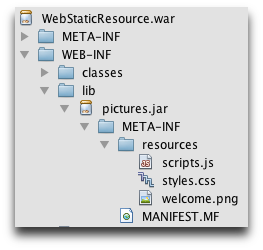
Before servlet 3.0, images could be made accessible from the root of the web applications but that meant copying the files to the WAR archive and keeping them up-to-date. This certainly meant a solution tightly coupled with the web application development and packaging. The other option was to place this static content in the docroot of the application server which was in turn time probably too loosely coupled allowing for anyone to access this and encouraging every application to use the same set of static content.

With Servlet 3.0, a JAR placed in WEB-INF/lib has static content from its META-INF/resource directory accessible from the web-context root. You can also parse this previous statement with **WEB-INF/lib/{\\*.jar}/META-INF/resources**. So this means you no longer need to use the ServletContext getResource() and getResourceAsStream() methods with some rather dumb rewriting.

In this simple web application WAR example :



... the static resources are available from :

http://host:port/*webcontext*/scripts.js

http://host:port/*webcontext*/styles.css

http://host:port/*webcontext*/welcome.png

where http://host:port/*webcontext*/ could be replaced with the relative path "./"

This makes for more modular applications. Other than images, think of how this applies to CSS and javascript. It's probably now a good idea to package JavaScript libraries such a jquery or dojo in a dedicated JAR (effectively a resource JAR).

The other use-case I can think of is configuration files. One could deploy with WEB-INF/lib/testing.jar or with WEB-INF/lib/production.jar each of which containing META-INF/resources/config.properties file with different content. The application code reading the configuration would always access it using *./config.properties* (or http://host:port/*webcontext*/config.properties).

Note this mechanism also applies to JSPs and that resource files placed in the document root take precedence. Get all the details from paragraph 10.5 of the [Servlet 3.0 specification](http://jcp.org/en/jsr/detail?id=315).

<http://www.jianshu.com/p/23bea93e34c7>

**静态资源加载**

我们在开发WEB页面的时候，还需要加载很多静态资源，比如js、图片、css等文件，那这些文件应该放在哪里呢？

我们在启动Spring Boot的时候，可以看到控制台输出如下一些信息

2016-09-20 15:18:55.445 INFO 16210 --- [ main] o.s.w.s.handler.SimpleUrlHandlerMapping : Mapped URL path [/webjars/\*\*] onto handler of type [class org.springframework.web.servlet.resource.ResourceHttpRequestHandler]
2016-09-20 15:18:55.445 INFO 16210 --- [ main] o.s.w.s.handler.SimpleUrlHandlerMapping : Mapped URL path [/\*\*] onto handler of type [class org.springframework.web.servlet.resource.ResourceHttpRequestHandler]
2016-09-20 15:18:55.474 INFO 16210 --- [ main] o.s.w.s.handler.SimpleUrlHandlerMapping : Mapped URL path [/\*\*/favicon.ico] onto handler of type [class org.springframework.web.servlet.resource.ResourceHttpRequestHandler]

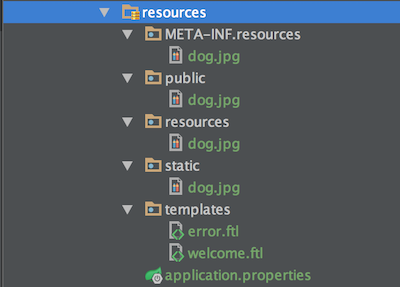
其中默认配置的 / **映射到 /static （或/public、/resources、/META-INF/resources）**

**其中默认配置的 /webjars/** 映射到 classpath:/META-INF/resources/webjars/

PS：上面的 static、public、resources 等目录都在 classpath: 下面（例如 src/main/resources/static）。

假如我们的文件是如下图放置，当我们输入[http://localhost:8080/dog.jpg的时候，显示的是哪张图片呢？大家可以自行验证一下，这边先告诉大家Spring](http://localhost:8080/dog.jpg%E7%9A%84%E6%97%B6%E5%80%99%EF%BC%8C%E6%98%BE%E7%A4%BA%E7%9A%84%E6%98%AF%E5%93%AA%E5%BC%A0%E5%9B%BE%E7%89%87%E5%91%A2%EF%BC%9F%E5%A4%A7%E5%AE%B6%E5%8F%AF%E4%BB%A5%E8%87%AA%E8%A1%8C%E9%AA%8C%E8%AF%81%E4%B8%80%E4%B8%8B%EF%BC%8C%E8%BF%99%E8%BE%B9%E5%85%88%E5%91%8A%E8%AF%89%E5%A4%A7%E5%AE%B6Spring) Boot的加载顺序是

