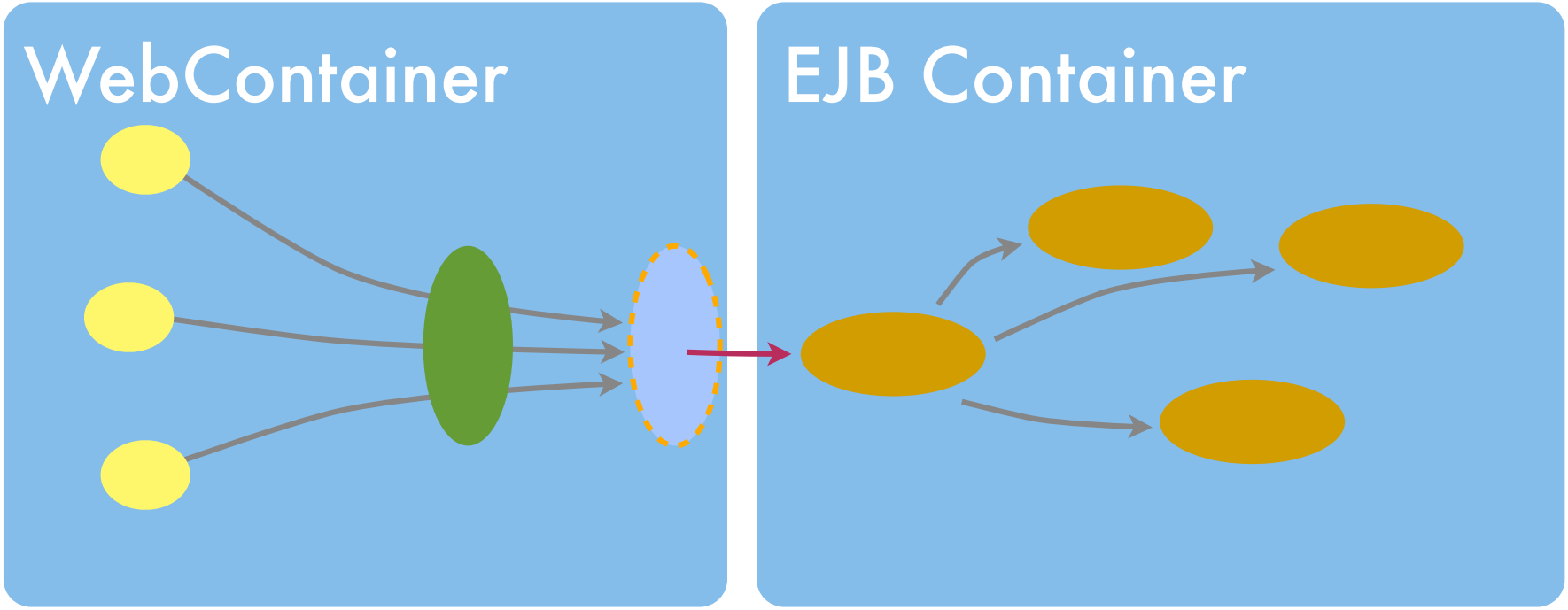
# Session Beans

joost.vennekens@kuleuven.be

# Session bean

- Session bean zorgt voor interactie van een client met business logic in EJBs
  - Bv. client = controller van web applicatie
- Session bean is niet persistent
  - Leeft zolang als zijn client

