

WEB-COMPONENTEN

---

**SERVLETS**

### WAT IS HET?

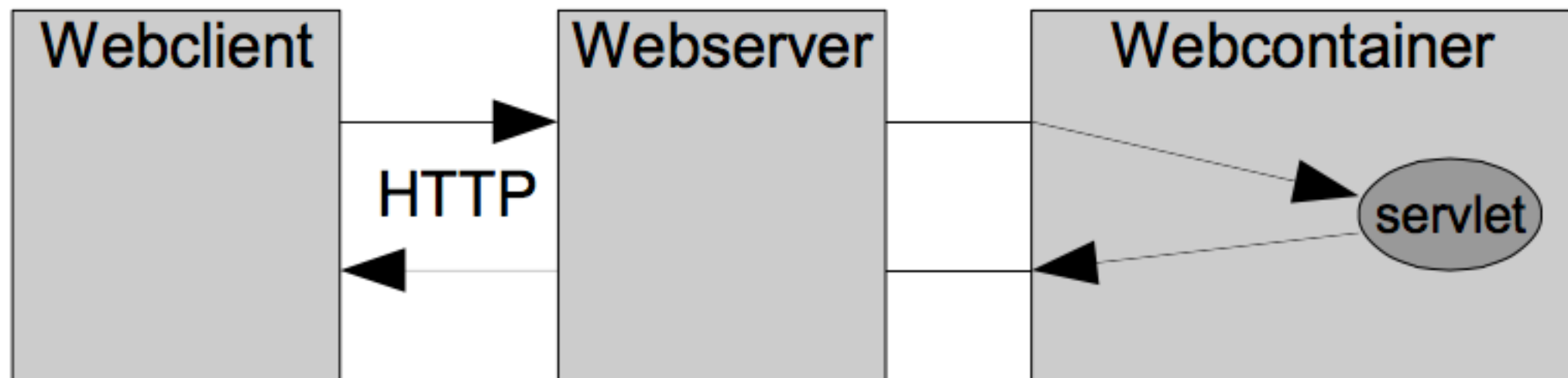
- ▶ onderdeel van een web-applicatie
  - ▶ dus géén aparte context (geen op zich staande applicatie)
  - ▶ een web-applicatie kan dus 1 of meerdere servlets bevatten
- ▶ maakt gebruik van diensten aangeboden door de web-container, bvb communicatie via HTTP
- ▶ kleine “programma’s” binnen een web-applicatie

### VOORBEELD

Een web-applicatie waar studenten hun studentenkaart kunnen aanvragen bevat bijvoorbeeld servlets voor

- ▶ aanmaken van een studentenkaart in pdf
- ▶ aanmaken van de lijst van alle studenten in excel
- ▶ bewaren van de door de student opgeladen pasfoto in het systeem

