

Sample Mappings



Nesta pagina

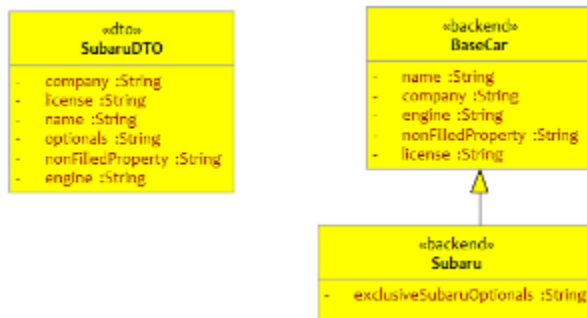
Esta página mostra exemplos de mapeamento entre DTOs e objetos de domínio diversos.

CONTEUDO

[1. Modelo de Dados](#) | [2. Código-Fonte](#)

1. Modelo de Dados

O modelo de dados considerado neste exemplo é dado pelo diagrama UML abaixo:



Neste modelo, existe um objeto DTO SubaruDTO e uma estrutura de domínio (*backend*) dada por uma classe BaseCar estendida por uma classe Subaru.

2. Código-Fonte

NOTA: Os métodos "get" e "set" foram omitidos para fins de legibilidade dos exemplos, mas considero-s como existentes.

2.1. Classes

```
public class SubaruDTO { //DTO
    @Multiple({
        @Fill(field="company", value="subaru.company") //Versao verbosa mapeando o campo 'company' via @Multiple
    })
    private String company;
    @Fill(field="license", value="subaru.license") //Versao intermediaria mapeando o campo 'license'
    private String license;
    @Fill("subaru.name") //Versao enxuta mapeando campo 'name'
    private String name;
    @Multiple({ //Usando @Multiple para mapear o mesmo campo 'optionals' do DTO para dois backends diferentes
        @Fill("mercedes.exclusiveMercedesOptionals"), //Mapeamento para classe "Mercedes"
        @Fill("subaru.exclusiveSubaruOptionals") //Mapeamento para classe "Subaru"
    })
    private String optionals;
    private String nonFilledProperty; //Observe que este campo nao tem mapeamento aqui no DTO...
    private String engine; //Nao tem mapeamento, ou seja, a partir do backend nao esta preenchendo o DTO...
}
```

```

public class BaseCar {
    @Fill("subarudto.name") //Mapeando o backend a partir do DTO
    private String name;
    @Multiple({
        @Fill(field=company, value="subarudto2.company"), //Este mapeamento nao atinge ninguem, pois a classe
'SubaruDTO2' nao existe
        @Fill("subarudto.company") //Mapeando a 'company' do DTO para o backend
    })
    private String company;
    @Fill("subarudto.license") //Mapeando a 'license' do DTO para o backend
    private String license;
    @Fill("subarudto.engine") //... mas a partir do DTO, se preenche a 'engine' do backend
    private String engine;
    private String nonFilledProperty; //...e tambem nem no backend existe mapeamento, logo este campo nao sera
preenchido
}

```

```

Multiple({
    @Fill(field="name", value="subarudto.name + ' override') //Sobreescrendo o mapeamento definido em 'BaseCar'
public class Subaru extends BaseCar {
    @Fill("subarudto.optionals") //Mapeando o o atributo "optionals" do DTO para o atributo
"exclusiveSubaruOptionals" do backend
    private String exclusiveSubaruOptionals;
}

```

2.2. Codigo Principal

```

//Create the DTO
SubaruDTO sdto = new SubaruDTO();
sdto.setEngine("FugiruNaKombi 99HP");
sdto.setName("TakaKaraNuMuru");
sdto.setLicense("VOID-0000");
sdto.setOptionals("Zaerofolio");
sdto.setNonFilledProperty("Has a value in DTO");

//Create the Backend
Subaru s = new Subaru();

//Fill the backend from a DTO
Murdoc.fill(s, sdto);

//Print the results
System.out.println("subaru.name.....: " + s.getName()); //TakaKaraNuMuru override
System.out.println("subaru.license.....: " + s.getLicense()); //VOID-0000
System.out.println("subaru.engine.....: " + s.getEngine()); //FugiruNaKombi 99HP
System.out.println("subaru.exclusiveSubaruOptionals.: " + s.getExclusiveSubaruOptionals()); //Zaerofolio
System.out.println("subaru.nonFilledProperty.....: " + s.getNonFilledProperty()); //null

```