# JSP & Servlets

Database toegang

# Datasource aanmaken

# Database connecties

# Connecties bewaren

## Probleem:

- Om een database bewerking te doen → altijd een connectie nodig
- Een connectie openen kost tijd
- Webapplicatie:
  - veelvuldige connecties nodig
  - gelijktijdige connecties

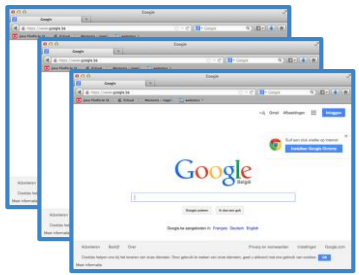
## Dus…

Bij een browser request **GEEN nieuwe Connection** maken, maar een **bestaande Connection hergebruiken**

# Hoe Connections bewaren ?

**Als private variabele in een servlet ???**

**browser**



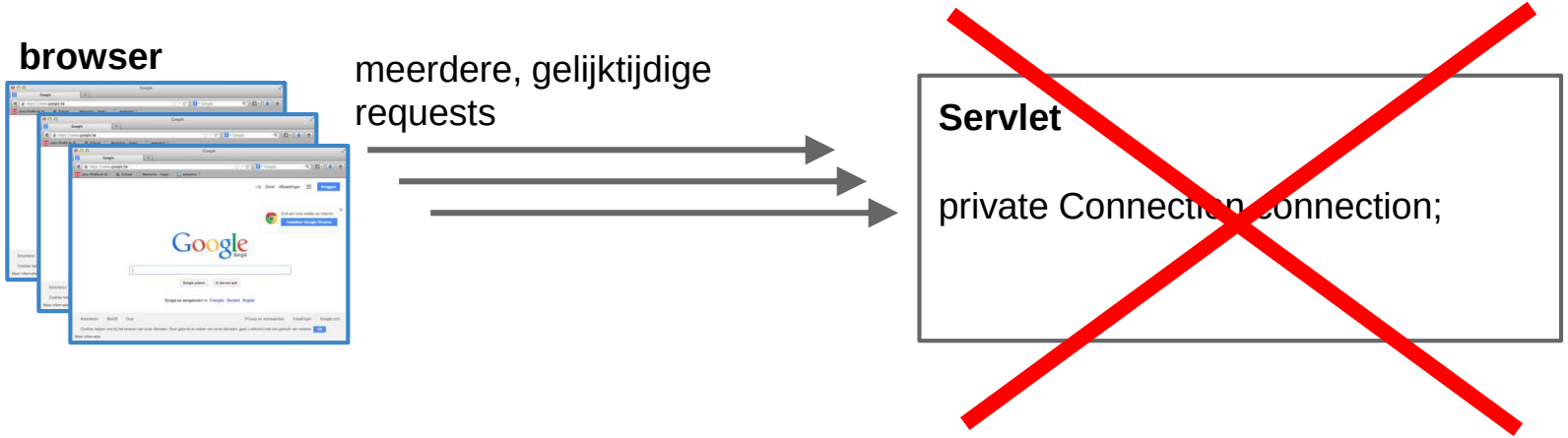

meerdere, gelijktijdige requests

**Servlet**

private Connection connection;

