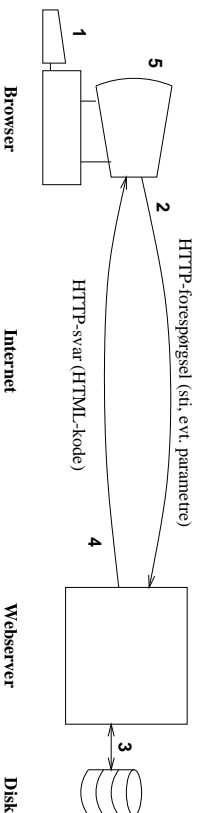


Projektperiode: forelæsning tirsdag 28. november 2000 Java-servletter på webserveren

- Kommunikation mellem browser og webserver (HTTP)
- CGI-skript: dynamisk generering af HTML-kode på webserveren
- Java servletter til dynamisk generering af HTML-kode
- To typer HTTP-forespørgsler: GET og POST
- Servletter der benytter databaser via Java Database Connectivity (JDBC)

Kommunikation mellem browser og webserver (HTTP, Hyper-Text Transfer Protocol)

1. Brugeren indtaster en URL (Uniform Resource Locator) i browseren, f.eks.
`http://www.it-c.dk/courses/GP/E2000/projekter.html`
2. Browseren sender en HTTP-forespørgsel (request) til webserveren `www.it-c.dk`
3. HTTP-forespørgslen indeholder websidens sti / `courses/GP/E2000/projekter.html`
4. Webserveren læser websiden fra disk og sender den HTML-kode som HTTP-svar (response) til browseren.
5. Browseren fortolker HTML-kodens mærker (tags), og viser HTML-siden på skærmen.



Statiske HTML-sider

En klassisk hjemmeside består blot af statisk HTML: en tekstfil der indeholder HTML-kode:

```
<html><head><title>Home Page for Peter Sestoft</title></head><body>
... bla bla bla ...
</body></html>
```

Dynamisk genererede HTML-sider

I stedet for at læse HTML-koden fra disk kan webserveren kalde et program der genererer HTML-koden.

Den genererede HTML-kode sendes til browseren som normalt. Browseren kan ikke se forskel.

Formularer (HTML-mærket FORM) i HTML-tekster

Dynamisk generering af HTML-sider bruges ofte i forbindelse med formularer (HTML-mærket <FORM>).

Formularer kan bruges til at sende data (tekster, filer, osv) til webserveren.

Så kalder webserveren et program der læser formularens data og genererer en HTML-side som svar.

CGI-skript

Programmer på webserveren der genererer HTML-kode kaldes *CGI-skript* (CGI = Common Gateway Interface).

CGI-skript kan skrives i alle mulige programmeringssprog: Perl, C, Standard ML, Tcl, Visual Basic, Python, ...

På alle større websteder genereres HTML-siderne automatisk af CGI-skript ved udtræk fra en database.

Det er alt for besværligt at vedligeholde hundrede (eller hundredund) individuelle statiske HTML-sider.

CGI-skript i Java: servletter

CGI-skript i Java, dvs. Java-programmer der kører på webserveren og genererer HTML, kaldes *servletter*.

Java-servletter udføres af en Java-fortolker (virtuel maskine) som er integreret med webserveren.

Det er effektivt og sikkert at integere Java 'skript' i webserveren på denne måde.

Apache JServ er et modul til Apache-webserveren (som er gratis og den mest udbredte).

Der findes adskillige andre Java-webservere, f.eks. Sun Microsystems JavaWebServer.

Java-servletter benyttes f.eks. af Internet-boghandlen `www.stop4you.com`

Java-servletter, teknisk

En servlet programmeres som en subklasse af den abstrakte klasse `HttpServlet` fra pakken `javax.servlet.http`.

For at fungere skal en servlet implementere en eller flere af disse metoder:

```
public void doGet(HttpServletRequest req, HttpServletResponse res) ...
public void doPost(HttpServletRequest req, HttpServletResponse res) ...
public void service(HttpServletRequest req, HttpServletResponse res) ...
```

Servlettens inddata og uddata

Inddata, altså servlettens argumenter fra webbrowseren, kommer fra `HttpServletRequest`-objektet `req`:

Metodekaldet `req.getParameterValues(fild)` returnerer en `String`-tabel:

Tabellen indeholder de argumenter der er bundet til servlettens parameter `fild`.

Resultater er `null` hvis servletten ikke fik nogen parameter kaldet `fild`.