# Scheiding web <> EJB

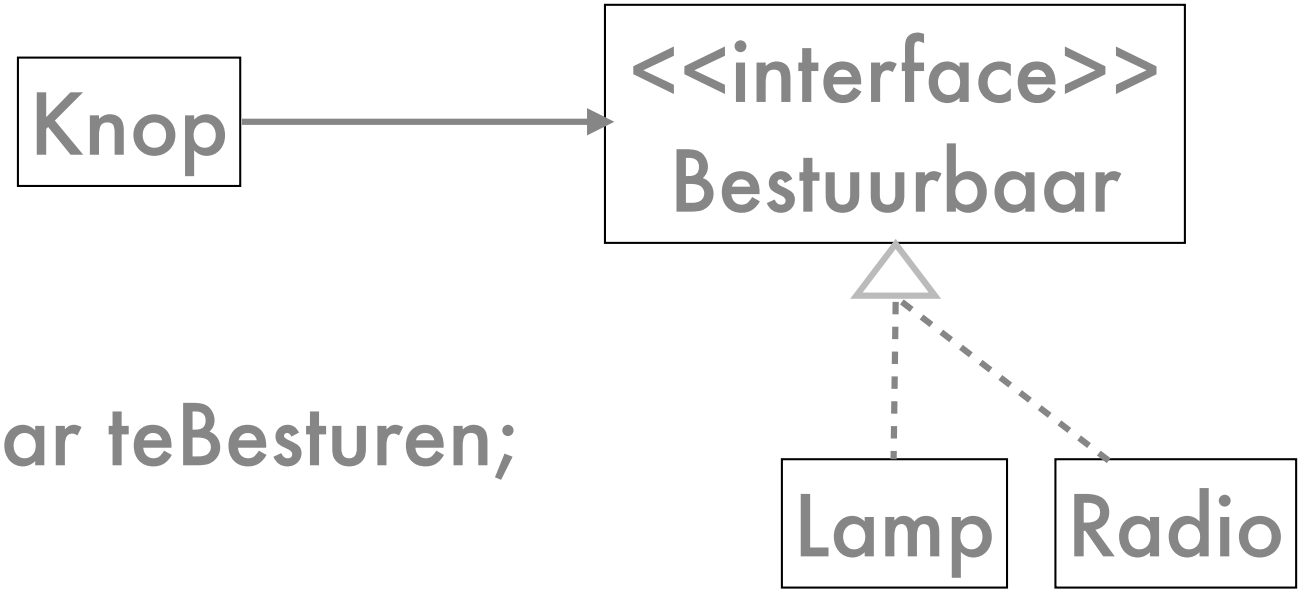


# Design




```
public class Knop {
  private Lamp teBesturen;
  public void druk() {
    if (lamp.isAan())
      lamp.zetAan();
    else lamp.zetUit();
  }
}
```

