# Oque vai ser feito?

Um programa em Java/Spring que retorna uma lista de veiculo com sua tabela fipe

# Oque vamos precisar?

1 pegar a api de tabela fipe

2 Transformar esses dados em objetos

# Iniciando e explicando o código!

## Criando os pacotes

* MODEL
* SERVICE
* PRINCIPAL

## Adicionando dependência no maven

### Criando a interface

package com.allephnogueira.TabelaFipe.service;  
  
public interface IConverterDados {  
 <T> T obterDados(String json, Class<T> classe);

<T> List<T> obterLista(String json, Class<T> classe);

}

Aqui estamos criando um obterLista porque os dados que vem do servidor esta vindo em forma de lista e não de apenas 1 objeto.

## Classe converte dados

public class ConverterDados implements IConverterDados {  
  
 private ObjectMapper mapper = new ObjectMapper();  
  
 public <T> T obterDados(String json, Class<T> classe) {  
 try  
 {  
 return mapper.readValue(json, classe);  
 }catch (JsonProcessingException e) {  
 throw new RuntimeException(e);  
 }  
 }  
  
 @Override  
 public <T> List<T> obterLista(String json, Class<T> classe) {  
 CollectionType lista = mapper.getTypeFactory().constructCollectionType(List.class, classe);  
  
 return null;  
 }  
}

{  
 return mapper.readValue(json, classe);  
 }catch (JsonProcessingException e) {  
 throw new RuntimeException(e);  
 }  
 }  
}

Explicando o código:

CollectionType: Representa uma lista e declara uma variável para receber esse tipo de coleção

getTypeFactory(): é uma fábrica para criar tipos complexos, como coleções e mapas, que o ObjectMapper pode usar para deserializar dados JSON.

**constructCollectionType(List.class, classe)**:

**constructCollectionType**: Este método do TypeFactory é usado para criar um CollectionType específico.

**List.class**: Este argumento especifica que queremos construir um tipo de coleção que é uma lista (List).

**classe**: Este é um argumento do tipo Class<?> que representa o tipo dos elementos que a lista conterá. Por exemplo, se classe for String.class, o CollectionType será para uma lista de strings (List<String>).

Outros exemplos do professor

Vamos chamar o CollectionType para armazenar em lista

Vamos chamar o getTypeFactory = para criar uma coleção

Chamar o constructCollectionType = falar assim: Construa uma coleção que vai ser com a Class/interface List e com a nossa classe

Em resumo ele vai criar uma nova lista, com o tipo que vamos escolher, no caso se a classe for String ele vai criar uma lista em String.

## Criando a classe consumo API

public class ConsumoApi {  
  
 public String obterDados(String endereco){  
  
  
 HttpClient client = HttpClient.*newHttpClient*();  
 HttpRequest request = HttpRequest.*newBuilder*()  
 .uri(URI.*create*(endereco))  
 .build();  
  
 HttpResponse<String> response = null;  
 try { // Aqui vamos fazer uma exceção, se tudo der certo faz isso se não  
 response = client  
 .send(request, HttpResponse.BodyHandlers.*ofString*());  
 } catch (IOException e) { // exibe um erro "AMIGAVEL"  
 throw new RuntimeException(e);  
 } catch (InterruptedException e) {  
 throw new RuntimeException(e);  
 }  
  
 // Aqui é oque vai ser devolvido se tudo der certo, dentro dessa variavel  
 String json = response.body();  
 return json; // Quando chamar a classe ele vai retornar essa informacao  
  
 }  
}

## Criando o record DADOS para receber ID e nome

Por que vamos criar uma classe genérica? Porque todos eles sendo moto, carro ou caminhão eles têm a mesma informação!

Lembrar que como temos o mesmo nome do que vem do servidor, não precisamos usar o @JsonAlies

package com.allephnogueira.TabelaFipe.model;  
  
public record Dados (String codigo, String nome){  
}