# Hoe Connections bewaren ?

**Als private variabele in een servlet ???** **NEEN !**

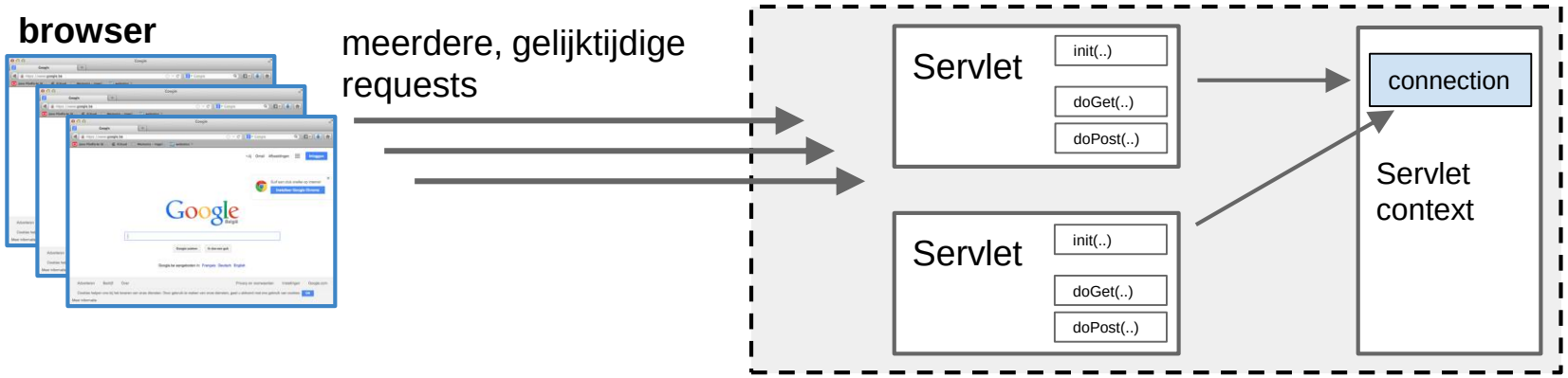



De webserver roept één servlet instance op met meerdere gelijktijdige threads. Deze threads zouden dan dezelfde Connection gebruiken.

Dit mag niet: **een Connection is niet thread-safe !**

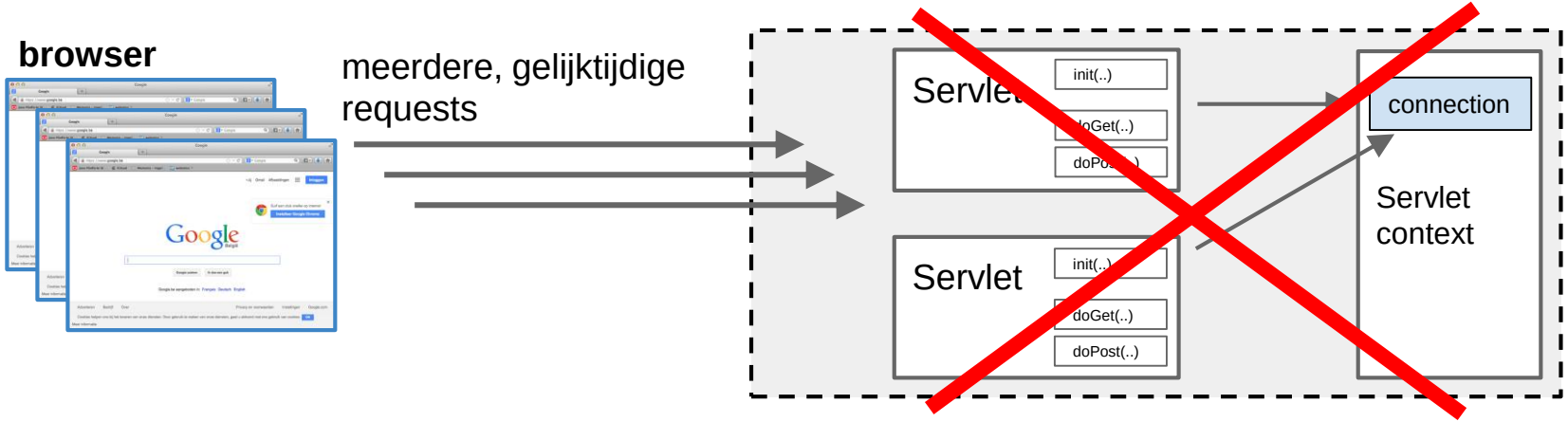
# Hoe Connections bewaren ?

**Als een servlet context attribuut ???**



# Hoe Connections bewaren ?

**Als een servlet context attribuut ???** **OOK NIET !**




De webserver gebruikt dan terug met meerdere gelijktijdige threads dezelfde Connection. En...

Dit mag niet: **een Connection is niet thread-safe !**

# Hoe Connections bewaren ?

**Als een session attribuut  dan maar ???** **Helaas neen...**

Je kunt enkel objecten, waarvan de class de interface Serializable implementeert, in een session attribuut bewaren.

