandler("/webjars/\*\*")

.addResourceLocations(

"classpath:/META-INF/resources/webjars/")

.setCachePeriod(cachePeriod));

}

String staticPathPattern = this.mvcProperties.getStaticPathPattern();

if (!registry.hasMappingForPattern(staticPathPattern)) {

customizeResourceHandlerRegistration(

registry.addResourceHandler(staticPathPattern)

.addResourceLocations(

this.resourceProperties.getStaticLocations())

.setCachePeriod(cachePeriod));

}

}

其中，针对webjars做了默认的处理：

registry.addResourceHandler("/webjars/\*\*")

.addResourceLocations(

"classpath:/META-INF/resources/webjars/")

其中，如果当前的ResourceHandlerRegistry里面资源映射没有 “/\*\*”， 则启用SpringBoot内置默认的静态资源处理：

String staticPathPattern = this.mvcProperties.getStaticPathPattern();//"/\*\*"

if (!registry.hasMappingForPattern(staticPathPattern)) {

customizeResourceHandlerRegistration(

registry.addResourceHandler(staticPathPattern)

.addResourceLocations(

this.resourceProperties.getStaticLocations())

.setCachePeriod(cachePeriod));

}

默认的静态资源映射路径在ResourceProperties类里面，相关代码如下：

private static final String[] CLASSPATH\_RESOURCE\_LOCATIONS = {

"classpath:/META-INF/resources/", "classpath:/resources/",

"classpath:/static/", "classpath:/public/" };

private static final String[] RESOURCE\_LOCATIONS;

static {

RESOURCE\_LOCATIONS = new String[CLASSPATH\_RESOURCE\_LOCATIONS.length

+ SERVLET\_RESOURCE\_LOCATIONS.length];

System.arraycopy(SERVLET\_RESOURCE\_LOCATIONS, 0, RESOURCE\_LOCATIONS, 0,

SERVLET\_RESOURCE\_LOCATIONS.length);

System.arraycopy(CLASSPATH\_RESOURCE\_LOCATIONS, 0, RESOURCE\_LOCATIONS,

SERVLET\_RESOURCE\_LOCATIONS.length, CLASSPATH\_RESOURCE\_LOCATIONS.length);

}

/\*\*

\* Locations of static resources. Defaults to classpath:[/META-INF/resources/,

\* /resources/, /static/, /public/] plus context:/ (the root of the servlet context).

\*/

private String[] staticLocations = RESOURCE\_LOCATIONS;

即默认配置的 /\*\* 映射到如下目录：

classpath:/META-INF/resources/  
classpath:/resources/  
classpath:/static/  
classpath:/public/

综上所述，Spring Boot 默认配置为：

| **页面请求路径模式** | **静态资源在工程的路径** | **优先级** |
| --- | --- | --- |
| /\*\* | classpath:/META-INF/resources/ | 第1优先 |
| /\*\* | classpath:/resources/ | 第2优先 |
| /\*\* | classpath:/META-INF/resources/ | 第3优先 |
| /\*\* | classpath:/static/ | 第4优先 |
| /webjars/\*\* | classpath:/META-INF/resources/webjars/ | - |

## 自定义静态资源映射

在SpringBoot集成Swagger的时候，我们需要增加swagger-ui.html映射到classpath:/META-INF/resources/，我们自定义配置类，继承WebMvcConfigurerAdapter，然后重写addResourceHandlers方法：

@Configuration

//@EnableWebMvc //这个注解会覆盖掉SpringBoot的默认的静态资源映射配置

class WebMvcConfig extends WebMvcConfigurerAdapter {

@Override

void addResourceHandlers(ResourceHandlerRegistry registry) {

//Swagger ui Mapping

registry.addResourceHandler("swagger-ui.html")

.addResourceLocations("classpath:/META-INF/resources/")

}

}

如果想要自己完全控制WebMVC（完全覆盖SpringBoot默认配置），就需要在@Configuration注解的配置类上增加@EnableWebMvc，当增加@EnableWebMvc注解以后，WebMvcAutoConfiguration中配置就不会生效，会自动覆盖了官方给出的/static, /public, META-INF/resources, /resources等存放静态资源的目录。而将静态资源定位于src/main/webapp。

在spring-boot-features.adoc中指出，如果你的应用要打成jar形式来运行的话，不要把静态资源放到src/main/webapp目录，虽然这是标准目录，但是仅在打war包的时候起作用。因为大多数的构建工具在打jar包的时候，会默认忽略此目录：

TIP: Do not use the src/main/webapp directory if your application will be packaged as a  
jar. Although this directory is a common standard, it will only work with war packaging  
and it will be silently ignored by most build tools if you generate a jar.

