Data Transfer Object (DTO)

Introdução

DTO é um padrão de projeto usado para transportar dados de um local para outro na aplicação. Muitas vezes esse transporte ocorre fora do processo da aplicação, se comunicando com servidor e cliente, entre servidores ou até entre outras partes da aplicação, mas nada impede que ocorra dentro da aplicação.

DTO é apenas um objeto de dados, portanto não costuma ter comportamento próprio (pode ter comportamentos necessários para a linguagem conseguir trabalhar com ele, mas não dentro da semântica do próprio objeto), e como serve para transportar vem o nome Data Transfer Object, ou Objeto de Transferência de Dados.

Introdução

O DTO se opõe ao *model* justamente por não ter comportamentos de regras de negócio, de persistência ou outra forma de manipulação de dados.

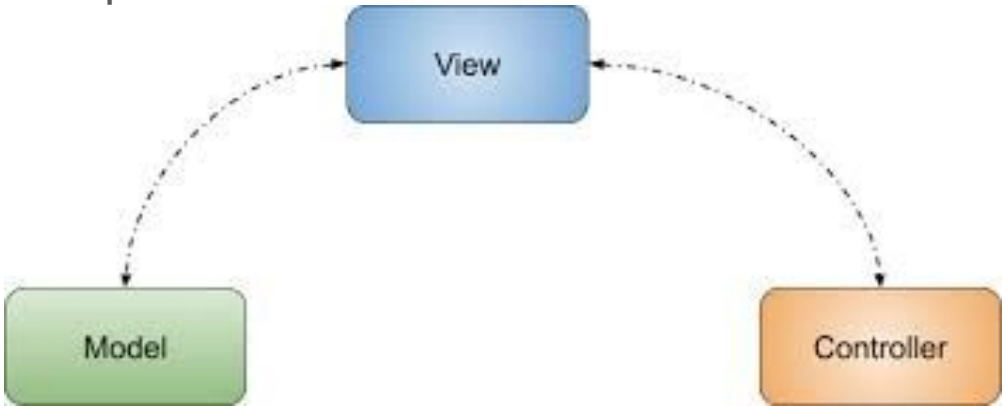
A forma de comunicação, especialmente fora do processo, não é algo que o DTO se importe ou até mesmo que saiba. É comum ser serializado, mas isso não precisa acontecer. Muito menos há um formato definido para essa comunicação. O programador sabe que haverá uma comunicação, mas o DTO não sabe, não precisa saber, e não deve ser responsável por isso.

Em geral, o DTO é considerado um objeto anêmico por não ter comportamento. O comportamento dificulta o uso desses dados em outro local porque precisaria manter controle de versão e garantir que o outro lado esteja sempre sincronizado. Sem comportamentos fica muito mais fácil manter isso, embora ainda há necessidade de sincronizar a estrutura dos dados, mas é muito mais fácil por ela ser mais estável.