# Scheiding

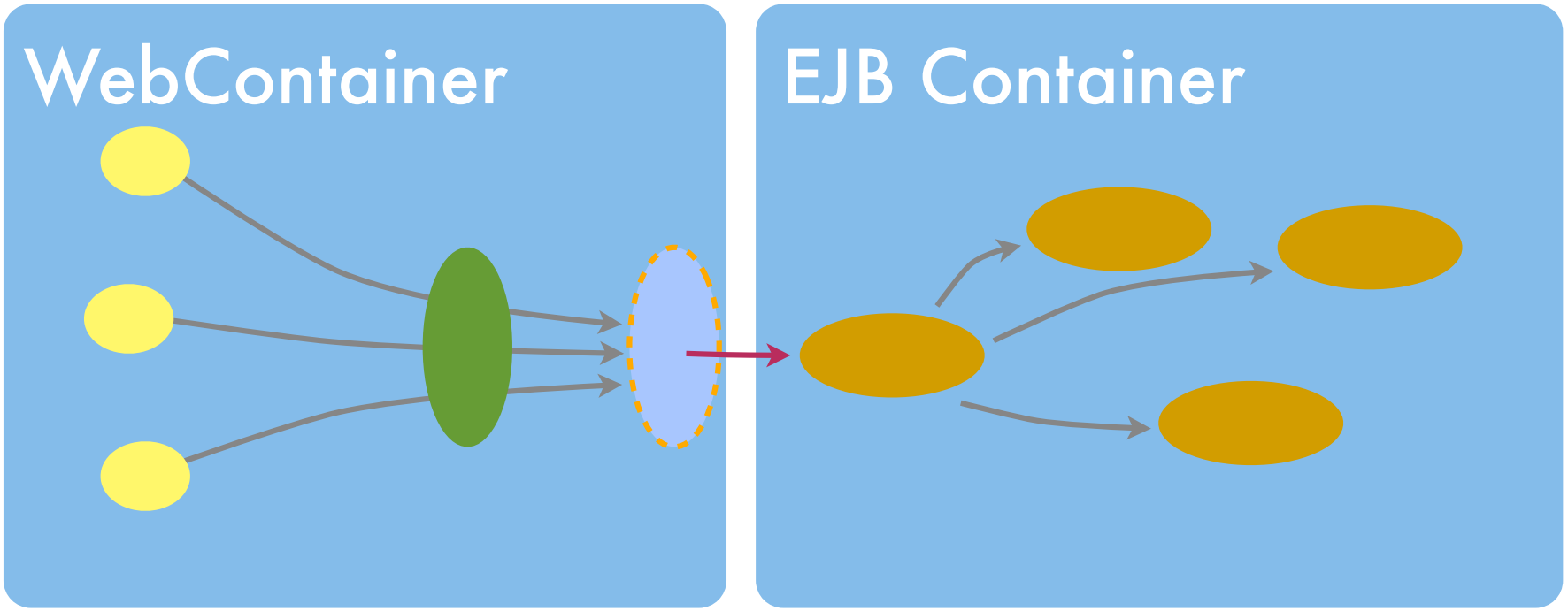


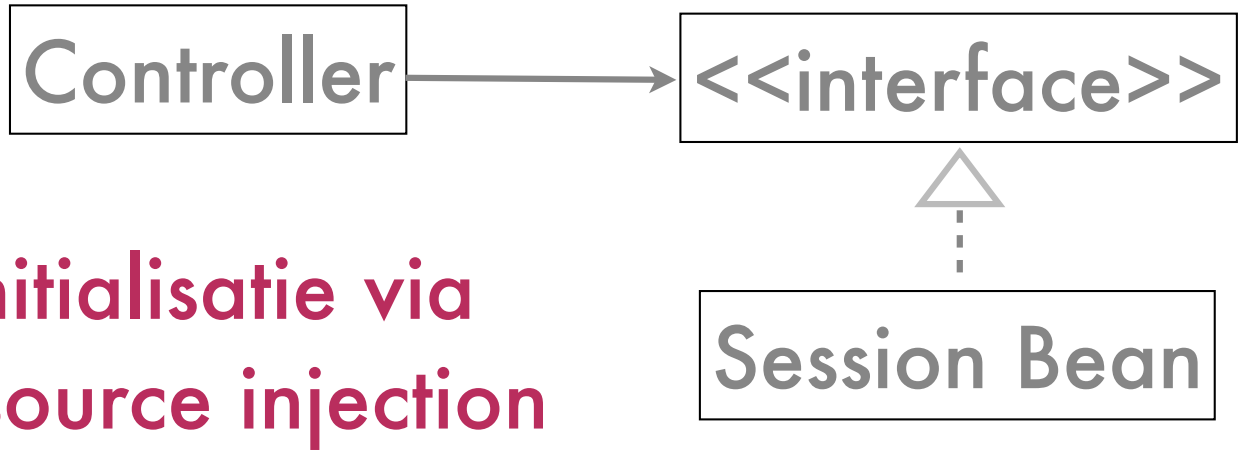

```
public class Knop {
  private Bestuurbaar teBesturen;
  ...
}
public class Lamp implements Bestuurbaar {
  private Bestuurbaar teBesturen;
  ...
}
```

# Probleem?

```
public class Knop {
  private Bestuurbaar teBesturen;
  public Knop () {
    teBesturen = new Lamp();
  }
}
```

# Scheiding web <> EJB




Initialisatie via resource injection

# Annotaties

- Nieuw Java feature
- Gestructureerde manier om extra info toe te voegen aan code (ipv. commentaar)

```
@Author(
   name = "Joost Vennekens",
   date = "20/03/2010"
)
class MyClass() { }
```

# Annotaties

- Gedeclareerd als interface

```
interface Author {
   String name();
   String date();
}
```

- Gedrag aan gekoppeld:
  - Door compiler
  - Door andere tools (javadoc,...)
  - At runtime

# @EJB

- Bij attribuut
- Boodschap aan de WebContainer:
  - Zoek EJB container
  - Zoek in JNDI object van een klasse die in dit attribuut past
  - Maak aan

