**Como consumir uma API externa na sua aplicação Spring Boot**

[#java](https://dev.to/t/java) [#api](https://dev.to/t/api) [#springboot](https://dev.to/t/springboot) [#feign](https://dev.to/t/feign)

[daienelima profile imageDaiene Lima](https://dev.to/daienelima)12 de out. de 2020 ・*Updated on 10 de mar.* ・2 min read

Em algum momento pode acontecer da nossa aplicação ter a necessidade de consumir ou se integrar com um serviço externo.  
Pensando nisso surgiu um projeto chamado Feign.

**O que é Feign?**

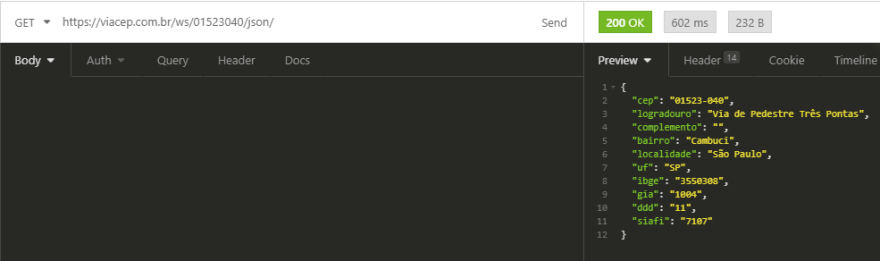
O Feign é um projeto que está dentro do Spring Cloud ele é  
utilizado para integração e consumo de serviços rest.

Ele torna o consumo de clients de serviço da web mais fácil. Foi  
inspirado no Retrofit e JAX-RS-2.0.  
Uma grande facilidade do uso do Feign é que não precisamos  
escrever nenhum código para chamar o serviço, a não ser uma  
definição de interface.

**@FeignClient** é a notação que torna o consumo do serviço  
extremamente simples.

Eu vou mostrar um exemplo simples consumindo a api  
viacep.com.br/ws/{cep}/json/

A api devolve um json na seguinte estrutura;

[](https://res.cloudinary.com/practicaldev/image/fetch/s--wyYcj49j--/c_limit%2Cf_auto%2Cfl_progressive%2Cq_auto%2Cw_880/https:/dev-to-uploads.s3.amazonaws.com/i/7rmunqlgxfd466v2lv5t.png)

* Vamos adicionar duas dependências ao nosso pom.xml

spring-cloud-starter-openfeign

<dependency>

<groupId>org.springframework.cloud</groupId>

<artifactId>spring-cloud-starter-openfeign</artifactId>

</dependency>

spring-cloud-dependencies

<dependencyManagement>

<dependencies>

<dependency>

<groupId>org.springframework.cloud</groupId>

<artifactId>spring-cloud-dependencies</artifactId>

<version>Hoxton.SR8</version>

<type>pom</type>

<scope>import</scope>

</dependency>

</dependencies>

</dependencyManagement>

* Configurações do Projeto

Precisamos anotar @EnableFeignClients na classe principal do  
projeto, para habilitar o Feign, no caso aqui a classe  
Application.  
Com essa anotação habilitamos uma busca pela interface client  
que devem ser consumida.

@SpringBootApplication

@EnableFeignClients

public class ConsumoApplication {

public static void main(String[] args) {

SpringApplication.run(ConsumoApplication.class, args);

}

}

* Criaremos a classe que vai retornar nosso objeto

public class Endereco {

private String cep;

private String logradouro;

private String complemento;

private String bairro;

private String localidade;

private String uf;

private String ibge;

private String gia;

private String ddd;

private String siafi;

}

* Criaremos uma interface onde o @FeignClient irá fazer a chamada do serviço.

Os parâmetros url e name são obrigatórios;  
onde url: é a base do serviço que vai ser consumido  
e name: o nome do cliente  
Precisamos dizer qual o Endepoint o buscarEnderecoPorCep vai  
utilizar;

@FeignClient(url= "https://viacep.com.br/ws/" , name = "viacep")

public interface CepService {

@GetMapping("{cep}/json")

Endereco buscaEnderecoPorCep(@PathVariable("cep") String cep);

}

* Agora é só criar a rota que usaremos para consumir o serviço;

@RestController

public class CepRestService {

@Autowired

private CepService cepService;

@GetMapping("/{cep}")

public ResponseEntity<Endereco> getCep(@PathVariable String cep) {

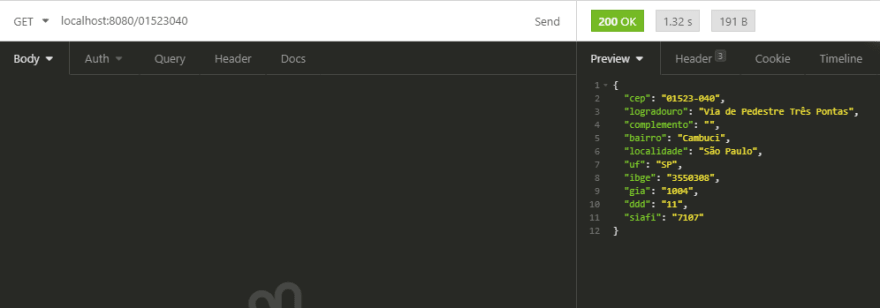
Endereco endereco = cepService.buscaEnderecoPorCep(cep);

return endereco != null ? ResponseEntity.ok().body(endereco) : ResponseEntity.notFound().build();

}

}

Pronto, agora sua aplicação está consumindo um serviço externo  
com poucas linhas e com o mesmo resultado que teríamos com  
RestTemplate.

[](https://res.cloudinary.com/practicaldev/image/fetch/s--JuxznjQo--/c_limit%2Cf_auto%2Cfl_progressive%2Cq_auto%2Cw_880/https:/dev-to-uploads.s3.amazonaws.com/i/skt0a3a1jt9ng7a9ttuq.png)

Link da aplicação modelo no git [Exemplo](https://github.com/daienelima/consumo)

Link do github (<https://github.com/daienelima/consumo>)

Link do conteudo (https://dev.to/daienelima/como-consumir-uma-api-na-sua-aplicacao-spring-boot-3p3a)

Erro que ocorreu

Olá, achei bem interessante esse projeto. Contudo, quando adicionei esse service no meu projeto ocorreu o seguinte erro:

org.springframework.beans.factory.BeanCreationException: Error creating bean with name 'configurationPropertiesBeans' defined in class path resource [org/springframework/cloud/autoconfigure/ConfigurationPropertiesRebinderAutoConfiguration.class]

É necessário algum arquivo @Configuration?

Resposta ao erro que ocorreu:

Tive o mesmo problema que você amigo.

Estava usando a versão 2.4.3 do Spring, aí dei uma gogleada e li relatos de incompatibilidade a partir da 2.4.0  
Achei estranho, mas aí troque a versão para a mesma usado no exemplo e ao fazer o rebuild rodou certinho.