JSP & Servlets

Servlets

Leerstof voor vandaag

Include en forward
Multithreading
Cookies

Include en forward

Include

Resultaat van andere resource insluiten (jps, html, servlet …)

→ REQUESTDISPATCHER

1. request.getRequestDispatcher(“/welkom”);
2. context.getRequestDispatcher(“/welkom”);
3. context.getNamedDispatcher(“WelkomServlet”); → Mapping niet nodig!

Forward

Verzoek doorsturen naar andere resource.

→ Methode forward() van REQUESTDISPATCHER.

Er kan extra info via attributen toegevoegd worden.

Meestal op het einde van de methode.

Redirect

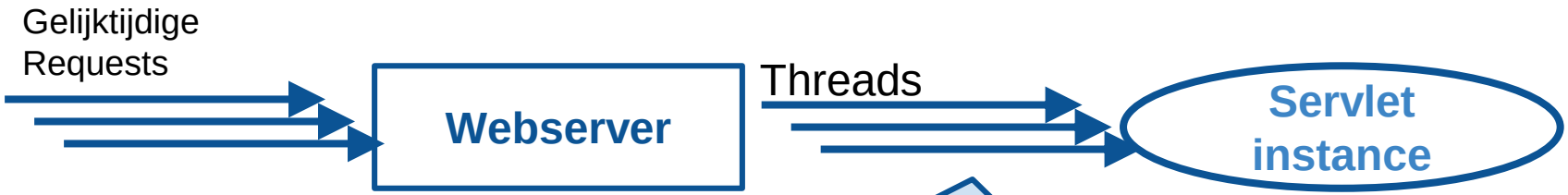
Verzoek doorsturen naar andere resource op andere URL.

BV: Servlet doorverwijzen naar URL op andere webserver.

→ Bv: response.sendRedirect(“http://www.google.be”);

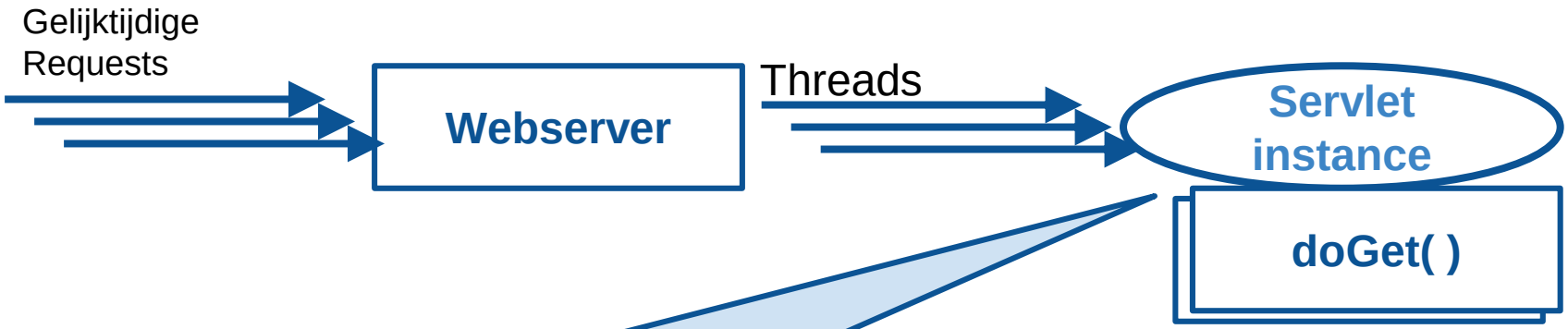
Multithreading

Algemeen



Als een webserver meerdere gelijktijdige requests binnenkrijgt, gebruikt hij meerdere gelijktijdige threads om deze requests te verwerken

Algemeen



Als eenzelfde servlet deze request verwerkt, roepen deze threads gelijktijdig op één servlet instance de method doGet of doPost op

Algemeen

Code in de servlets moet threadsafe zijn !

Is threadsafe:

- Request en response **parameters** van doGet() en doPost()
- **Lokale variabelen** in doGet() en doPost()

Is threadsafe unsure:

- **Instance variabelen** (slechts threadsafe als final primitieve type variabele)
- Objecten waarvan de Class niet threadsafe is

→ Op te lossen door het gebruik van het sleutelwoord **synchronized**.

