虽然使用Swagger可以为Spring MVC编写的接口生成了API文档，但是在微服务化之后，这些API文档都离散在各个微服务中，是否有办法将这些接口都整合到一个文档中？

准备工作

上面说了问题的场景是在微服务化之后，所以我们需要先构建两个简单的基于Spring Cloud的微服务，命名为swagger-service-a和swagger-service-b。

下面只详细描述一个服务的构建内容，另外一个只是名称不同，如有疑问可以在文末查看详细的代码样例。

- 第一步：构建一个基础的Spring Boot应用，在pom.xml中引入eureka的依赖、web模块的依赖以及swagger的依赖：

```
<dependencies>
    <dependency>
        <groupId>org.springframework.cloud</groupId>
        <artifactId>spring-cloud-starter-eureka</artifactId>
    </dependency>

    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-web</artifactId>
    </dependency>

    <dependency>
        <groupId>com.spring4all</groupId>
        <artifactId>swagger-spring-boot-starter</artifactId>
        <version>1.7.0.RELEASE</version>
    </dependency>
</dependencies>
```

- 第二步：编写应用主类和控制接口：

```
@EnableSwagger2Doc
@EnableDiscoveryClient
@SpringBootApplication
public class SwaggerServiceaApplication {
    public static void main(String[] args) {
        new
SpringApplicationBuilder(SwaggerServiceaApplication.class).web(true).run(args);
    }
}
```

其中，@EnableSwagger2Doc注解是我们自制Swagger Starter中提供的自定义注解，通过该注解会初始化默认的Swagger文档设置。下面还创建了四个通过Spring Boot编写的HTTP接口，用来后续在文档中查看使用。

```
@RestController
```

```
public class ServiceAController {

    @Autowired
    DiscoveryClient discoveryClient;

    @GetMapping("/service-a1")
    public String serviceA1() {
        String services = "ServiceA1: " + discoveryClient.getServices();
        System.out.println(services);
        return services;
    }

    @GetMapping("/service-a2")
    public String serviceA2() {
        String services = "ServiceA2: " + discoveryClient.getServices();
        System.out.println(services);
        return services;
    }

    @GetMapping("/service-a3")
    public String serviceA3() {
        String services = "ServiceA3: " + discoveryClient.getServices();
        System.out.println(services);
        return services;
    }

    @GetMapping("/service-a4")
    public String serviceA4() {
        String services = "ServiceA4: " + discoveryClient.getServices();
        System.out.println(services);
        return services;
    }
}
```

- 第三步：设置配置文件内容：

```
spring.application.name=swagger-service-a
server.port=2351
eureka.client.serviceUrl.defaultZone=http://localhost:1001/eureka/
swagger.base-package=lizzy.springcloud
```

其中`swagger.base-package`参数制定了要生成文档的package，只有`lizzy.springcloud`包下的Controller才会被生成文档。

注意：上面构建了swagger-service-a服务，swagger-service-b服务可以如法炮制，不再赘述。

构建API网关并整合Swagger

在16 服务网关（D版 基础）.note一节中，已经非常详细的介绍过使用Spring Cloud Zuul构建网关的详细步骤，这里主要介绍在基础网关之后，如何整合Swagger来汇总这些API文档。

- 第一步：在pom.xml中引入swagger的依赖，主要的依赖包含下面这些：

```
<dependencies>
    <dependency>
        <groupId>org.springframework.cloud</groupId>
        <artifactId>spring-cloud-starter-zuul</artifactId>
    </dependency>
    <dependency>
        <groupId>org.springframework.cloud</groupId>
        <artifactId>spring-cloud-starter-eureka</artifactId>
    </dependency>
    <dependency>
        <groupId>com.spring4all</groupId>
        <artifactId>swagger-spring-boot-starter</artifactId>
        <version>1.7.0.RELEASE</version>
    </dependency>
</dependencies>
```

