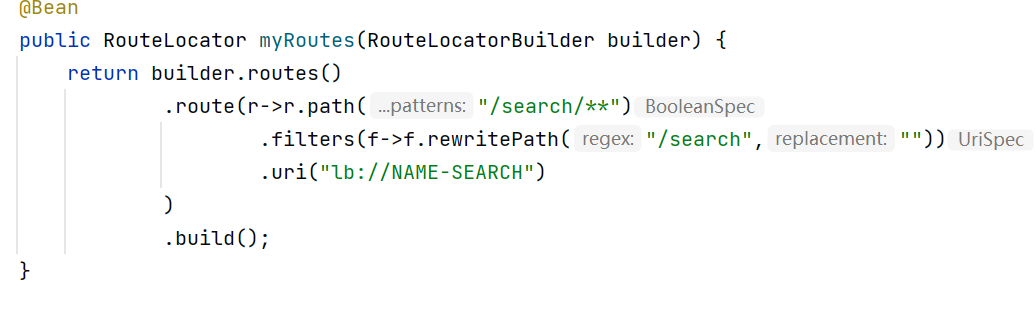
体系架构作业7

1. 我选择的是**任务I，实现输入书名输出对应作者信息的功能**
2. **路由工具**：使用的是老师例子中使用的**RouteLocator**，使用方式如下:



1. **部署以及使用方式**：

对于需要实现的输入书名输出作者的**微服务**，我将其部署在了**localhost:11130**端口上，将**gateway部署在localhost:8090**端口上，将**registry(使用的是eureka)部署在localhost:8040**端口上。具体的使用方式为，client端向gateway发送请求http://localhost:8090/search/SearchBookByName?name={name}形式的请求，gateway在接收到上述请求之后，会先向registry询问服务NAME0SEARCH(对这个输入书名输出作者的服务起的服务名)的位置，之后会将接受的request解析为http://localhost:11130/SearchBookByName?name={name}格式的请求，并将其重定向到11130端口，之后11130端口的服务处理结果之后在发送给client端。

1. **运行截图**：

搜索前：



搜索后：



**微服务架构中gateway和registry的作用**：

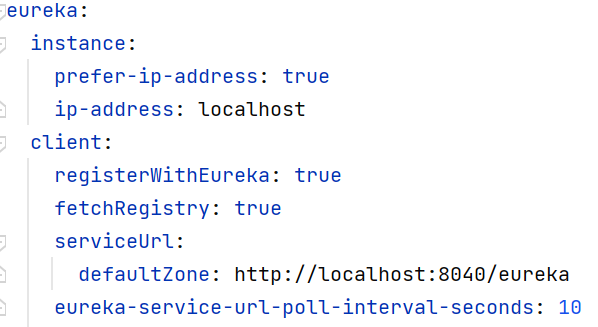
gateway在微服务架构中主要作用是与**client直接通信**，(如在这次实现的微服务中client会先将请求发给gateway而不会直接发给其他位置)并将后端的实现细节向**client端屏蔽(即进行反向代理)**(如在这次实现的微服务中，client会与gateway直接通信而并不知道其调用的搜索微服务的具体位置)并且还有将client端的请求**重定向到对应的service的位置**的作用(在这次实现的微服务中，是gateway将请求重定向到了11130端口的，而不是client自己直接发过去的)。

Registry在微服务架构中的主要作用是**登记每一个注册的微服务的IP，port信息**，从而使得gateway在进行重定向的时候可以知道应该重定向到哪里，即起到了重要的**协调以及通信作用**。如在这次作业的实现的微服务中，gateway需要先与位于8040端口的registry通信才能知道其应该将client的请求转发到11130端口。

后端代码实现以及配置：

Gateway：





NameSearch(即实现的具体微服务代码，由于其后端实现逻辑简单，只体现配置在这里)



Registry(使用eureka实现，只展示了配置信息)