META-INF/resources > resources > static > public



静态资源加载顺序

文／蓝色的咖啡（简书作者）

原文链接：http://www.jianshu.com/p/23bea93e34c7

著作权归作者所有，转载请联系作者获得授权，并标注“简书作者”。

**默认配置**

Spring Boot默认提供静态资源目录位置需置于classpath下，目录名需符合如下规则：

/static

/public

/resources

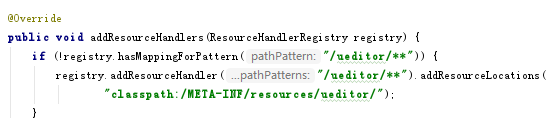
/META-INF/resources

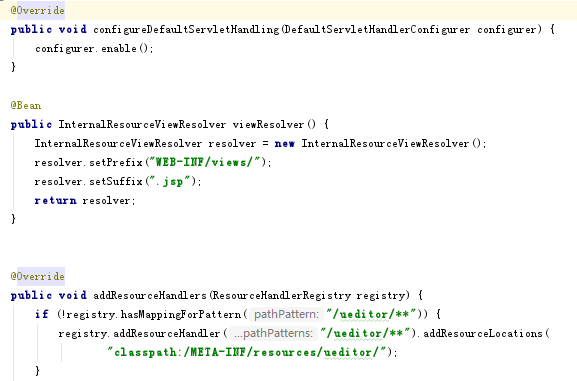
举例：我们可以在src/main/resources/目录下创建static，在该位置放置一个图片文件。启动程序后，尝试访问http://localhost:8080/D.jpg。如能显示图片，配置成功。



以上图的结构，在其他地方使用时要加上如下代码：

extends WebMvcConfigurerAdapter





<**mvc:resources location="classpath:/META-INF/resources/" mapping="/kindeditor/"**/>

在jsp中如下使用：



如果用servlet3.0写一些servlet时，用在spring boot时需要注册servlet,可以使用@ServletComponentScan扫描

[**Spring Boot使用Servlet、Filter或Listener的方式**](http://www.cnblogs.com/larryzeal/p/7576875.html)

根据官方文档说明，有两种方式可以在你的Spring Boot应用中使用Servlet、Filter或Listener。

其一：将Servlet、Filter或Listener注册成Spring Bean。

注意：

由于这三者与容器有关，它们不会被延迟初始化！所以，一般情况下不能将其他Spring Bean 注入到这三者的属性中 -- 但不是说不能获取，你完全可以使用某些手段，在调用之前再注入！

Servlet、Filter可以使用ServletRegistrationBean、FilterRegistrationBean来设置初始化参数和映射路径。

默认，如果不指定某个filter的dispatcherType的话，会匹配FORWARD、INCLUDE和REQUEST。如果启用了async，那还会匹配ASYNC。

如果只想使用Bean，而不想在容器中安装它们的话，registration.setEnabled(false) 即可。

其二：使用类路径扫描，具体来说就是在@Configuration class 上使用@ServletComponentScan。

注意：

需要配合@WebServlet、@WebFilter或@WebListener使用； //这是Servlet 3的东西，javax.servlet。

默认扫描当前配置文件所在包及子包。

ps：

如果是Spring MVC项目，且是Java config的，那可以在选择上面的第一种方式；

或者在WebApplicationInitializer实现类中设置Filter - 应该是有一个添加多个Filter的方法，但只能用于dispatcherServlet！

或者，还可以在WebApplicationInitializer实现类的onStartup()方法中获取ServletContext，并注册自己的Servlet、Filter或Listener。

WebApplicationInitializer 这个东西很有意思，属于SPI，但又不用自己设置加载的文件，所以应该是由Spring框架负责提供给ServletContainerInitializer的吧。

还需要补充一点，这是Servlet 3.0+才支持的功能。