当需要重新定义好资源所在目录时，则需要主动添加上述的那个配置类，来Override addResourceHandlers方法。你需要自己来配置需要的每一项。

这种方式会在默认的基础上增加

swagger-ui.html映射到classpath:/META-INF/resources/

不会影响默认的方式，可以同时使用。

## 前端资源的引用方法

在index.ftl中该如何引用上面的静态资源呢？  
推荐默认的写法如下：

<link rel="stylesheet" type="text/css" href="/css/index.css">

<script type="text/javascript" src="/js/index.js"></script>

注意：默认配置的/\*\*映射到/static（或/public ，/resources，/META-INF/resources）

当请求/css/index.css的时候，Spring MVC 会在/static/目录下面找到。

如果配置为/static/js/index.js

<script src="${request.contextPath}/static/js/index.js"></script>

那么上面配置的几个目录下面都没有/static目录，因此会找不到资源文件。

这个时候，就需要另外添加自定义的映射了：

registry.addResourceHandler("/static/\*\*")

.addResourceLocations("classpath:/static/")

所以，当我们使用SpringBoot默认静态资源配置的时候，写静态资源位置不要带上映射的目录名（如/static/，/public/ ，/resources/，/META-INF/resources/）。

## 使用WebJars

Spring Boot 在支持 Spring MVC的静态资源处理的特性的同时, 允许使用jar包版本的静态资源和使用版本无关的URL的静态资源的引用。它就是Webjars[1]。

例如，使用jQuery，添加依赖：

compile group: 'org.webjars.bower', name: 'jquery', version: '3.2.1'

然后，在前端html代码中，就可以直接如下使用：

<script type="text/javascript" src="/webjars/jquery/3.2.1/jquery.js"></script>

你可能注意到href中的1.11.3版本号了，如果仅仅这么使用，那么当我们切换版本号的时候还要手动修改href，比较麻烦。我们完全可以约定一套目录规则，把后端webjars的依赖版本，直接传递到后端。而负责完成维护管理这套目录规则的人就是webjars-locator。webjars-locator通过在classpath中寻找需要加载的静态资源，然后引入前端页面。查找路径的逻辑的方法是WebJarAssetLocator类里的getFullPath方法。

我们要想使用webjars-locator，先要添加依赖：

//Group: org.webjars.bower

// https://mvnrepository.com/artifact/org.webjars/webjars-locator

compile group: 'org.webjars', name: 'webjars-locator', version: '0.32'

然后，增加一个WebJarController：

package com.easy.springboot.h5perf.controller

import org.springframework.core.io.ClassPathResource

import org.springframework.http.HttpStatus

import org.springframework.http.ResponseEntity

import org.springframework.stereotype.Controller

import org.springframework.web.bind.annotation.PathVariable

import org.springframework.web.bind.annotation.RequestMapping

import org.springframework.web.bind.annotation.ResponseBody

import org.springframework.web.servlet.HandlerMapping

import org.webjars.WebJarAssetLocator

import javax.servlet.http.HttpServletRequest

/\*\*

\* Created by jack on 2017/4/22.

\*/

@Controller

class WebJarsController {

WebJarAssetLocator assetLocator = new WebJarAssetLocator();

@ResponseBody

@RequestMapping("/webjarslocator/{webjar}/\*\*")

def ResponseEntity<?> locateWebjarAsset(@PathVariable String webjar, HttpServletRequest request) {

try {

String mvcPrefix = "/webjarslocator/" + webjar + "/"; // This prefix must match the mapping path!

String mvcPath = (String) request.getAttribute(HandlerMapping.PATH\_WITHIN\_HANDLER\_MAPPING\_ATTRIBUTE);

String fullPath = assetLocator.getFullPath(webjar, mvcPath.substring(mvcPrefix.length()));

return new ResponseEntity<>(new ClassPathResource(fullPath), HttpStatus.OK);

} catch (Exception e) {

return new ResponseEntity<>(HttpStatus.NOT\_FOUND);

}

}

}

然后，前端代码使用方式如下：

<link href="${request.contextPath}/webjarslocator/datatables/jquery.dataTables.min.css" rel="stylesheet" type="text/css">

<link href="${request.contextPath}/webjarslocator/tether/tether.min.css" rel="stylesheet" type="text/css">

<link href="${request.contextPath}/css/index.css" rel="stylesheet" type="text/css">

<link href="${request.contextPath}/css/report.css" rel="stylesheet" type="text/css">

<script src="${request.contextPath}/webjarslocator/jquery/jquery.min.js"></script>

<script src="${request.contextPath}/js/index.js"></script>

<script src="${request.contextPath}/webjarslocator/tether/tether.min.js"></script>

<script src="${request.contextPath}/webjarslocator/bootstrap/bootstrap.min.js"></script>

<script src="${request.contextPath}/webjarslocator/datatables/jquery.dataTables.min.js"></script>

这里是freemarker的模板view代码示例。其中，request对象是内置对象。在application.yml配置如下：

spring:

# 在freemarker获取request对象

freemarker:

request-context-attribute: request

注意：这里不需要在写版本号了，但是注意写url的时候，只是在原来url基础上去掉了版本号，其他的都不能少！

## 静态资源动态版本

当我们资源内容发生变化时，由于浏览器缓存，用户本地的静态资源还是旧的资源，为了防止这种情况导致的问题，我们在请求url的时候加个版本号。

版本号如：

<script type="text/javascript" src="/js/index.js?v=1.0.1"></script>

这个版本号1.0.1，可以由后端代码传到前端页面${version}。

# 小结

本章节主要探讨了Spring Boot 静态资源处理的内容。当我们在开发中，遵循SpringBoot的默认配置，可以大大减少了我们静态资源处理的工作。

本章节完整的工程代码：

[https://github.com/EasySpringBoot/h5perf](https://link.jianshu.com/?t=https://github.com/EasySpringBoot/h5perf)

参考资料：

1.WebJars：[http://www.webjars.org/](https://link.jianshu.com/?t=http://www.webjars.org/)