



## Consumindo Outras API's - Rest template

Um modo de consumir uma API é usando um **REST TEMPLATE**, é um objeto que facilita a requisição de outra API, retornando um **ResponseEntity<T>**:

```
public void consumirAPIComRestTemplate(){
    HashMap<String, String> parametros = new HashMap<>();
    parametros.put("amount", "600.0");
    parametros.put("from", "USD");
    parametros.put("to", "BRL");

    ResponseEntity<QualquerObjeto> response = new RestTemplate()
        .getForEntity(
            url: "http://xxxxx/{amount}/{from}/{to}",
            QualquerObjeto.class,
            parametros
        );

    QualquerObjeto corpoDaRequisicao = response.getBody();
    HttpHeaders headersDaRequisicao = response.getHeaders();
    HttpStatus statusDaRequisicao = response.getStatusCode();
}
```

Só um map com os parâmetros que precisam ser passados em uma determinada requisição

Em uma chamada simples da pra passar a URL da requisição, qual classe vai ter no body da resposta e os parametros

ResponseEntity permite recuperar outras partes da resposta para eventual lógica.

Essa abordagem com **Rest Template** funciona, mas não é lá muito bonito.



## Consumindo Outras API's - Feign

Existe um projeto sob o guarda-chuva de Spring Cloud que facilita a vida na hora de trabalhar com requisições API para API. Esse projeto é o **OpenFeign**, a definição que o projeto se impõe é “um cliente WEB declarativo”.

A ideia do **OpenFeign** é tornar o consumo de API's (**clientes web no geral**) mais fácil de ser feita, você precisa da **dependência** e da **anotação** para fazer funcionar:

```
@SpringBootApplication
@EnableFeignClients
public class CambioServiceApplication {

    public static void main(String[] args) {

    }

}
```

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

Precisa de uma **Interface** que vai ter o **nome do serviço** e a **URL principal de consumo**, a interface vai **carregar as assinaturas das requisições** e abstrair de forma dinâmica sua implementação, agora

é só injetar a Interface no componente e usar:

```
@FeignClient(name = "cambio-service", url = "localhost:8080")
public interface CambioProxy {

    @GetMapping(value =("/{amount}/{from}/{to}" )
    Cambio getCambio(@PathVariable BigDecimal amount, @PathVariable String from, @PathVariable String to);
}
```

```
@RestController
@RequestMapping("v1/cambio-service")
public class CambioController {

    private final CambioProxy proxy;

    public CambioController(CambioProxy proxy) {
        this.proxy = proxy;
    }

    public void consumirApiComOpenFeign(){
        Cambio cambio = proxy.getCambio(new BigDecimal("99.0"), from: "USD", to: "BRL");
        System.out.println(cambio);
    }
}
```