## WAAR?

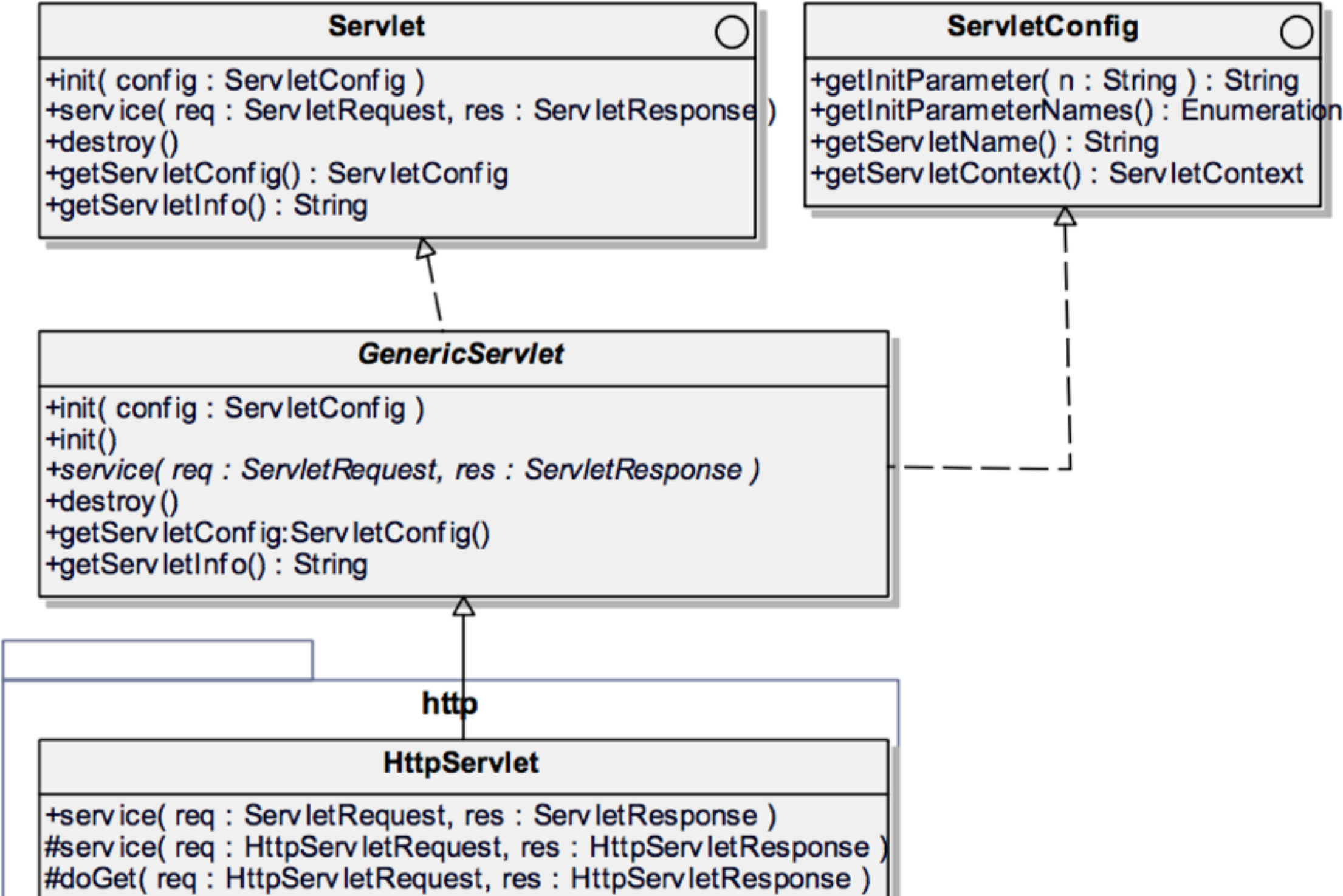


# STRUCTUUR

---

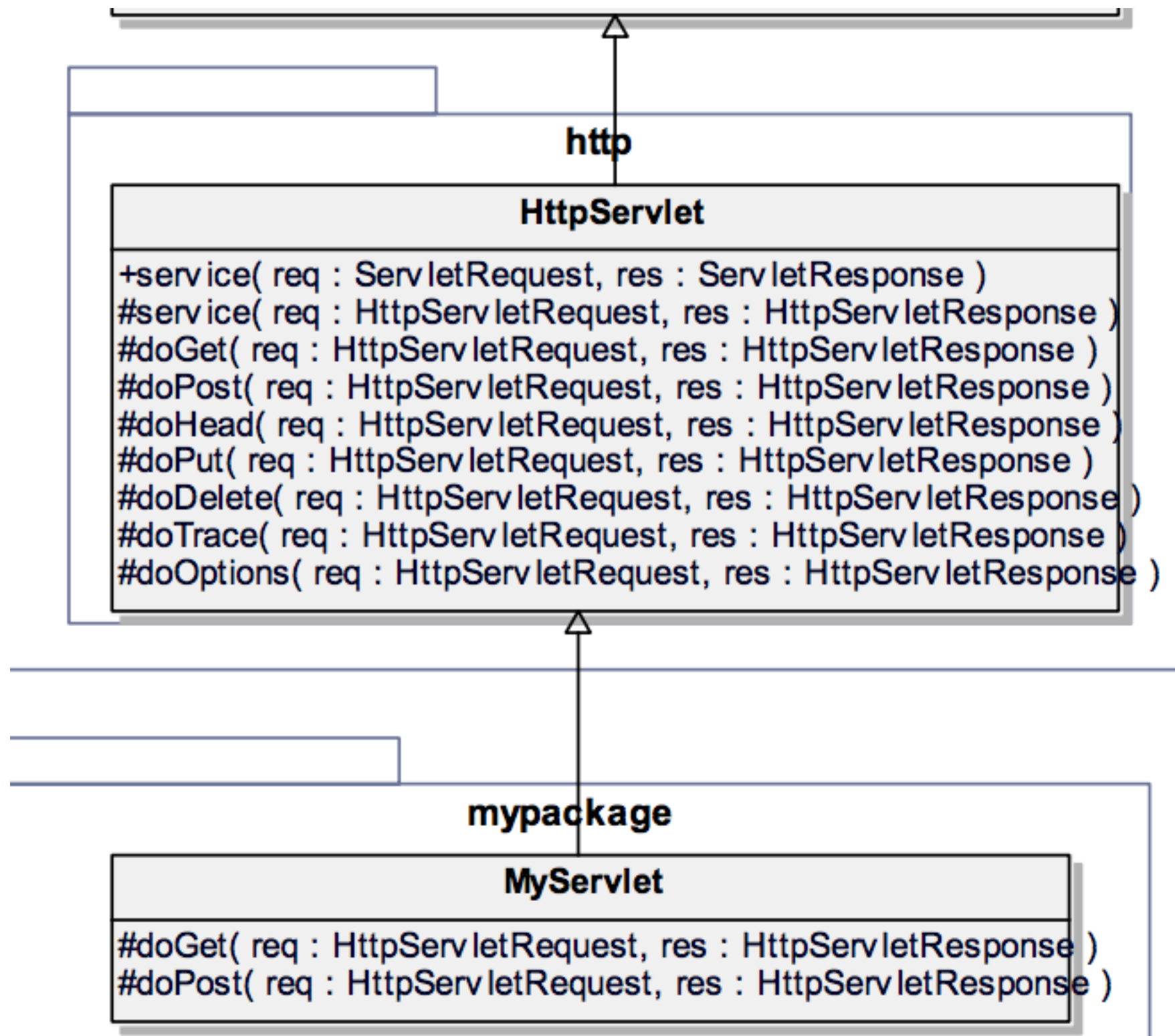
# SERVLETS

## javax.servlet



# SERVLETS

---



### HOE?

- ▶ implementeer interface `javax.servlet.Servlet`
- ▶ convenience: extend **GenericServlet** en implementeer enkel de `#service` methode (voor alle andere methodes voorziet deze abstracte class een implementatie). Een **GenericServlet** kan je implementeren voor gelijk welk protocol
- ▶ convenience: extend **HttpServlet** en implementeer enkel de methodes voor de HTTP operaties (PUT, GET, POST,...) die je wil ondersteunen.



# 'HELLO WORLD' SERVLET

Deze servlet maakt een html pagina die de tekst "hello world" naar je browser stuurt

Stappen:

1. code schrijven
2. servlet configureren
3. web-applicatie opstarten
4. servlet aanroepen

## 'HELLO WORLD' SERVLET

---

```
import java.io.*;
import javax.servlet.annotation.*;
import javax.servlet.http.*;

@WebServlet("/HelloWorld")
public class HelloWorldServlet extends HttpServlet {
    @Override
    protected void doGet(HttpServletRequest request,
        HttpServletResponse response) throws IOException {
        response.setContentType("text/html");
        response.setCharacterEncoding("UTF-8");
        try (PrintWriter out = response.getWriter()) {
            out.println("<!DOCTYPE html>");
            out.println("<html>");
            out.println("<head>");
            out.println("<title>Hello World Servlet</title>");
            out.println("</head>");
            out.println("<body>");
            out.println("Hello World");
            out.println("</body>");
            out.println("</html>");
        }
    }
}
```


## 'HELLO WORLD' SERVLET - OUTPUT

---

```
<!DOCTYPE html>
<html>
<head>
<title>Hello World Servlet</title>
</head>
<body>
Hello World
</body>
</html>
```

## 'HELLO WORLD' SERVLET - BENODIGDE CLASSES

---



```
<dependency>
  <groupId>javax</groupId>
  <artifactId>javaee-web-api</artifactId>
  <version>7.0</version>
  <scope>provided</scope>
</dependency>
```

Je kan servlets configureren via **annotaties** of via de **deployment descriptor(web.xml)**.