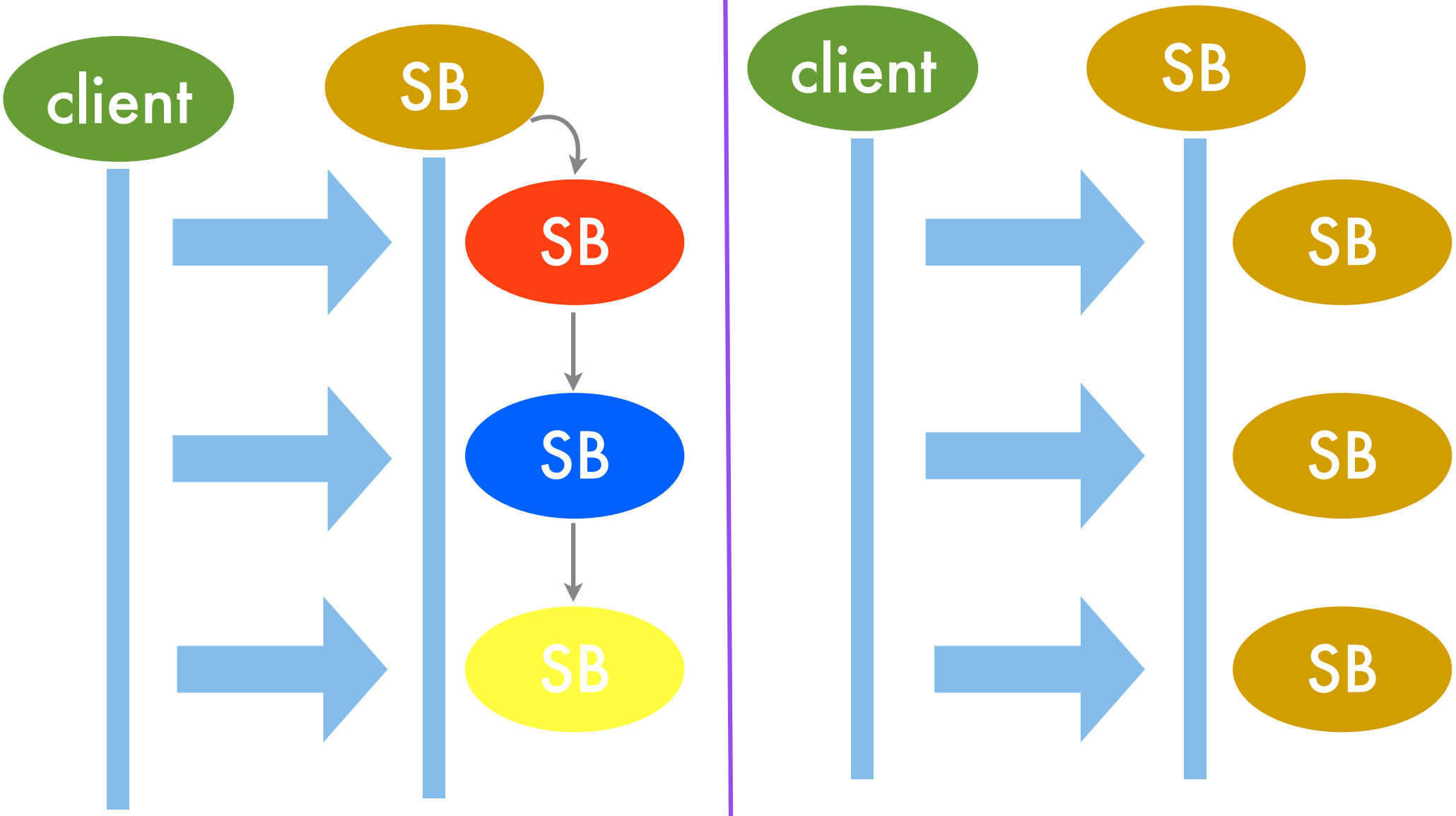
# Session bean

- Verschillende soorten
  - Stateful <-> Stateless
- Werkt via interface
  - Remote <-> Local