**Connection implementeert Serializable niet.**

**De oplossing: Een DataSource**

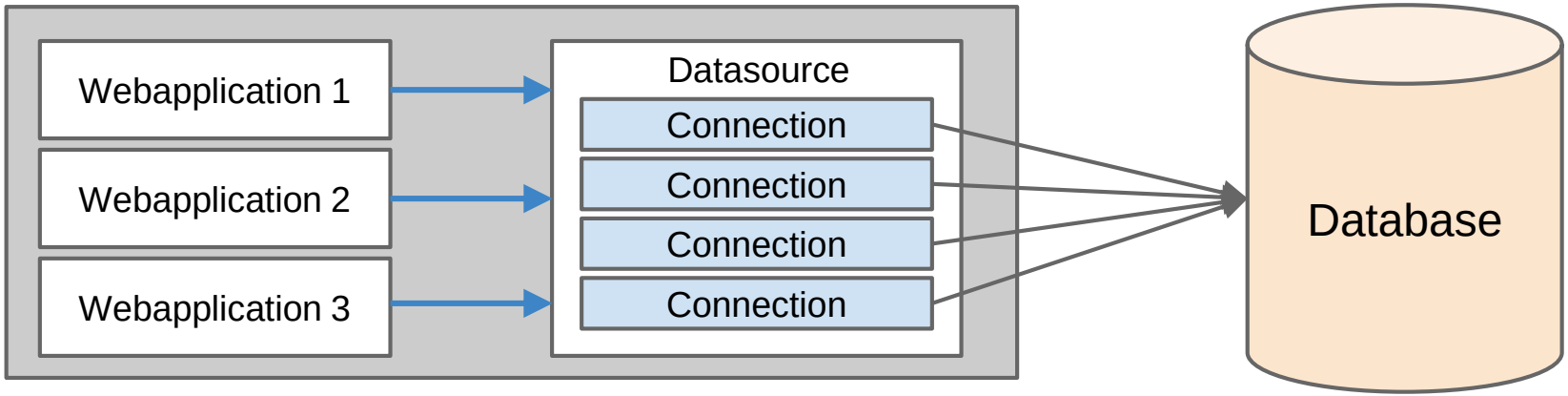
# DataSource

# Wat is een DataSource ?

## Wat is een DataSource?

Een **connection pool**

- Houdt connecties naar de database open ten dienste van
  - één website (**Local DataSource**)
  - of meerdere (**Global DataSource**)

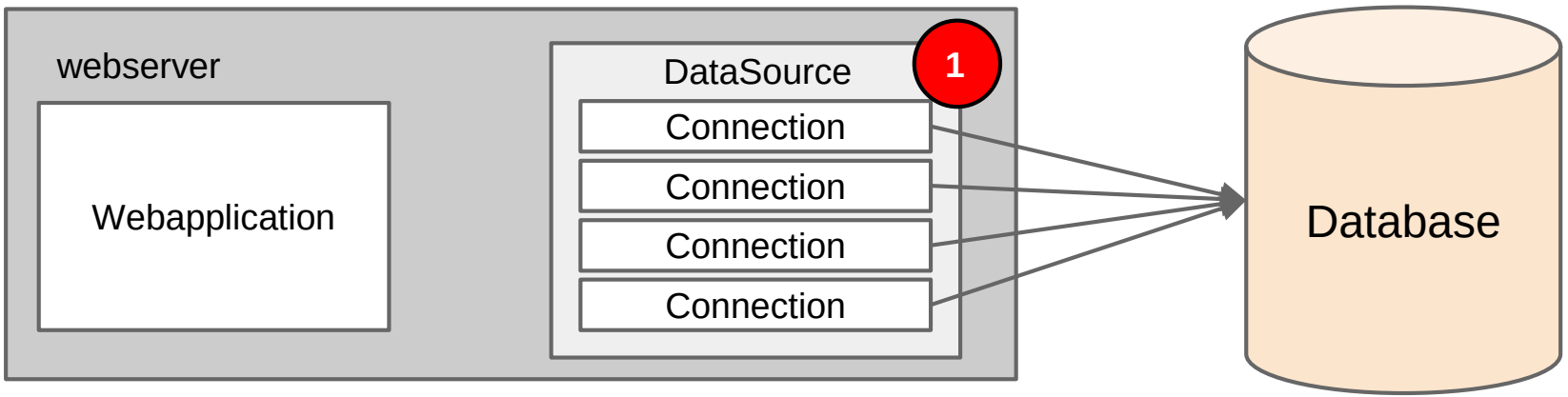


# Wat is een DataSource ?

## DataSource

- Levert aanzienlijke snelheidswinst.
- Een DataSource is een **threadsafe object** waarvan de class de interface ***javax.sql.DataSource*** implementeert.
- Eén database per DataSource !

