Aanvullingen architectuur

Data Transfer Objecten

DTOs staat voor **Data Transfer Objects**. Ze zijn een programmeerconcept dat wordt gebruikt om gegevens tussen verschillende lagen van een applicatie of tussen systemen over te dragen. DTOs worden vaak gebruikt in softwareontwikkeling, vooral bij architecturen zoals **MVC (Model-View-Controller)** of in **microservices**.

Waar worden DTOs voor gebruikt?

DTOs worden ontworpen om:

1. **Data transporteren**: Overbrengen van data tussen systemen of lagen zonder logica.
2. **Loskoppelen van lagen**: Ze voorkomen directe afhankelijkheid tussen lagen zoals de database en de UI.
3. **Structureren van data**: Ze bieden een duidelijke structuur voor de data die overgedragen moet worden.
4. **Efficiëntie verbeteren**: Beperken van de hoeveelheid data die over een netwerk wordt verzonden door alleen relevante velden op te nemen.
5. **Beveiliging**: DTOs kunnen gevoelige gegevens uitsluiten die niet naar de client of andere lagen mogen worden verzonden.

Kenmerken van DTOs:

- DTOs zijn meestal eenvoudige, lichtgewicht objecten.
- Ze bevatten geen bedrijfslogica.
- Ze bestaan voornamelijk uit **velden (attributen)** en optioneel **getters/setters**.

Voorbeeld van DTOs

Stel, je hebt een database-entiteit van een gebruiker:

```
public class UserEntity {
   private Long id;
   private String name;
   private String email;
   private String password;
   // Getters en setters
}
```

Dit object bevat veel details, zoals het wachtwoord, die niet naar een frontend of externe API verzonden hoeven worden. Hier komt een DTO van pas.

Een eenvoudige UserDTO kan er als volgt uitzien:

```
public class UserDTO {
   private String name;
   private String email;
   // Getters en setters
}
```

Bij het ophalen van gegevens van de database kan een service de gegevens van UserEntity kopiëren naar UserDTO en alleen de relevante informatie doorgeven.

Wanneer gebruik je DTOs?

1. **API's**: Wanneer gegevens via REST of GraphQL API's worden verzonden, gebruik je DTOs om de response-structuur te definiëren.
2. **Microservices**: Voor communicatie tussen microservices.
3. **Complexe applicaties**: Om lagen zoals controllers, services en repositories los te koppelen.

Voor- en nadelen van DTOs

Voordelen:

- **Loskoppeling**: Verbetert de modulariteit van de code.
- **Veiligheid**: Verbergt gevoelige of onnodige data.
- **Flexibiliteit**: Vergemakkelijkt het aanpassen van gegevensstructuren zonder de onderliggende database of logica aan te passen.

Nadelen:

- **Meer code**: Het creëren en onderhouden van DTOs voegt complexiteit toe.
- **Prestatie-impact**: Het mappen van entiteiten naar DTOs kan extra tijd en geheugen verbruiken.

Conclusie

DTOs zijn een krachtig hulpmiddel in softwareontwikkeling, vooral in gestructureerde en schaalbare applicaties. Ze maken datatransport veiliger, efficiënter en flexibeler door een duidelijke scheiding te bieden tussen verschillende lagen of systemen.

(bron : ChatGPT 26/11/2024)

Oefening

We schrijven een REST-service voor het beheren van projecten. Een project heeft een unieke numerieke id, een naam en een beschrijving. We werken in een gelaagde architectuur waarbij de business-laag centraal staat. Ids voor de projecten worden door de datalaag aangemaakt. Schrijf een REST-service voor het beheer van de projecten.