1. 创建springBoot项目之后，会在根目录下会有一个artifactId+Application命名规则的入口类。如果使用了@SpringBootApplication注解的话，系统会去入口类的同级包以及下级包去扫描实体类。因此一般讲入口类的位置放在groupId+arctifactID组合的包名下。@SpringBootApplication(exclude =DataSourceAutoConfiguration.class)也可以这样配置扫描特定的类。
2. 修改banner：启动时如果想改变启动时的图案，在resources下添加banner.txt，复制字符图案进去即可。

关闭banner:

public static void main(String[] args) {

SpringApplicationBuilder builder = new SpringApplicationBuilder(Test19SpringBoot2Application.class);

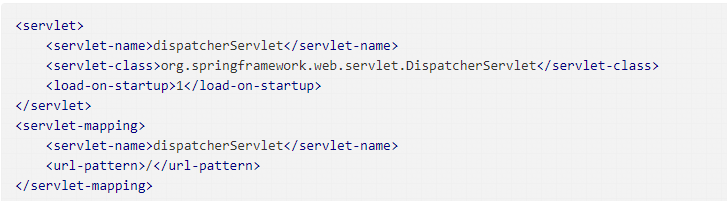
//修改Banner的模式为OFF

builder.bannerMode(Banner.Mode.OFF).run(args);

}

1. Web配置-自动配置viewResolver

DispatcherServlet：在springmvc中需要配置DispatcherServlet,一般在web.xml中配置



但是在springboot中不需手动配置，springboot的autoconfigure（@EnableAutoConfigure）会默认配置成这样，

ViewResolver:在springmvc中，视图处理的配置一般在springmvc的xml配置中添加：



1. springboot静态资源配置：Spring Boot 默认为我们提供了静态资源处理，使用 中WebMvcAutoConfiguration 的配置各种属性。提供的静态资源映射如下:

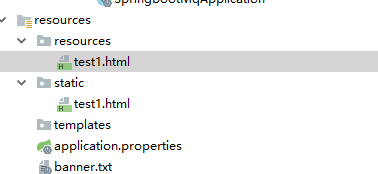
classpath:/META-INF/resources

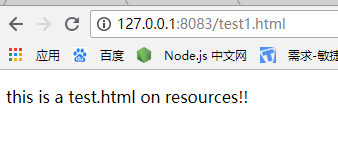
classpath:/resources

classpath:/static

classpath:/public

静态资源的映射路径，优先级顺序为：META-INF/resources > resources > static > public





在resources下有两个默认的path:resources、static,这两个文件夹下都有相同名字的文件，在页面进行默认path访问的时候会先去访问有限级高的默认path,即resources.

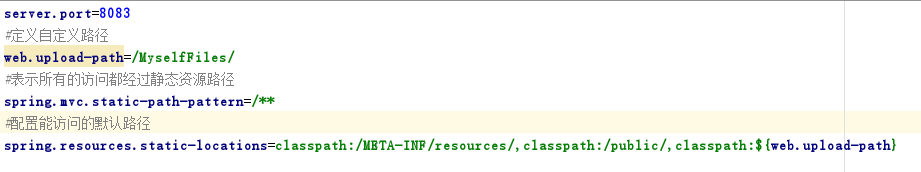
\*\*修改默认path:

可修改propeties或者java代码覆盖。

使用java代码覆盖：重写addRourceHandler方法。

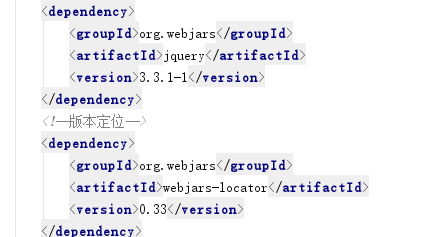


使用配置：

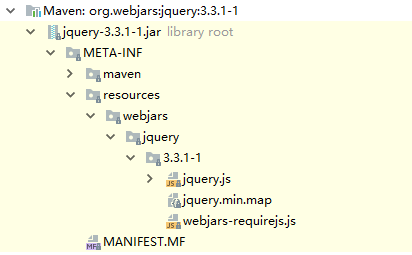


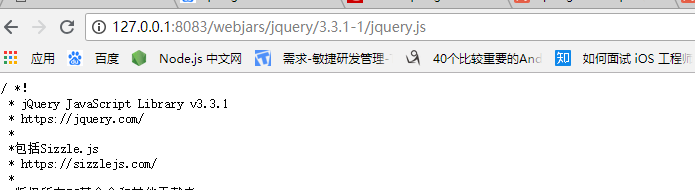
配置了自定义访问路径之后就能访问到自定义的路径下的静态文件。未配置的将不能访问到；