# Werking van een DataSource

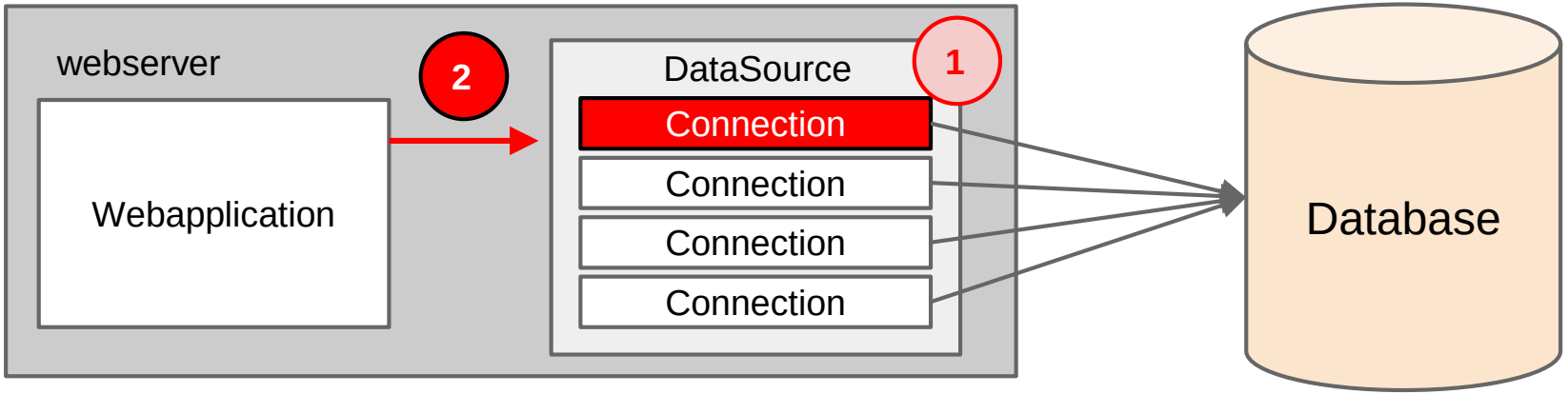



1

Wanneer de webserver de website start,

- maakt hij de DataSource aan
- opent enkele Connection objecten naar de database
- en onthoudt deze Connection objecten in de DataSource

# Werking van een DataSource

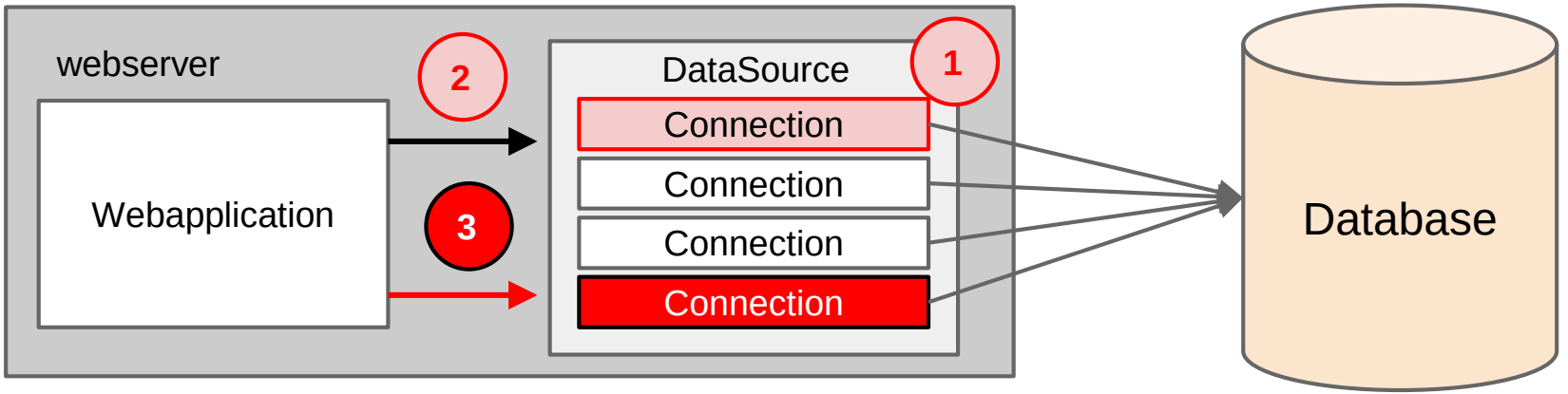



2

Een browser request komt binnen…

- De webserver voert met een thread je code uit. Je vraagt daarin een connectie.
- De webserver geeft je die connectie en onthoudt dat deze nu in gebruik is.