- 第二步：在应用主类中配置swagger，具体如下：

```
@EnableSwagger2Doc
@EnableZuulProxy
@SpringCloudApplication
public class SwaggerApiGatewayApplicaiton {
    public static void main(String[] args) {
        new
SpringApplicationBuilder(SwaggerApiGatewayApplicaiton.class).web(true).run(args);
    }
}

@Component
@Primary
public class DocumentationConfig implements SwaggerResourcesProvider {
    @Override
    public List<SwaggerResource> get() {
        List resources = new ArrayList<>();
```

```
        resources.add(swaggerResource("service-a", "/swagger-service-a/v2/api-docs", "2.0"));
        resources.add(swaggerResource("service-b", "/swagger-service-b/v2/api-docs", "2.0"));
        return resources;
    }

    private SwaggerResource swaggerResource(String name, String location, String version) {
        SwaggerResource swaggerResource = new SwaggerResource();
        swaggerResource.setName(name);
        swaggerResource.setLocation(location);
        swaggerResource.setSwaggerVersion(version);
        return swaggerResource;
    }
}
```

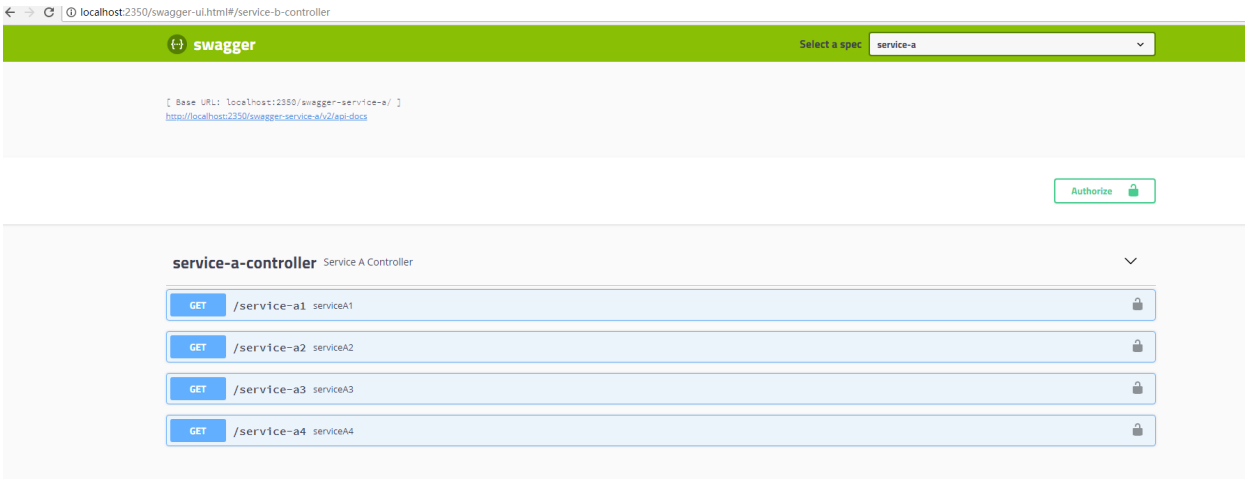
说明：`@EnableSwagger2Doc`上面说过是开启Swagger功能的注解。这里的核心是下面对`SwaggerResourcesProvider`的接口实现部分，通过`SwaggerResource`添加了多个文档来源，按上面的配置，网关上Swagger会通过访问`/swagger-service-a/v2/api-docs`和`swagger-service-b/v2/api-docs`来加载两个文档内容，同时由于当前应用是Zuul构建的API网关，这两个请求会被转发到`swagger-service-a`和`swagger-service-b`服务上的`/v2/api-docs`接口获得到Swagger的JSON文档，从而实现汇总加载内容。

- 第三步：创建配置文件，具体如下：

```
spring.application.name=swagger-api-gateway
server.port=2350
eureka.client.serviceUrl.defaultZone=http://localhost:1001/eureka/
```

测试验证

将上面构建的两个微服务以及API网关都启动起来之后，访问网关的swagger页面，比如：`http://localhost:2350/swagger-ui.html`，此时可以看到如下图所示的内容：



可以看到在分组选择中就是当前配置的两个服务的选项，选择对应的服务名之后就会展示该服务的API文档内容。