Instance variabelen

Instance variabelen die

- een primitief datatype hebben
- én voorzien zijn van het sleutelwoord `final`

zijn **wel thread safe**.

Je kunt final variabelen enkel initialiseren:

- bij hun declaratie

  (bvb. `private final int GETAL= 666;`)
- of in de servlet constructor

*De webserver voert beide initialisaties altijd met **één** thread uit.*
Daarna kunnen andere threads deze variabelen lezen, niet wijzigen.

synchronized

Instance variabelen die

- een primitief datatype hebben
- én NIET voorzien zijn van het sleutelwoord `final`

zijn **NIET thread safe**.

Je moet ze daarom benaderen vanuit een method met het sleutelwoord **synchronized**

Terwijl één thread een ***`synchronized`*** *method aanspreekt, plaatst Java andere threads die dezelfde method willen aanspreken op pauze,*
tot de eerste thread de method heeft uitgevoerd.

synchronized

Instance variabelen kunnen een object zijn:

Threadsafe of niet → afhankelijk van de class

voorbeeld:

- StringBuffer → thread-safe
- StringBuilder → niet thread-safe

Benaderen vanuit synchronized method.

Een eigen class thread-save maken

Hoe maken we een eigen class thread-save?

Twee mogelijkheden:

1. **De class immutable maken**
2. **De class voorzien van synchronized methods**

Een eigen class thread-save maken

Immutable class

- Instance variabelen **final** en **private** maken
- via één of meerdere constructors deze initialiseren
- andere methodes kunnen dan enkel deze variabelen lezen, dus GEEN setters voorzien.
- Laat niet toe dat subclasses de methods kunnen overschrijven. (Maak de class final)

```
public class ServletInfo {
     private final Date servletInstanceGestart;

    public ServletInfo() {
                    servletInstanceGestart = new Date();
    }

    public long getLevensduurServletInstance() {
         Date nu = new Date();
          return nu.getTime() - servletInstanceGestart.getTime();


     }
}
```

Een eigen class thread-save maken

mutable class

- De methods die wijzigingen aanbrengen aan de instance variabelen voorzien van het keyword **synchronized**

```
public class ServletInfo {
    private int aantalGetRequests;
    private int aantalPostRequests;

    public synchronized int verhoogAantalGetRequests() {

         return aantalGetRequests++;
    }

     public synchronized int verhoogAantalPostRequests() {
         return aantalPostRequests++;

     }
}
```

Entities

- Worden meestal als **lokale variabelen** gedeclareerd.
- Mag ook als **servlet instance variabele** worden gedeclareerd, op voorwaarde dat:
  - De methods doGet of doPost deze variabele niet wijzigt
  - De entity gedurende de levensduur van de website niet wijzigt.

Cookies

Algemeen

Wat is een cookie?

= Een stukje data dat de browser bijhoudt van een website.

Voorbeelden

- gebruikersnaam
- voorkeursinstellingen

Structuur

- Bestaat uit twee strings: **naam** en **waarde**
- Geen vreemde tekens toegelaten. De vertaling kan gebeuren door een **URLEncoder** en een **URLDecoder**
- maximum 4 KB (4096 bytes) groot
- maximum 20 per website

Soorten

Soorten

Tijdelijke cookies

- worden opgeslagen in het interne geheugen
- Als de browser wordt afgesloten dan is de data niet meer beschikbaar

Permamente cookies

- worden op de harde schijf van de gebruiker opgeslagen
- hebben een vervaltijdstip. Na dit tijdstip verwijdert de browser de cookie