# Werking van een DataSource

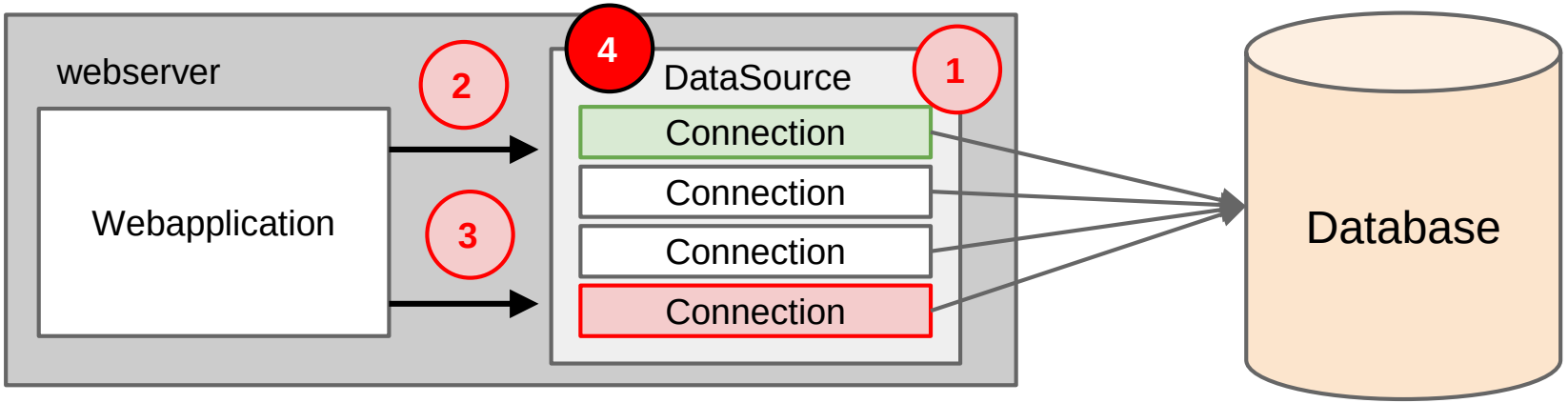



3

Een andere browser request komt binnen…

- De webserver voert met een thread je code uit. Je vraagt daarin terug een connectie.
- De webserver geeft je een vrije connectie en onthoudt dat deze nu in ook gebruik is.

# Werking van een DataSource




4

Je sluit in de code, uitgevoerd door de eerste thread, de Connection.

- De DataSource onderschept dit sluiten en laat de Connection naar de database open
- De DataSource onthoudt dat deze Connection niet meer in gebruik is

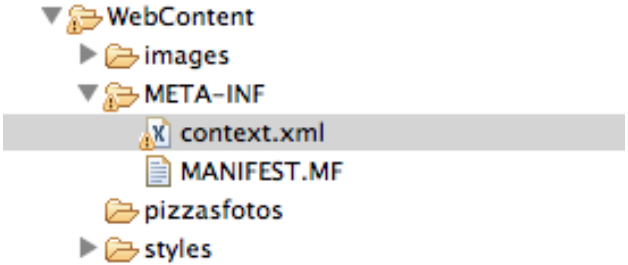
# Een local DataSource definiëren

# Een local DataSource definiëren

Een DataSource wordt gedefinieerd in een XML bestand. De naam, plaats en inhoud van dit bestand zijn per type webserver anders.

**Bij Tomcat**

- is de naam van het bestand context.xml
- is de locatie de folder META-INF van je website



# context.xml

```xml
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE Context>
<Context>
    <Resource
    name="jdbc:/datasources/PizzaDS"
    auth="container"
    type="javax.sql.DataSource"
    maxActive="100"
    maxIdle="30"
    maxWait="1000"
    username="javaexperts"
    password="j@v@x1617"
    driverClassName="com.mysql.jdbc.Driver"
    url="jdbc:mysql://javaexperts.be:3306/javaexpertsDB">
    </Resource>
</Context>
```

- Als je de gebruikersnaam en paswoord vermeldt in context.xml, tik je bij **auth** het woord **Container**. Een minder gebruikt alternatief is de gebruikersnaam en paswoord mee te geven in de Java code wanneer je een Connection vraagt aan de DataSource. In dat geval tik je **Application** bij auth.