Introdução

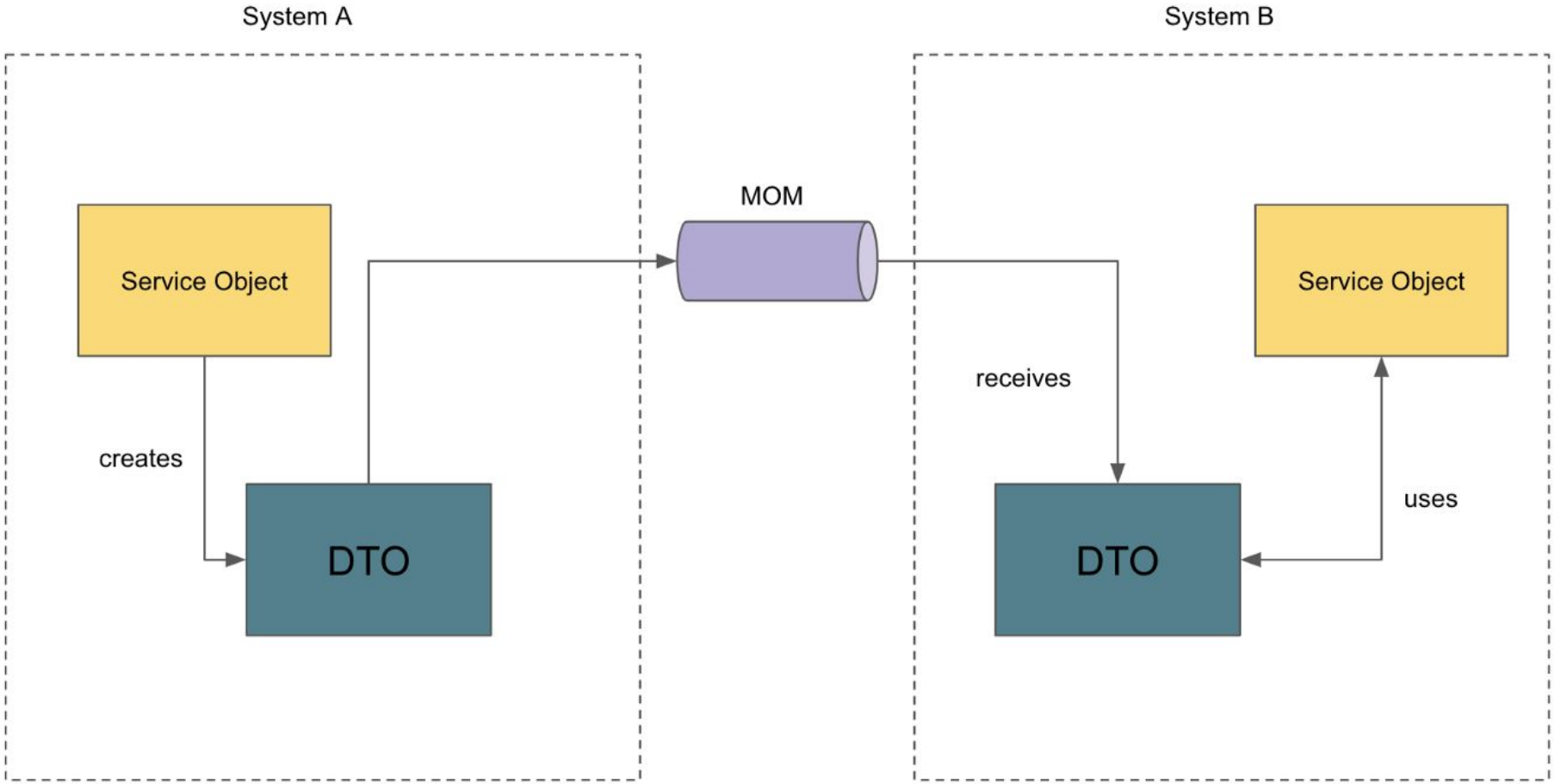
O DTO deve ser uma estrutura estática durante o seu tempo de vida, permitindo que as classes mais completas possam assumir outras formas.

Usa-se o DTO para mapear informações obtidas a partir das entidades presentes no banco de dados e então usa-los numa *View* (MVC). Fazemos isso para otimizar a apresentação dos dados, afinal o *Controller* já recebe as informações prontas para uso.



Introdução

Note que DTO não é algo do Java, e funciona igual em todas as tecnologias, podendo até mesmo ser usado entre linguagens diferentes. O que pode ser específico, nem do Java, mas de um framework específico, é a forma de implementar o DTO.


System A
System B
MOM
Service Object
Service Object
creates
receives
uses
DTO
DTO

Model Mapper

O objetivo do Model Mapper é facilitar o mapeamento de objetos, determinando automaticamente como um modelo de objeto é mapeado para outro. Dessa forma, podemos simular o que um humano faria ao lidar com casos de uso específicos.

Referencia: https://modelmapper.org/

Adicionando a ferramenta Model Mapper ao Pom.xml

Para adicionarmos a ferramenta a nossa API, vamos adicionar a seguinte dependência ao Pom.xml

```
<dependency>
    <groupId>org.modelmapper</groupId>
    <artifactId>modelmapper</artifactId>
    <version>3.0.0</version>
</dependency>
```

Definindo um Bean para o Model Mapper

Para utilizarmos a ferramenta Model Mapper, vamos precisar de uma classe de configuração. Para isso, vamos criar a seguinte classe:

```
@Dependent
public class ModelMapperConfig {
     @Produces
     @DefaultBean
     public ModelMapper modelMapper() {
          var modelMapper = new ModelMapper();
          return modelMapper;
     }
}
```