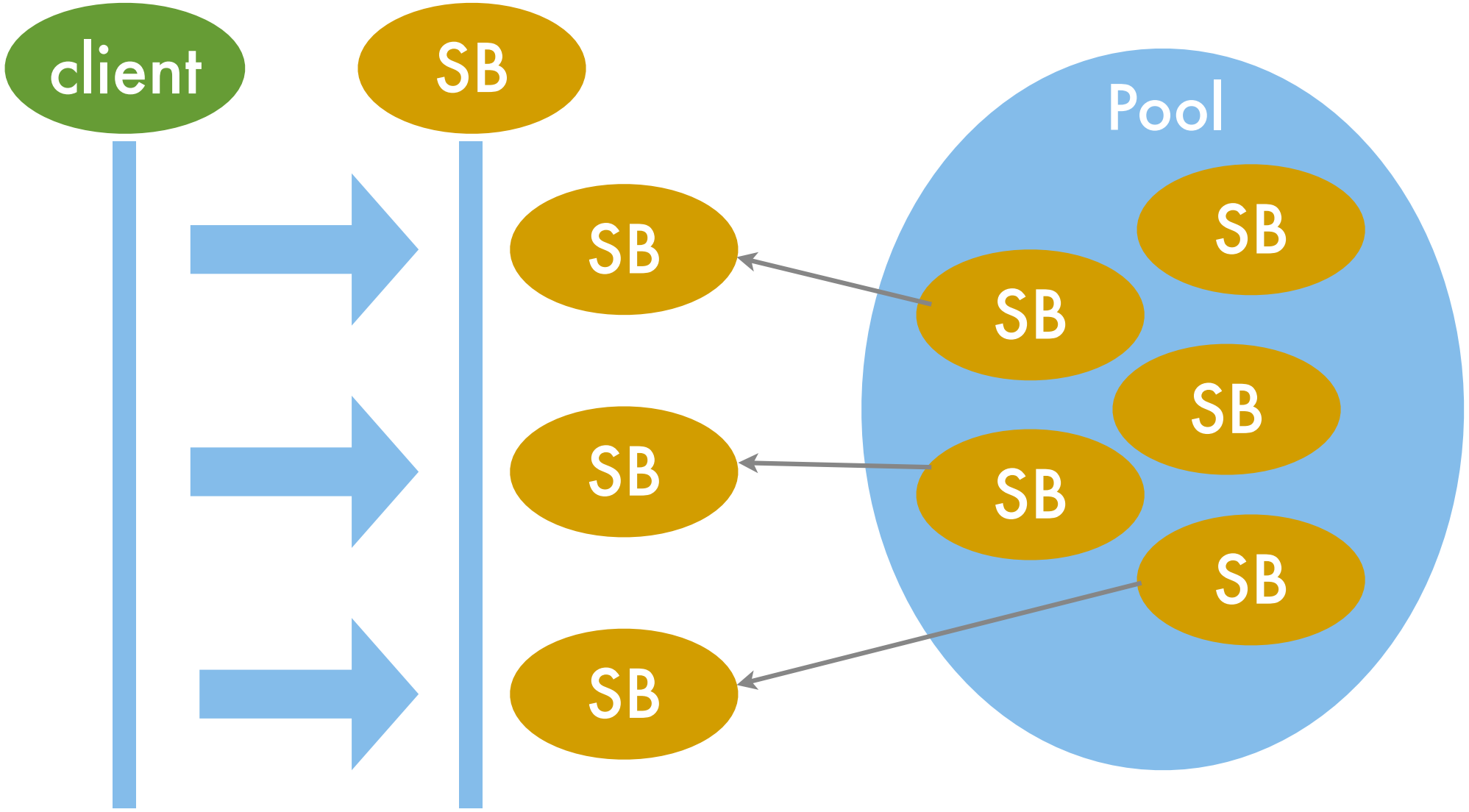
# Twee soorten

- Stateful
  - Session bean houdt informatie bij in attributen
  - Toestand kan veranderen
- Stateless
  - Session bean houdt geen informatie bij in attributen
  - Heeft altijd dezelfde toestand