Uddata, altså servlettens svar til webbrowseren, sendes via `HttpServletResponse`-objektet `res`.

Med `out = res.getOutputStream()` får man et `ServletOutputStream`-objekt `out`.

Dette objekt bruges til at sende HTML-kode tilbage til webbrowseren som svar på forespørgslen.

Man sender HTML-kode ved at kalde `out.println("<HTML><BODY>")` eller lignende.

En HTML-formular der kalder en servlet som lægger to tal sammen

```
<HTML><HEAD><TITLE>Call servlet to add numbers</HEAD>
<BODY><H1>Form that calls a Java servlet to add numbers</H1>
&ltP><FORM ACTION="http://ellomose.dina.kvl.dk/servlet/AdditionServlet"
METHOD=GET>
&ltP><First number to add: <INPUT TYPE=TEXT NAME=number1>
&ltP><Second number to add: <INPUT TYPE=TEXT NAME=number2>
&ltP><INPUT TYPE=SUBMIT VALUE="Add the numbers">
</FORM>
...
</BODY>
</HTML>
```

Den servlet der modtager tallene og beregner deres sum

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

public class AdditionServlet extends HttpServlet {

    public void doGet(HttpServletRequest req, HttpServletResponse res)
        throws ServletException, IOException {
        res.setContentType("text/html");
        ServletOutputStream out = res.getOutputStream();

        out.println("<HTML><BODY>");
        out.println("We are pleased to inform you that the result " +
            "of your calculation is now ready:");
        String[] number1params = req.getParameterValues("number1");
        int number1 = Integer.parseInt(number1params[0]);
        String[] number2params = req.getParameterValues("number2");
        int number2 = Integer.parseInt(number2params[0]);
        int sum = number1 + number2;
        out.println("<P>You kindly sent us the numbers " + number1 +
            " and " + number2);
        out.println("<P>The sum of the numbers is " + sum);
        out.println("</BODY></HTML>");
    }
}
```

Hvordan virker det?

Når man trykker på 'Add the numbers' i formularen sker der følgende:

- Formularen sendes til webserveren `ellomose.dina.kvl.dk`
- På webserveren startes servleten `AdditionServlet`
- Servletten læser formularens felter `number1` og `number2`
- Servletten genererer HTML-koder
- HTML-koderne sendes fra webserveren til browseren
- Browseren viser HTML-siden

Man kan også kalde `doGet` med en URL af formen

```
http://ellomose.dina.kvl.dk/servlet/AdditionServlet?number1=67&number2=99
```

Her gives servlettens parametre i forlængelse af servletnavnet, efter spørgsmålstegnet.

Det kendes fra forespørgsler til søgemaskiner mv.

Typen af HTTP-forespørgsler: GET og POST

En forespørgsel af type GET er gentagelig; den svarer nogenlunde til læsning fra serveren.

- Den ændrer ikke på serveren (ved at skrive i databaser eller lignende).
- Gentagne kald af servletten er 'uskedelige'.

En forespørgsel af type POST har typisk sideeffekter, eller er ikke-gentagelig:

- Den kan ændre på serveren (ved at skrive i databaser eller lignende)
- Et kald af servletten vil typisk svare til en transaktion.

Føks. køb af en bog, betaling af en regning. Det er jo ikke ligegyldigt om man gør det én eller to gange.

Hvis en servlet indeholder metoden doGet så vil den svare på GET-forespørgsler.

Hvis en servlet indeholder metoden doPost så vil den svare på POST-forespørgsler.

Hvis en servlet indeholder metoden service så vil den svare på både GET og POST forespørgsler.

En servlet bør ikke svare på GET-forespørgsler med mindre den er fri for sideeffekter på webserveren.

Livsforløbet for servletten Countrequests

1. Servlettens .class-fil lægges i et bestemt katalog på webserveren (f.eks. /usr/share/java/servlets/)
2. Filen Countrequests.class læses ind i webserveren, der skabes et Countrequests-objekt ud fra klassen Countrequests, og objektets init-metode kaldes
3. Servletten Countrequests kan nu modtage og besvare forespørgsler fra webbrugere ved hjælp af metoderne doGet eller service
4. Når webserveren lukkes ned, så kaldes servlettens destroy-metode

Der laves ét eksemplar af en given servlet, dvs. ét objekt af en given servlet-klasse.

Trådsikkerhed: flere samtidige kald

Hvis servletten kaldes samtidig fra flere webbrugere kan der være flere aktive kald til servletmetoderne samtidig.

Hvert kald til servlettens doGet, doPost- eller service-metode kører i sin egen tråd.

Hvis disse metoder læser og ændrer felter i objektet (eller filer på disk), så bør metoderne være *trådsikre*.

Metoderne skal erklæres med `synchronized public void service(...)`

Man kan også bare lade servlet-klassen implementere klassegrænsefladen SingleThreadModel.

En servlet der ved hvor mange gange den er blevet kaldt

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

public class Countrequests
    extends HttpServlet implements SingleThreadModel {

    private int howmanyrequests = 0;

    public void service(HttpServletResponse req, HttpServletResponse res)
        throws ServletException, IOException {
        res.setContentType("text/html");
        ServletOutputStream out = res.getOutputStream();

        out.println("<HTML><BODY>");
        out.println("This servlet has been called " + howmanyrequests + " times.");
        howmanyrequests++;
        out.println("</BODY></HTML>");
    }
}
```

