

 **Tutorial:**

**Kommuner og regioner**

**Prog. 2 @ EK**

# Målet

**Byg en lokal version af offentligt tilgængeligt data om regioner og kommuner**

1. **Kilde:** DAWA (Danish Web Access) API.
2. **Opgave:** Hent hver **region** og **kommune**.
3. **Udfordring:** Oprethold relationen mellem dem i vores egen database.

# Data relationer(revisited)

## Hierakiet:

**Region (Parent):** Folderen (f.eks., "Hovedstaden").

**Kommune (Child):** Filen (f.eks., "København").

## Reglen:

En region har **Mange** kommuner.

En kommune hører til **En** region.

# "Null" fælde

I Java hvis vi laver en List eller Set uden at sætte værdien så er den `null`.

```
private Set<Kommune> kommuner; // Værdien er null!
```

Hvis vi prøver at tilføje en kommune hertil... **CRASH** ✨ ( `NullPointerException` ).

Altid initialiser Collections med det samme!

```
// Her har vi en tom kasse, klar til brug.  
private Set<Kommune> kommuner = new HashSet<>();
```

# Lists vs. Sets

Vi bruger begge idag, hvorfor?

## 1. List

**Brugt til:** Transport af data (API kald).

**Hvorfor:** Tingene er i rigtig orden, først har vi A så B.  
*Godt når vi henter data ned.*

## 2. Set

**Brugt til:** Gemme data (Entities).

**Hvorfor:** Forbyder **Duplicates**. "Der kan kun være en Vesthimmerlands."  
*Godt når vi gemmer data(bevarer integritet).*

# Hvad vi skal idag

**Hent:** Hent JSON fra den eksterne API.

**Map:** Map det til vores Java Entities (Region & Kommune).

**Gem:** Få dataen gemt i MySQL.

**Tjek:** Åben Workbench/DataGrip og tjek at relationen virker.

**Lad os gå igang** 