# Stateful vs stateless

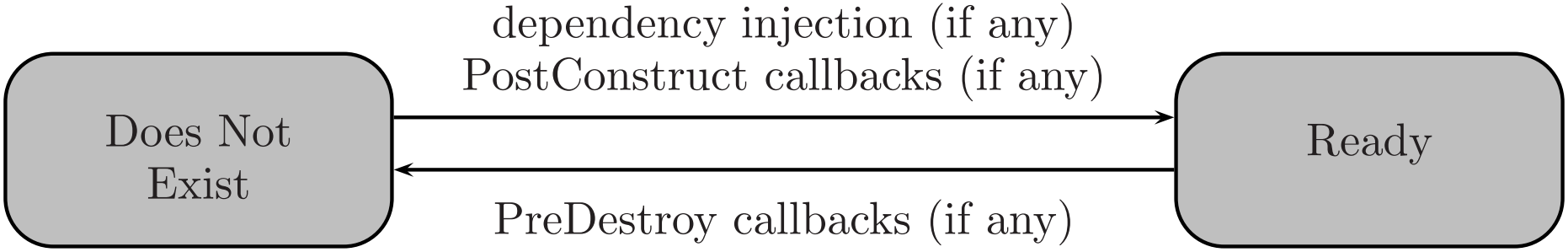


# Voordeel van stateless



# Levenscyclus

## Stateless

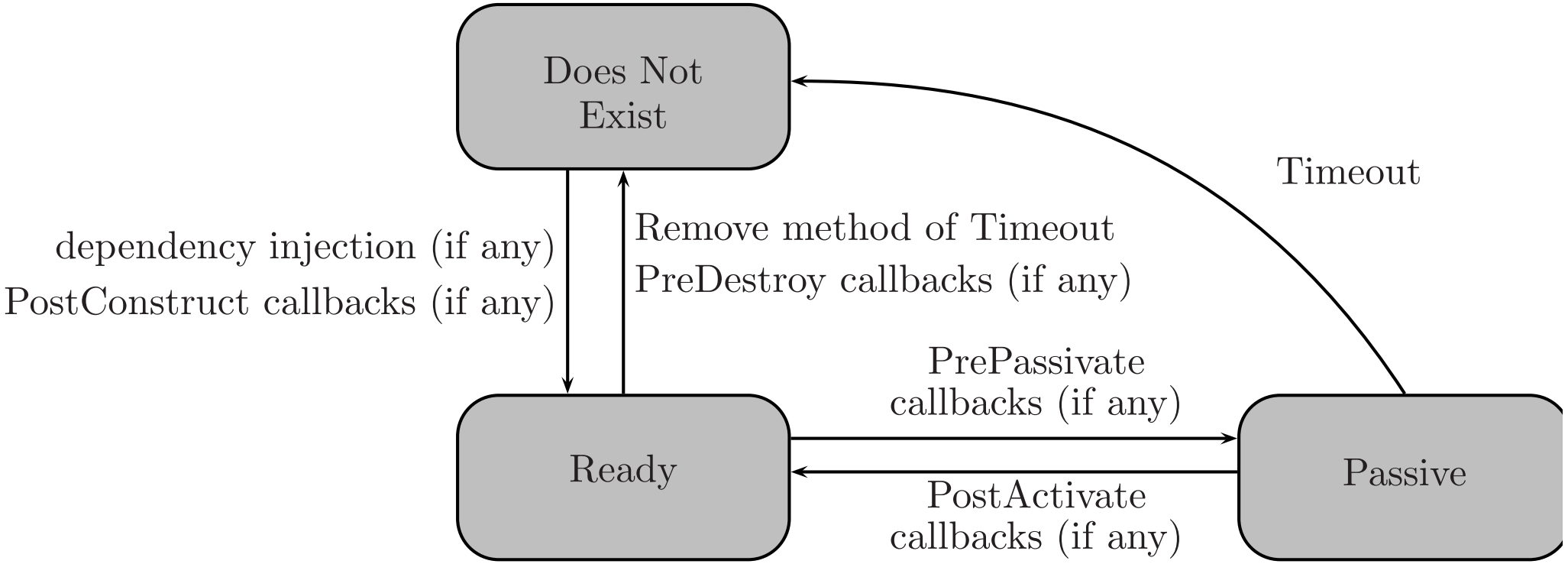


# Callback

- @ javax.annotation.PostConstruct
  - Na volledige initialisatie
  - Aantal en naam naar wens
- @ javax.annotation.PreDestroy