En HTML-formular der bestiller en gangetabel

```
<HTML><HEAD><TITLE>Call servlet to create table</TITLE></HEAD>
<BODY>
<H1>Form that calls a Java servlet</H1>

<P><FORM ACTION="http://ellernose.dina.kvl.dk/servlet/TableServlet"
    METHOD=GET>

    <BR>Choose a color:
    <SELECT NAME=COLOR>
    <OPTION VALUE=red>Red
    <OPTION VALUE=green>Green
    <OPTION VALUE=blue>Blue
    </SELECT>

    <P>Choose a table size: <INPUT TYPE=TEXT NAME=lines>
    <INPUT TYPE=SUBMIT VALUE="Make table">
    </FORM>
    . . .
</BODY>
</HTML>
```

Den genererede HTML-kode (for 3 og grøn)

```
<HTML><BODY>
  A servlet-generated multiplication table:
<P><TABLE BORDER><TR ALIGN=RIGHT><TD>
<TH BGCOLOR=green>1
<TH BGCOLOR=green>2
<TH BGCOLOR=green>3
<TR ALIGN=RIGHT><TH BGCOLOR=green>1<TD>1<TD>2<TD>3
<TR ALIGN=RIGHT><TH BGCOLOR=green>2<TD>2<TD>4<TD>6
<TR ALIGN=RIGHT><TH BGCOLOR=green>3<TD>3<TD>6<TD>9
</TABLE>
</BODY></HTML>
```

Den interessante del af servleten

```
out.println("A servlet-generated multiplication table:");
String[] lineparams = req.getParameterValues("lines");
int lines = Integer.parseInt(lineparams[0]);
if (lines > 50) // Do not make tables larger than 50 by 50
    lines = 50;
String[] colorparams = req.getParameterValues("color");
String color = colorparams[0];
out.println("<P><TABLE BORDER><TR ALIGN=RIGHT><TD>");
for (int i=1; i <= lines; i++)
    out.println("<TH BGCOLOR=" + color + ">" + i);
for (int i=1; i <= lines; i++) {
    out.print("<TR ALIGN=RIGHT><TH BGCOLOR=" + color + ">" + i);
    for (int j=1; j <= lines; j++)
        out.print("<TD>" + (i*j));
    out.println();
}
out.println("</TABLE>");
```

Den servlet der opbygger gangetabellen

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

public class TableServlet extends HttpServlet {

    public void service(HttpServletRequest req, HttpServletResponse res)
        throws ServletException, IOException {
        res.setContentType("text/html");
        ServletOutputStream out = res.getOutputStream();

        out.println("<HTML><BODY>");

        ... den interessante del af servleten er her ...

        out.println("</BODY></HTML>");
    }
}
```

Servlet der tilføjer tekst til fil på webserveren

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

public class SkrivServlet
    extends HttpServlet implements SingleThreadModel {
    final static String filnavn = "/tmp/testfil";

    public void doPost(HttpServletRequest req, HttpServletResponse res)
        throws ServletException, IOException {
        ServletOutputStream out = res.getOutputStream();

        out.println("<HTML><BODY>");
        String[] tekst = req.getParameterValues("tekst");
        if (tekst != null && tekst.length >= 1) {
            Writer wri = new FileWriter(filnavn, true); // append-mode
            PrintWriter outfile = new PrintWriter(wri);
            outfile.println(tekst[0]);
            outfile.println("-----");
            outfile.close();
            wri.close();
            out.println("Teksten blevet tilføjet til filen " + filnavn);
        } else
            out.println("Der er ingen tekst?");
        out.println("</BODY></HTML>");
    }
}
```