\*\*\*采用webjars，即通过jar包引入静态资源；WebJars是将Web前端Javascript和CSS等资源打包成Java的Jar包，这样在Java Web开发中我们可以借助Maven这些依赖库的管理，保证这些Web资源版本唯一性。



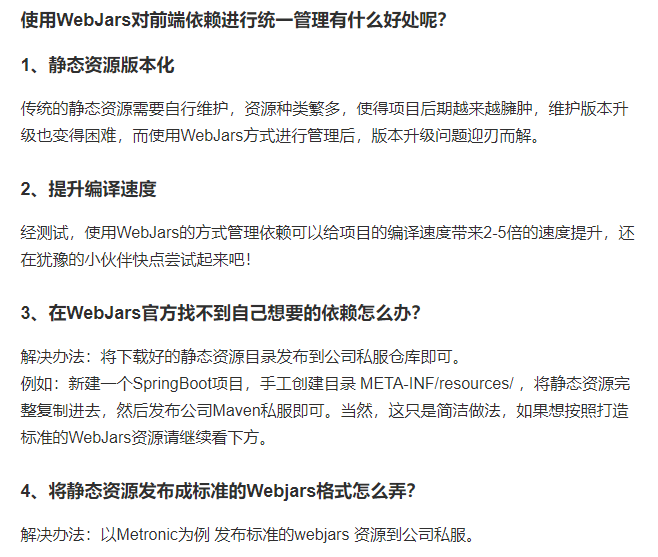
默认会到 classpath:/META-INF/resources/webjars/下去找资源。





作用：加入了webjars-locator,会自动定位版本号，比如下面的测试页面，引用jquery的时候可以不写版本号也可以直接访问，这样，如果版本号发生了改变，只需要修改引入webjars包的版本，不用修改html引用页面。





1. 自动配置formatter和converter:

Converter可以从任意源类型，转换为任意目标类型。而Formatter则是从String类型转换为任务目标类型，有点类似PropertyEditor。可以感觉出Converter是Formater的超集，实际上在Spring中Formatter是被拆解成PrinterConverter和ParserConverter，然后再注册到ConverterRegistry，供后续使用。

应用：（项目也采用该方法）

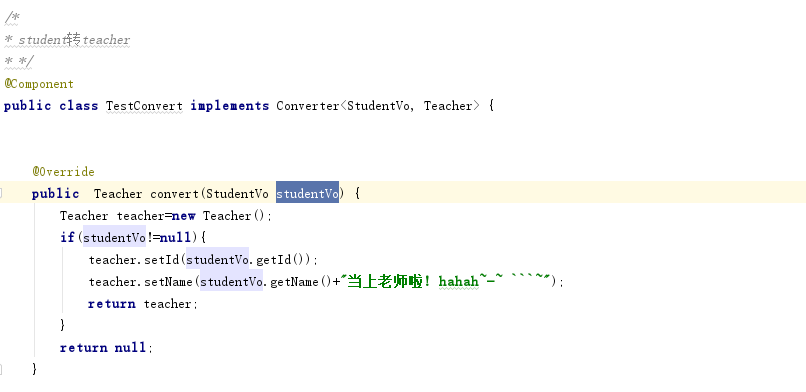


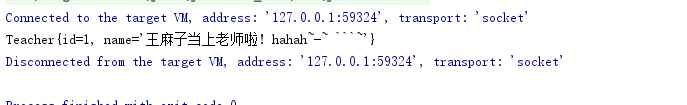
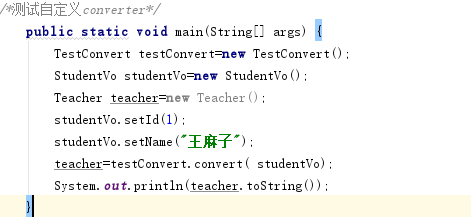




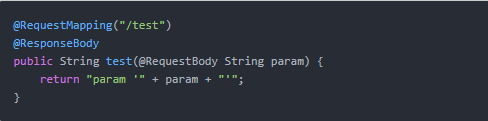
测试方法将对象转为String。

\*\*自定义converter/formatter:

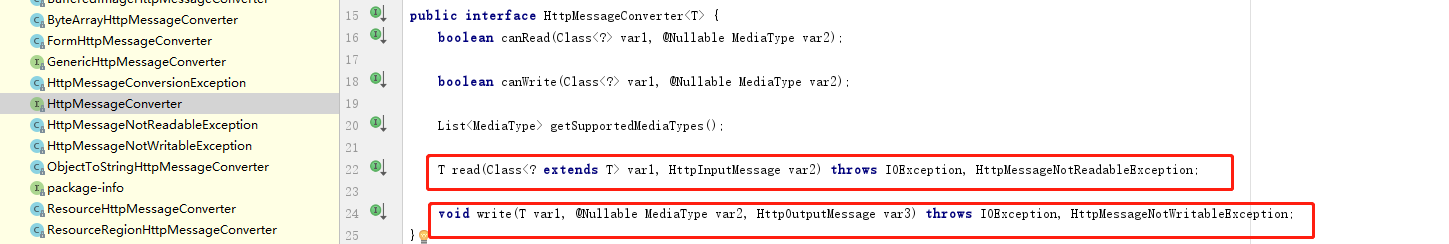


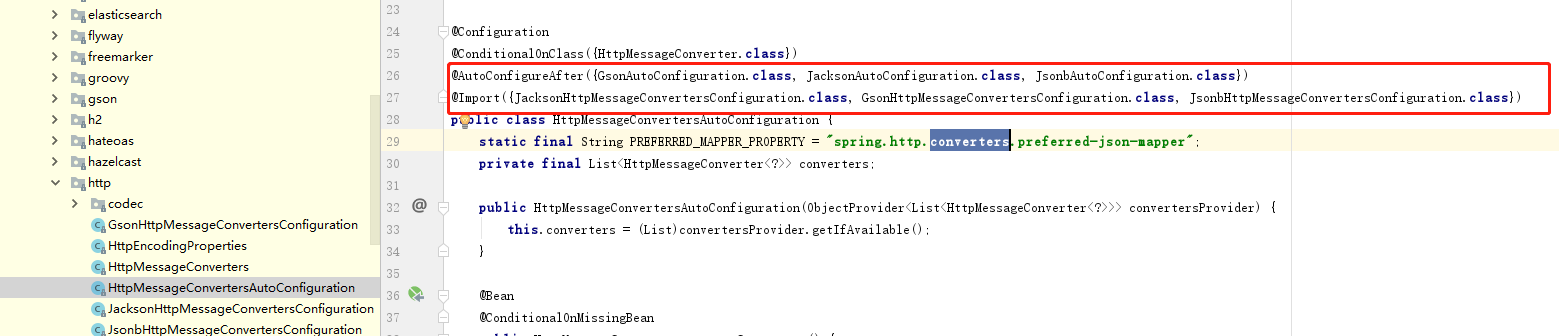


1. 自动配置HttpMessageConverter:

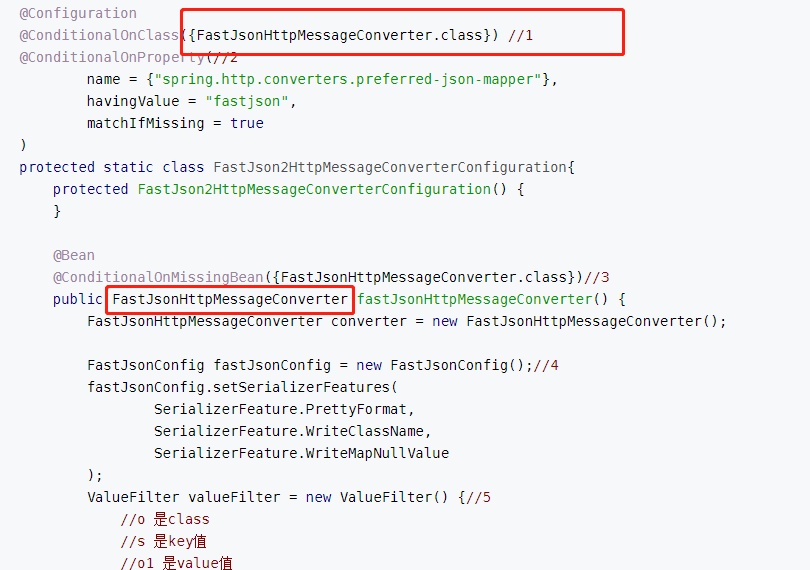


请求进入test方法前，会根据@RequestBody注解选择对应的HttpMessageConverter实现类来将请求参数解析到param变量中，因为这里的参数是String类型的，所以这里是使用了StringHttpMessageConverter类，它的canRead()方法返回true，然后read()方法会从请求中读出请求参数，绑定到test()方法的param变量中。同理当执行test方法后，由于返回值标识了@ResponseBody，SpringMVC / SpringBoot将使用StringHttpMessageConverter的write()方法，将结果作为String值写入响应报文，当然，此时canWrite()方法返回true。

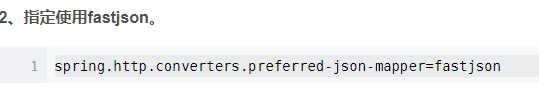


httpmessageconvert在自动配置时就会自动初始化gson(也是用来转换为json)和jackson，默认使用用jackson，找到了配置名为：spring.http.converters.preferred-json-mapper,使用该配置名可以配置使用哪种转换方式。  
  