In ons voorbeeld `@WebServlet("/helloworld")` geeft de annotatie het relatieve pad aan ten opzichte van de web-app context.

Stel dus dat we een web-app met naam **studenten.war** hebben en we deployen deze op de webserver [www.pxl.be](http://www.pxl.be) dan is ons servlet bereikbaar via [www.pxl.be/studenten/helloworld](http://www.pxl.be/studenten/helloworld)

## 'HELLO WORLD' SERVLET - CONFIGURATIE VIA ANNOTATIES

---

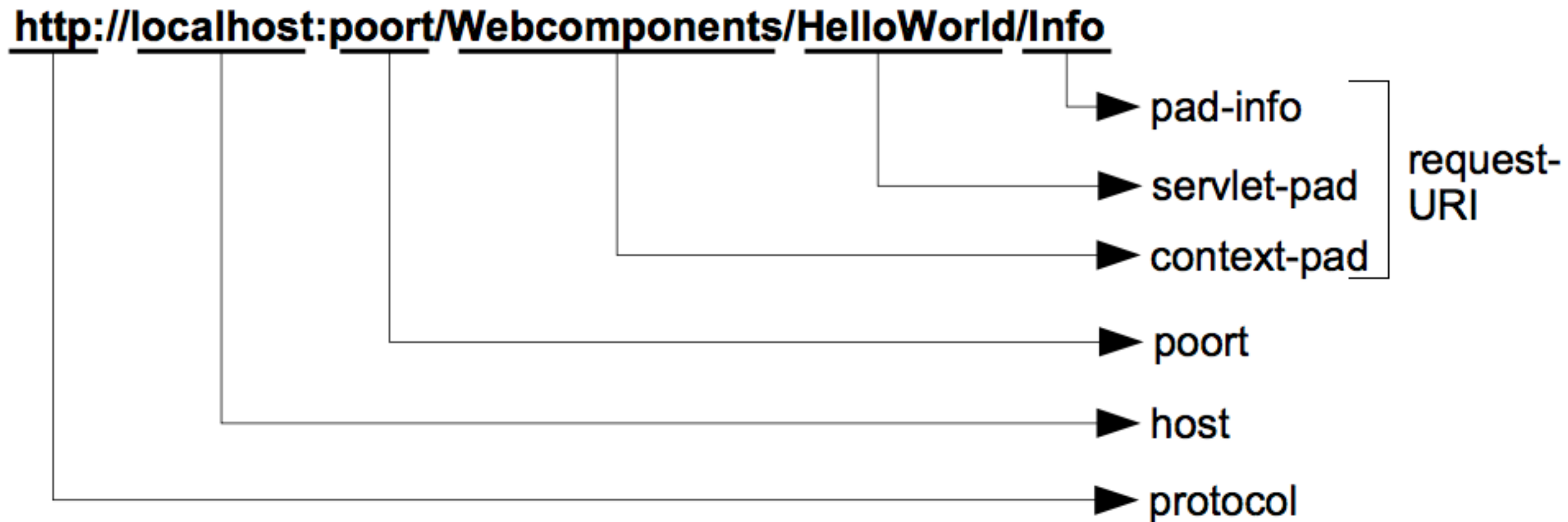
De `@WebServlet` annotatie kan geconfigureerd worden met volgende parameters.

<b><i>Element</i></b>	<b><i>Omschrijving</i></b>
<code>asyncSupported</code>	Geeft aan of asynchrone afhandeling door deze <i>servlet</i> ondersteund wordt.
<code>description</code>	Een optionele beschrijving.
<code>displayName</code>	Een optionele naam die getoond kan worden door de webcontainer.
<code>initParams</code>	Een lijst van initialisatieparameters.
<code>largeIcon</code>	Grote afbeelding.
<code>loadOnStartup</code>	De opstartvolgorde.
<code>name</code>	Naam van de <i>servlet</i> .
<code>smallIcon</code>	Kleine afbeelding.
<code>urlPatterns</code>	Lijst van URL-patronen.
<code>value</code>	Lijst van URL-patronen.

