

Øvelse 26 – Fra DTO til model med domæneprimitiver

Information

Når ASP.NET Core-frameworket modtager data fra en anden applikation, håndteres denne data først i en Controller-klasse (også benævnt servicelaget). Dataen kan enten være en simpel type såsom int eller string, men den kan også være en kompleks type, som du så i [Opgave 20](#). Komplekse typer, der bliver brugt til at modtage data, omtales ofte som et *Data Transfer Object* (herefter DTO). Eksemplet fra opgave 20 er vist herunder.

```
public class PersonDTO
{
    [NotNull]
    public string Firstname { get; set; }
    [NotNull]
    public string Lastname { get; set; }
}
```

Som navnet antyder, er dette objekt beregnet til at sende data til andre applikationer eller modtage data fra andre applikationer. Selvom objektet har attributter og kan inputvalideres, er der intet til hinder for, at objektet kan oprettes et andet sted i softwaren uden at overholde nogen domæne-regler. Et objekt af typen `PersonDTO` kan oprettes hvor som helst i softwaren med et konstruktørkald: `PersonDTO personDTO = new PersonDTO();`. Dette er u hensigtsmæssigt, da en udvikler kan komme til at anvende dette objekt andre steder i koden, uden at der er nogen garanti for, at data i objektet overholder gældende domæne-regler. Derfor bør DTO-typer kun eksistere og anvendes i servicelaget (Controller-klasser i ASP.NET Core). Øvrige dele af softwaren bør kun anvende objekter, der som minimum håndhæver invarians, men også gerne domæneprimitiver.

I [Opgave 20](#) blev det modtagne DTO-objekt konverteret til et `Person`-objekt, der håndhæver invarians, som vist nedenfor i udsnittet fra `NameController` klassens `ValidateName`-metode.

```
if (!ModelState.IsValid)
{
    return View("Index", new { nameIsValid = false, showNameEvaluation = true });
}

// DI intentionally omitted here for clarity
PersonRepository personRepository = new PersonRepository();

Person personWithInvariance = new Person(person.Firstname, person.Lastname);
```

```
personRepository.AddPerson(personWithInvariance);
```

Dette er god praksis. Men Person-objektet bliver ikke oprettet med domæneprimitiver. Dette skal gøres i denne opgave. Projektet, der skal arbejdes med, er stadig

InvarianceDTOandModel fra repoet

<https://github.com/mesn1985/InputValidationsBasicExercises>.

□ Instruktioner

1. □ Opret en Firstname-domæneprimitiv i mappen *Models* (reglerne for navnet er de samme som tidligere).
2. □ Opret en Lastname-domæneprimitiv i mappen *Models* (reglerne for navnet er de samme som tidligere).
3. □ Udskift de to private fields i Person-klassen med domæneprimitiverne.
4. □ Lad konstruktøren i Person-klassen tage de to domæneprimitiver som argumenter og brug dem til at sætte de to fields.
5. □ I klassen *NameController* skal du i `ValidateName`-metoden sikre dig, at Person bliver initialiseret korrekt med domæneprimitiverne.
6. □ Test det ved at køre applikationen flere gange med forskellige gyldige inputs.

□ Refleksionsspørgsmål

1. □ Hvilke fordele opnår man ved at anvende domæneprimitiver fremfor rå datatyper i modeller?
2. □ Hvordan understøtter brugen af domæneprimitiver principperne om sikker softwareudvikling og defense in depth?
3. □ Hvordan hjælper denne tilgang med at undgå, at ugyldige data forplanter sig gennem systemet?
4. □ Hvilke overvejelser bør man gøre sig omkring performance og kompleksitet, når man bruger domæneprimitiver i større systemer?
5. □ Hvordan kobler denne metode sammenhængen mellem DTO og domænemodellen i softwarearkitekturen?

□ Kobling til relevante standarder og principper

Element i øvelsen	Relateret standard / princip	Kommentar
Invarianshåndhævelse	ISO/IEC 27002:2022, afsnit 8.26 (applikationssikkerhed)	Anbefaler validering og integritetstjek af input til en applikation.
	OWASP ASVS 5.0.0, Kontrol 16.5 (Error Handling)	Krav om korrekt håndtering af fejl(undtagelser)
	Secure by Design (Johnsson et al.)	Fremhæver princippet om at skabe robuste domænemodeller og sikre objekters tilstand gennem konstruktion. Herunder domæne primitiver

▮ Domæneprimitiver hjælper med at sikre, at kun gyldige data kan eksistere i systemet, hvilket forebygger mange almindelige fejl og sårbarheder.

Last update: 2025-10-30 12:07:19