Werking

Werking

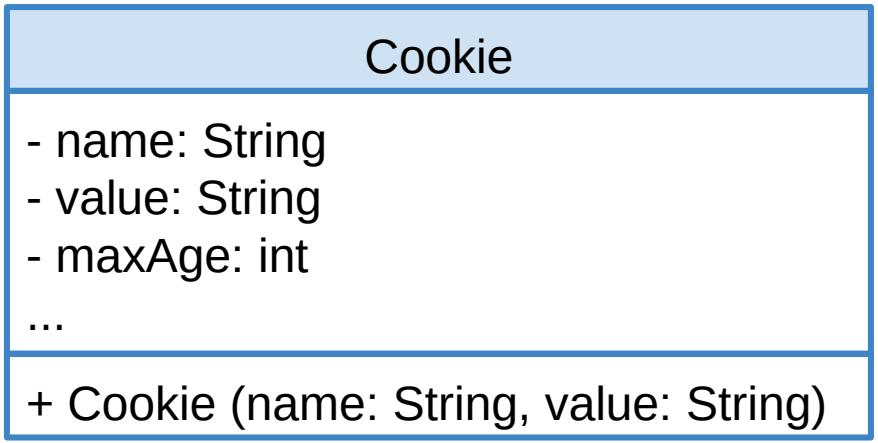
- De browser stuurt bij iedere request alle cookies die horen bij de website mee in de request header met de naam **Cookie**
- Als een website een cookie wil toevoegen, wijzigen of verwijderen, geeft hij dit aan in de response header **Set-Cookie.**

Opmerking

- Stop geen vertrouwelijke info in een cookie
- De gebruiker kan het gebruik van cookies uitzetten

De class Cookie

De Class **Cookie** in de package **javax.servlet.http** stelt een cookie voor



- Alle attributen (name, value, maxAge) hebben getters en setters.
- Een Cookie met een **maxAge gelijk aan - 1** is een tijdelijke cookie.
- Een Cookie met een **positieve waarde** in **maxAge** is een **permanente cookie**.
- **maxAge** geeft dan aan na hoeveel seconden de cookie verdwijnt

Meerdere eigenschappen

Eigenschap	Omschrijving
Name	Iedere cookie heeft een naam
Value	De inhoud van een cookie. Dit is steeds in de vorm van een string.
Comment	Commentaar m.b.t. de cookie
Domain	Cookies worden door de browser enkel verzonden naar het domein waartoe ze behoren. Daarom bezit iedere cookie de naam van het domein.
Path	Cookies worden enkel verzonden naar een bepaald pad van een domein.
MaxAge	Een cookie heeft een bepaalde levensduur, uitgedrukt in seconden.
Secure	Beveiligde cookies kunnen vertrouwelijke informatie bevatten en mogen enkel over een beveiligde verbinding naar de server gestuurd worden (HTTPS)

Een cookie toevoegen

Een cookie voeg je op volgende manier toe:

1. Maak een **Cookie object** met de Cookie constructor.
   Je geeft de **naam** en de **waarde** van de cookie mee als parameters.

Voorbeeld 1

```
Cookie cookie = new Cookie(“naam”, “Patrick”);
```

Voorbeeld 2

```
Cookie cookie = new Cookie(“naam”,
    URLEncoder.encode(request.getparameter(“naam”, “UTF-8”));
```

(De parameter “naam” bevat misschien vreemde tekens → verboden)

Een cookie toevoegen

2

De methods **doGet** en **doPost** hebben een **response parameter**.
Voert op die response parameter de method **addCookie** op en geef daarbij het Cookie object mee als parameter.

Voorbeeld

Cookie cookie = new Cookie(“naam”, “Patrick”);
response.addCookie(cookie)

Cookies lezen

Je kunt **niet één** bepaalde cookie lezen, maar **wel álle** cookies, op volgende manier:

- De methods doGet en doPost hebben een **request parameter.**
- Voer op die request parameter de method **getCookies** uit.
- De method geeft je alle cookies als **een array van Cookie objecten**.

Voorbeeld

```
        if (request.getCookies( ) != null) {
            for (Cookie cookie:request.getCookies( )) {
                    …
        }
}
```

Afzonderlijke cookies lezen

Opgelet!

Je kan niet één afzonderlijke cookie lezen met een specifieke naam.

→ Itereren over de Cookie[ ] array en daarbij telkens de cookie naam checken:

```
Cookie[] cookies = request.getCookies();

String userId = null;
for(Cookie cookie : cookies){
    if("uid".equals(cookie.getName())){
        userId = cookie.getValue();
    }
}
```

Cookies lezen en wijzigen

Opmerkingen

Een cookie die je maakt in één servlet van je website kan je in **álle** servlets en JSP’s van dezelfde website lezen en wijzigen:

- Gebruik de method **addCookie** van de response.
- Geef een Cookie object mee.
- Dit Cookie object bevat de naam van de te wijzigen cookie en de gewijzigde waarde voor de cookie.