## 'HELLO WORLD' SERVLET - CONFIGURATIE VIA WEB.XML

---

Voor de configuratie van een servlet via de web.xml zijn er tags die equivalent zijn aan de attributen van de annotatie

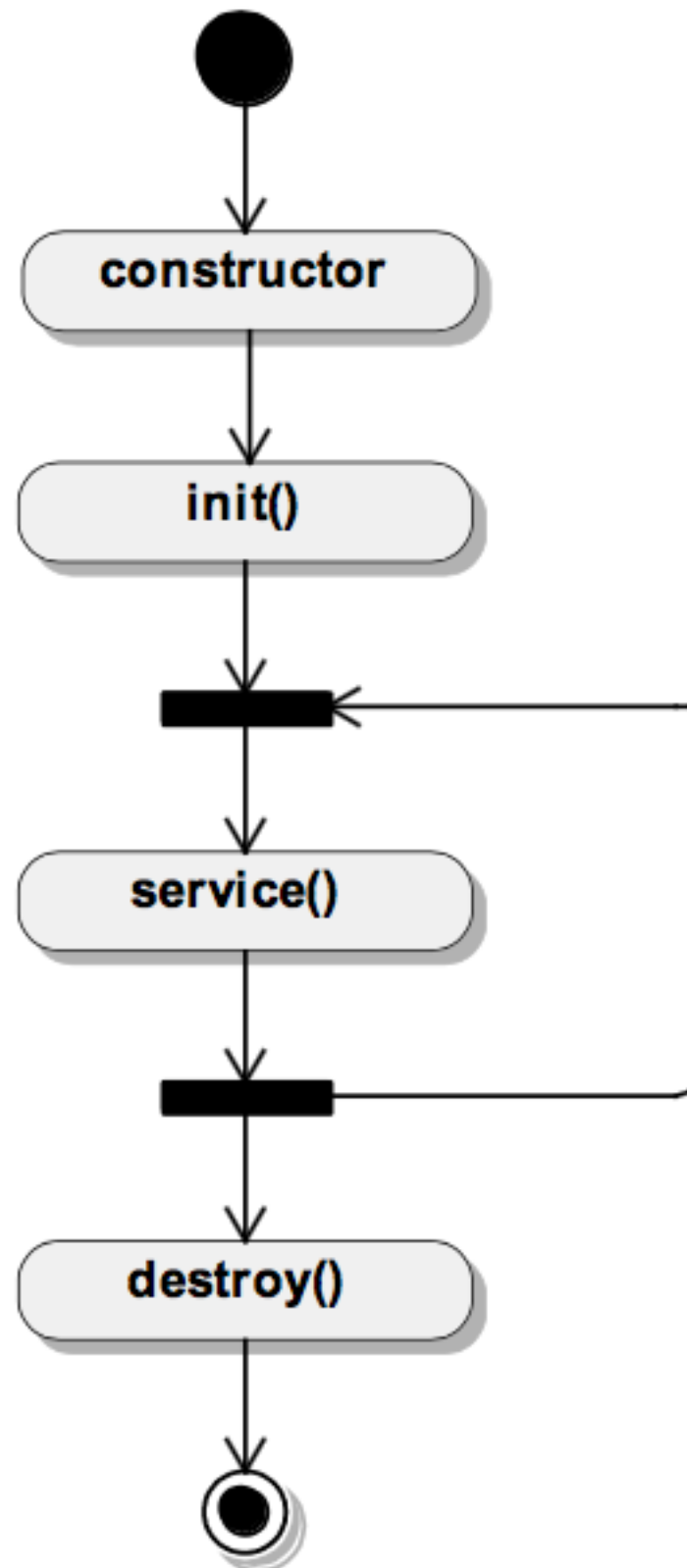




Indien het patroon enkel '/' bevat, dan wordt er naar het default servlet gerouteerd.

