


```
org.springframework.boot.ApplicationRunner;import
org.springframework.boot.SpringApplication;import
org.springframework.boot.WebApplicationType;import
org.springframework.boot.autoconfigure.SpringBootApplication;import
org.springframework.core.io.buffer.DataBuffer;import
org.springframework.core.io.buffer.DataBufferUtils;import
org.springframework.web.reactive.function.client.WebClient;import
reactor.core.publisher.Flux;@SpringBootApplicationpublic class
ReactiveFileServiceClient implements ApplicationRunner {    public static void main(String[]
args) throws Exception {        SpringApplication app = new
SpringApplication(ReactiveFileServiceClient.class);
app.setWebApplicationType(WebApplicationType.NONE);    app.run(args);
@Override    public void run(ApplicationArguments args) throws Exception {        URL url
= null;        try {            url = new URL(args.getOptionValues("&#34;url&#34;").get(0));
        } catch (Exception e) {            System.err.println("&#34;missing --url option
argument&#34;);            this.help();            return;        }        String out =
null;        try {            out = args.getOptionValues("&#34;out&#34;").get(0);
        } catch (Exception e) {            String surl = url.getProtocol() + "&#34;://&#34; + url.getHost() +
&#34;:&#34; + url.getPort();            String path = url.getFile();
Flux&amp;lt;DataBuffer&amp;gt; data =
WebClient.create(surl).get().uri(path).retrieve().bodyToFlux(DataBuffer.class);        if (out !=
null) {            try (FileOutputStream fos = new FileOutputStream(out)) {
DataBufferUtils.write(data, fos).map(DataBufferUtils::release).blockLast();
System.out.println("&#34;result written to file &#34; + out);            }            private void
help() {                System.out.println("&#34;usage java -jar reactivefileclient-0.0.1-SNAPSHOT.jar
--url= --out=&#34;);                System.out.println("&#34;example:&#34;);
System.out.println("&#34;java -jar reactivefileclient-0.0.1-SNAPSHOT.jar
--url=http://localhost:8080/rest/file?path=in/bigimage.jpg --out=out/bigimage.jpg&#34;);
}            Das File bigimage.jpg kann ersetzt werden durch eine real existierende Datei analog
kann der Output Pfad angepasst werden.
```

Client und Service sind am besten in 2 separaten Spring Boot Projekten zu programmieren z.B. mit der Eclipse IDE.

Feedback

War dieser Blog für Sie wertvoll. Wir danken für jede Anregung und Feedback

Kontakt

Simtech AG
Finkenweg 23
3110 Münsingen
Schweiz

Impressum

Das Copyright für sämtliche Inhalte dieser Website liegt bei Simtech AG, Schweiz.
Beachten Sie auch unsere Hinweise zum Urheberrecht, Datenschutz und Haftungsausschluss.
Jeder Hinweis auf Fehler nehmen wir gerne entgegen.

Copyright

2024 Simtech AG, All rights reserved, Powered by stack.ch written in Golang by Daniel Schmutz

<https://www.simtech-ag.ch/getbytes>