横云断岭的专栏

深入Spring Boot：显式配置 @EnableWebMvc 导致静态资源访问失败

2018-08-08 | 2020-03-02 | 技术

现象

当用户在自己的spring boot main class上面显式使用了 `@EnableWebMvc`，发现原来的放在 `src/main/resources/static` 目录下面的静态资源访问不到了。

```
@SpringBootApplication
@EnableWebMvc
public class DemoApplication {
        public static void main(String[] args) {
                SpringApplication.run(DemoApplication.class, args);
        }
}
```

比如在用户代码目录 `src/main/resources` 里有一个 `hello.txt` 的资源。访问 `http://localhost:8080/hello.txt` 返回的结果是404：

```
Whitelabel Error Page

This application has no explicit mapping for /error, so you are seeing this as a fal

Thu Jun 01 11:39:41 CST 2017
There was an unexpected error (type=Not Found, status=404).
No message available
```

静态资源访问失败原因

`@EnableWebMvc` 的实现

那么为什么用户显式配置了 `@EnableWebMvc`，spring boot访问静态资源会失败？

我们先来看下 `@EnableWebMvc` 的实现：

```java
@Import(DelegatingWebMvcConfiguration.class)
public @interface EnableWebMvc {
}
```

```java
/**
 * A subclass of {@code WebMvcConfigurationSupport} that detects and delegates
 * to all beans of type {@link WebMvcConfigurer} allowing them to customize the
 * configuration provided by {@code WebMvcConfigurationSupport}. This is the
 * class actually imported by {@link EnableWebMvc @EnableWebMvc}.
 *
 * @author Rossen Stoyanchev
 * @since 3.1
 */
@Configuration
public class DelegatingWebMvcConfiguration extends WebMvcConfigurationSupport {
```

可以看到`@EnableWebMvc` 引入了 `WebMvcConfigurationSupport`，是spring mvc 3.1里引入的一个自动初始化配置的`@Configuration` 类。

spring boot里的静态资源访问的实现

再来看下spring boot里是怎么实现对 `src/main/resources/static` 这些目录的支持。

主要是通过 `org.springframework.boot.autoconfigure.web.WebMvcAutoConfiguration` 来实现的。

```java
@Configuration
@ConditionalOnWebApplication
@ConditionalOnClass({ Servlet.class, DispatcherServlet.class,
                WebMvcConfigurerAdapter.class })
@ConditionalOnMissingBean(WebMvcConfigurationSupport.class)
@AutoConfigureOrder(Ordered.HIGHEST_PRECEDENCE + 10)
@AutoConfigureAfter({ DispatcherServletAutoConfiguration.class,
                ValidationAutoConfiguration.class })
public class WebMvcAutoConfiguration {
```

可以看到 `@ConditionalOnMissingBean(WebMvcConfigurationSupport.class)`，这个条件导致spring boot的 `WebMvcAutoConfiguration` 不生效。

总结下具体的原因：

0. 用户配置了 `@EnableWebMvc`
1. Spring扫描所有的注解，再从注解上扫描到 `@Import` ，把这些 `@Import` 引入的bean信息都缓存起来
2. 在扫描到 `@EnableWebMvc` 时，通过 `@Import` 加入了 `DelegatingWebMvcConfiguration` ，也就是 `WebMvcConfigurationSupport`
3. spring再处理 `@Conditional` 相关的注解，判断发现已有 `WebMvcConfigurationSupport` ，就跳过了spring bootr的 `WebMvcAutoConfiguration`

所以spring boot自己的静态资源配置不生效。

其实在spring boot的文档里也有提到这点： http://docs.spring.io/spring-boot/docs/current/reference/htmlsingle/#boot-features-spring-mvc-auto-configuration

- If you want to keep Spring Boot MVC features, and you just want to add additional MVC configuration (interceptors, formatters, view controllers etc.) you can add your own @Configuration class of type WebMvcConfigurerAdapter, but without @EnableWebMvc. If you wish to provide custom instances of RequestMappingHandlerMapping, RequestMappingHandlerAdapter or ExceptionHandlerExceptionResolver you can declare a WebMvcRegistrationsAdapter instance providing such components.

Spring Boot ResourceProperties的配置

在spring boot里静态资源目录的配置是在 `ResourceProperties` 里。

```
@ConfigurationProperties(prefix = "spring.resources", ignoreUnknownFields = false)
public class ResourceProperties implements ResourceLoaderAware {

        private static final String[] SERVLET_RESOURCE_LOCATIONS = { "/" };

        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.lengtl
                                + SERVLET_RESOURCE_LOCATIONS.length];
                System.arraycopy(SERVLET_RESOURCE_LOCATIONS, 0, RESOURCE_LOCATIONS
                                SERVLET_RESOURCE_LOCATIONS.length);
                System.arraycopy(CLASSPATH_RESOURCE_LOCATIONS, 0, RESOURCE_LOCATIOI
                                SERVLET_RESOURCE_LOCATIONS.length, CLASSPATH_RESOUI
        }
```

然后在 `WebMvcAutoConfigurationAdapter` 里会初始始化相关的ResourceHandler。

```java
//org.springframework.boot.autoconfigure.web.WebMvcAutoConfiguration.WebMvcAutoCon
@Configuration
@Import({ EnableWebMvcConfiguration.class, MvcValidatorRegistrar.class })
@EnableConfigurationProperties({ WebMvcProperties.class, ResourceProperties.class 
public static class WebMvcAutoConfigurationAdapter extends WebMvcConfigurerAdapter

  private static final Log logger = LogFactory
      .getLog(WebMvcConfigurerAdapter.class);

  private final ResourceProperties resourceProperties;

  @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));
    }
  }
```

用户可以自己修改这个默认的静态资源目录，但是不建议，因为很容易引出奇怪的404问题。

欢迎您扫一扫上面的微信公众号，订阅横云断岭的专栏

spring-boot # spring # web

< 深入Spring Boot：编写兼容Spring Boot1和Spring Boot2的Starter

深入Spring Boot：利用Arthas排查NoSuchMethodError >

Related Issues not found

Please contact @hengyunabc to initialize the comment

Login with GitHub