## SERVLETS - LIFECYCLE

---





## SERVLETS - INIT PARAMS (VIA ANNOTATIES)

---

```
package eu.noelvaes.web;
....

@WebServlet(name="HelloServlet", value="/HelloWorld",
            initParams=@WebInitParam(name="text",value="Hello World"))
public class HelloWorldServlet extends HttpServlet {
    private String text;

    @Override
    public void init() throws ServletException {
```

```
        text = getInitParameter("text");
        if (text == null)
            throw new ServletException("Parameter text not found");
    }

    @Override
    protected void doGet(HttpServletRequest request,
        HttpServletResponse response) throws IOException {
        response.setContentType("text/html");
        response.setCharacterEncoding("UTF-8");
        try (PrintWriter out = response.getWriter()) {
            out.println("<!DOCTYPE html>");
            out.println("<html>");
            out.println("<head>");
            out.println("<title>Hello Servlet</title>");
            out.println("</head>");
            out.println("<body>");
            out.println(text);
            out.println("</body>");
            out.println("</html>");
        }
    }
}
```

## SERVLETS - INIT PARAMS (VIA WEB.XML)

---

Es gibt eine Reihe von Parametern, die über die folgende XML-Datei...

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://xmlns.jcp.org/xml/ns/javaee"
xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
http://xmlns.jcp.org/xml/ns/javaee/web-app_3_1.xsd" version="3.1">

    <servlet>
        <servlet-name>HelloServlet</servlet-name>
        <servlet-class>
            eu.noelvaes.web.HelloWorldServlet
        </servlet-class>
        <init-param>
            <param-name>text</param-name>
            <param-value>Hello Mars</param-value>
        </init-param>
    </servlet>
</web-app>
```

## SERVLETS -OPGEPAST!

---

Tevens moet men er rekening mee houden dat *servlets* steeds *multithreaded* zijn; simultane verzoeken aan dezelfde *servlet* worden door verschillende *threads* tegelijkertijd afgehandeld. Dit heeft tot gevolg dat *instance variabelen* van de *servlet* door meerdere *threads* tegelijkertijd gebruikt kunnen worden. Indien nodig, dient men de toegang tot deze variabelen te synchroniseren.

## SERVICE METHODES

HttpServlet delegeert door op basis van het HTTP-verzoek

Deze methodes hebben 2

parameters: request en response

- HttpServletRequest
- HttpServletResponse

<b>Methode</b>	<b>HTTP verzoek</b>
<code>doDelete()</code>	DELETE
<code>doGet()</code>	GET
<code>doHead()</code>	HEAD
<code>doOptions()</code>	OPTIONS
<code>doPut()</code>	PUT
<code>doTrace()</code>	TRACE
<code>doPost()</code>	POST

# SERVICE METHODES

- ▶ Opvragen van parameters adhv #getParameter
  - ▶ bvb. <http://www.pxl.be/administratie/student?naam=Jefke>
- ▶ Antwoord genereren
- ▶ Header instellen (bvb. text/html, maar ook image/jpeg, ..)
- ▶ Het bekomen van een OutputStream of PrintWritter van de response, om daar dan je op antwoord te schrijven



# GET

```
import java.io.*;
import javax.servlet.http.*;

@WebServlet("/Echo")
public class EchoServlet extends HttpServlet {
    protected void doGet(HttpServletRequest request,
        HttpServletResponse response) throws IOException {
        // Step 1 : get parameters
        String text = request.getParameter("text");
        // Step 2: generate content

        // Step 3: set headers
        response.setContentType("text/html");
        response.setCharacterEncoding("UTF-8");

        // Step 4: get PrintWriter
        try(PrintWriter out = response.getWriter()) {
        // Step 5: write content
        out.println("<!DOCTYPE html>");
        out.println("<html><head><title>Echo Servlet");
        out.println("</title></head><body>");
        out.println(text);
        out.println("</body></html>");
        }
    }
}
```

# GET (VIA FORMULIER)

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8" />
<title>Echo</title></head>
<body>
<form method="GET" action="Echo">
Enter the text:
<input type="text" name="text" />
<input type="submit" value="Submit" />
</form>
</body>
</html>
```

## POST ('ALTIJD' VIA FORMULIER)

```
<html>
<head>
<meta charset="utf-8" />
<title>Addition</title>
</head>
<body>
<form method='POST' action='Addition'>
Enter two numbers:
<input type='text' name='number1'>
<input type='text' name='number2'>
<input type='submit' value='Calculate'>
</form>
</body>
</html>
```

## SERVLETS (MANUEEL) TESTEN VIA POSTMAN

---

installeer de Chrome browser plugin <https://www.getpostman.com/>

PENDING !!!!!