# Levenscyclus

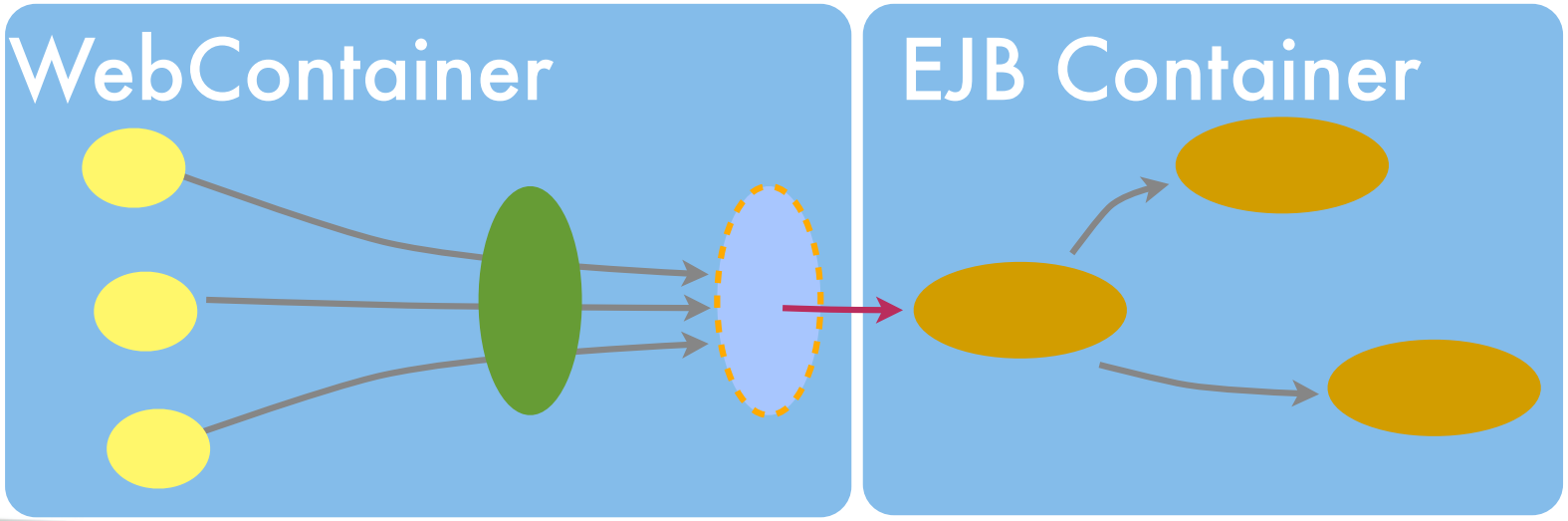
## Stateful



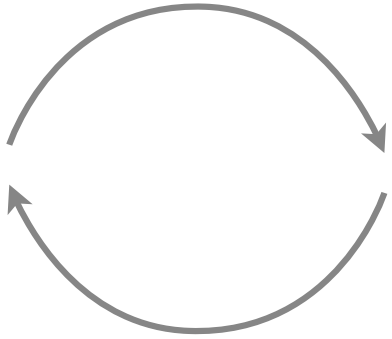
# Callback

- @javax.ejb.PrePassivate
- @javax.ejb.PostActivate

# Local versus remote

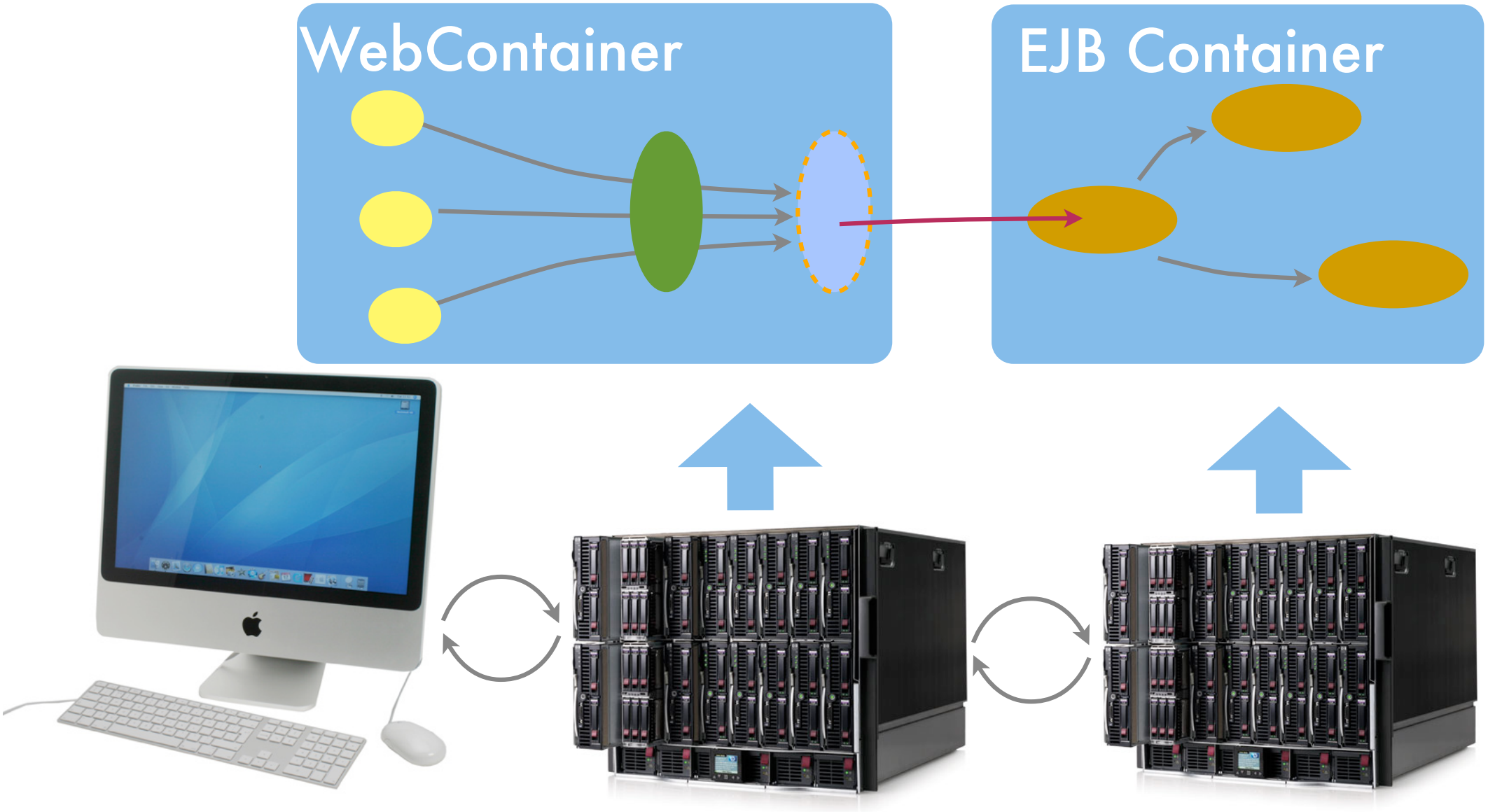




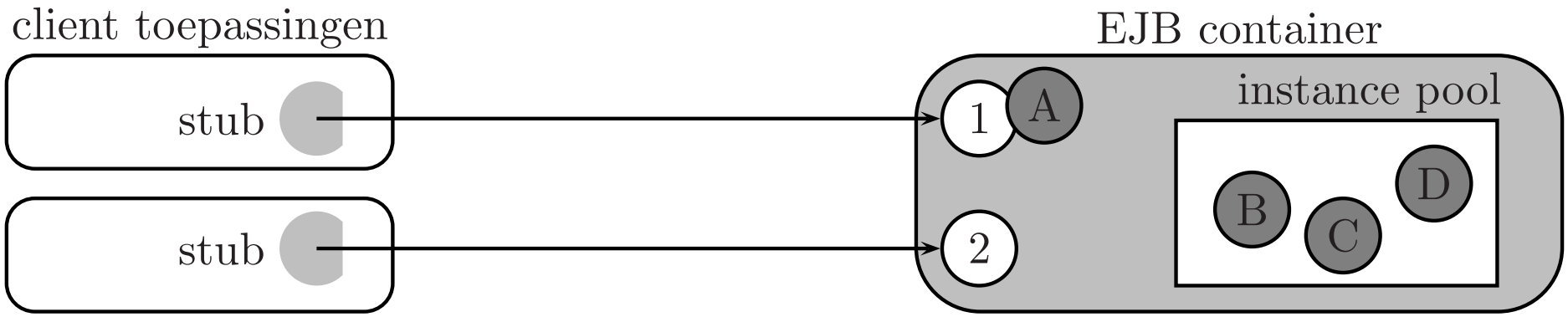





# Local versus remote



# Implementatie




Voordeel: flexibel

Nadeel: performantie

# Create SB

- Creatie via “dependency injection”
  (Niet in JSPs, wel in Servlets)
- JNDI lookop:
  InitialContext ic = new InitialContext();
  ic.lookup(sessie.class.getName());

# Voorbeeld

- BierMetSessionBean