\*\*自定义HttpMessageConvert，采用配置方式，添加fastJson转换方式，然后使用



在配置文件中使用spring.http.converters.preferred-json-mapper



1. 注册servlet

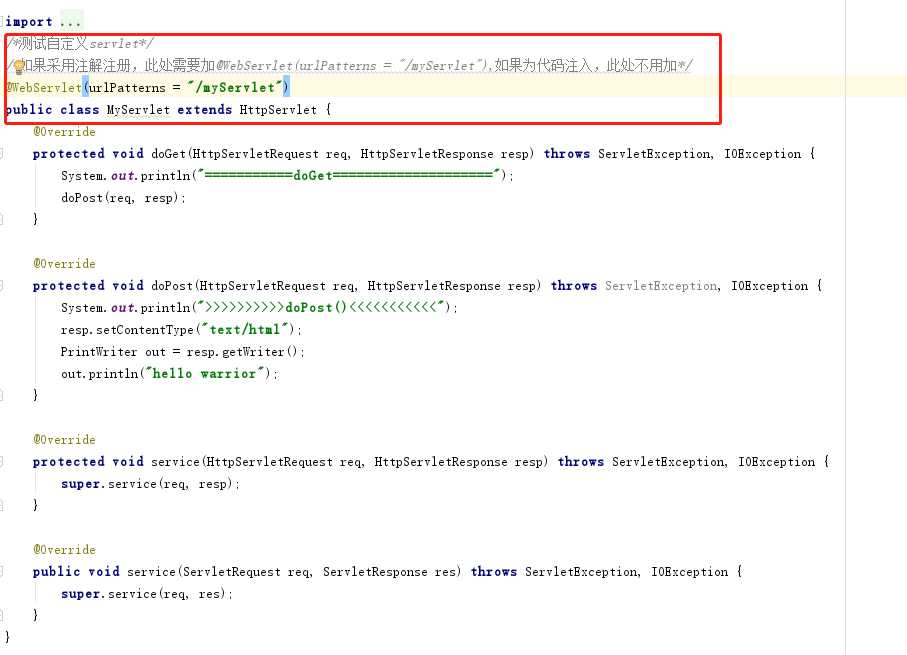
Springboot的主servlet是DispacherServlet，它默认的url-pattern是“/”，如果我们还需要其他的servlet就需要开发人员自己进行定义和注册，有两种注册方式:

\*\*一：代码注册：

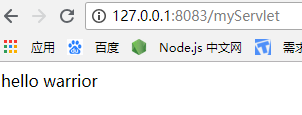


\*\*二：注解注册：  


自定义servlet:



测试结果：



配置filter、listener方法相似，参考：<https://www.cnblogs.com/NeverCtrl-C/p/8191920.html>

1. 内嵌servlet（dispatherServlet）容器配置-公用配置:

\*\* Sever.port：默认8080，可以在properties中修改，可用jar包方式修改：SpringBoot.jar --server.port=8000，也可代码修改

\*\*server.session-timeout:

\*\*server.context-path:默认是/

1. 内嵌servlet（dispatherServlet）容器配置-tomcat配置:

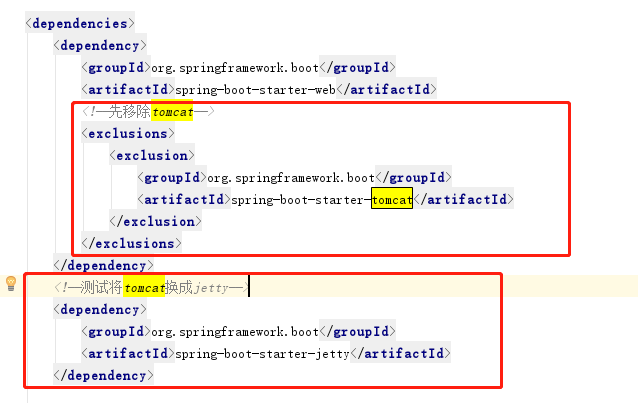
\*\*server.tomcat.uri-encoding：默认utf-8

\*\*server.tomcat.cpmpression:是否开启压缩，默认为false;

1. servlet（dispatherServlet）容器配置-替换tomcat:

\*\*jetty

\*\*undertow



1. Favicon配置

默认开启，



\*\*关闭favicon:**spring.mvc.favicon.enabled** = **false**

\*\*设置favicon:只需将自己的favicon.ico文件放置在类路径根目录、类路径META-INF/resources/下、类路径resources/下、类路径static/下或类路径public/下即可。

1. WebSocket

Websocket是一个持久化的协议（长连接），只需要经过一次HTTP请求，就可以做到源源不断的信息传送了，

\*\*广播式

\*\*点对点式