Servlet der læser tekst fra fil på webserveren

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

public class LaessServlet extends HttpServlet {

    final static String filnavn = "/tmp/testfil";

    public void doGet(HttpServletRequest req, HttpServletResponse res)
        throws ServletException, IOException {
        res.setContentType("text/html");
        ServletOutputStream out = res.getOutputStream();

        out.println("<HTML><BODY><PRE>");
        Reader inp = new FileReader(filnavn);
        BufferedReader binp = new BufferedReader(inp);
        String line = binp.readLine();
        while (line != null) {
            out.println(line);
            line = binp.readLine();
        }
        binp.close();
        inp.close();
        out.println("</PRE></BODY></HTML>");
    }
}
```

Fhøjskolen

Grundlæggende Programmering, efterår 2000

Side servlet-17

HTML-formular til at skrive og læse fra tekstfil på webserveren

```
<HTML><HEAD><TITLE>Formular til servlet som skriver til en fil</TITLE></HEAD>
<BODY>
<H1>Formular til servlet som skriver til en fil</H1>
<P>
<FORM ACTION="http://ellemose.dina.kvl.dk/servlet/SkrivServlet"
      METHOD=POST>
<P>Teksten der skal tilføjes til filen:
<BR>TEXTAREA NAME=tekst ROWS=5 COLS=80>Skriv her</TEXTAREA>
</FORM>
<P><FORM ACTION="http://ellemose.dina.kvl.dk/servlet/LaessServlet"
      METHOD=GET>
<INPUT TYPE=SUBMIT VALUE="Læs hele filen">
</FORM>
... <A HREF="http://ellemose.dina.kvl.dk/servlet/LaessServlet">...</A>
</BODY>
</HTML>
```

Fhøjskolen

Grundlæggende Programmering, efterår 2000

Side servlet-18

En server for tilfældige tal

```
public class Randomservlet extends HttpServlet implements SingleThreadModel {

    private Random rgen = new Random();
    private int[] outcome = new int[1000];
    private int howmanyrequests = 0;

    public void service(HttpServletRequest req, HttpServletResponse res)
        throws ServletException, IOException {
        res.setContentType("text/html");
        ServletOutputStream out = res.getOutputStream();

        out.println("<HTML><BODY>");
        out.println("<H2>Random number server</H2>");
        int result = (int)(1 + 6 * rgen.nextDouble());
        outcome[howmanyrequests] = result;
        howmanyrequests++;
        out.println("The outcome was <B>" + result + "</B>.");
        out.println("<P>The die has been thrown " + howmanyrequests + " times.");
        out.println("<P>The previous results were:");
        out.println("<P><TABLE BORDER=<TR>TH>Throw<TH>Outcome");
        for (int i=0; i<howmanyrequests; i++)
            out.println("<TR><TD align=right>" + (i+1) + "<TH>" + outcome[i]);
        out.println("</TABLE>");
        out.println("</BODY></HTML>");
    }
}
```

Fhøjskolen

Grundlæggende Programmering, efterår 2000

Side servlet-19

Servlet der kommunikerer med en database-server

```
public class ReadDBServlet extends HttpServlet {

    ...
    public void init(ServletConfig config) throws ServletException
    { ... /* Åbn forbindelsen til databasen når servlet-objektet skabes */ }
    public void destroy()
    { ... /* Luk forbindelsen til databasen når servlet-objektet nedlægges */ }

    public void doGet(HttpServletRequest req, HttpServletResponse res)
        throws ServletException, IOException {

        ...
        try {
            String query = "SELECT * FROM student ORDER BY fornavn";
            st.execute(query);
            out.println("<HTML><BODY><TABLE>\n<H2>Tabellen Student</H2><P>");
            out.println("<TR><TH>Fornavn<TH>Efternavn<TH>Studienr");
            ResultSet rs = st.getResultSet();
            while (rs.next()) {
                String enavn = rs.getString("efternavn");
                String fnavn = rs.getString("fornavn");
                String stdnr = rs.getString("studienummer");
                out.print("<TR>");
                out.println("<TD>" + fnavn + "<TD>" + enavn + "<TD>" + stdnr);
            }
            out.println("</TABLE></BODY></HTML>");
        } catch (SQLException e) { System.out.println("Database error: " + e); }
    }
}
```

Fhøjskolen

Grundlæggende Programmering, efterår 2000

Side servlet-20

Ekspirimeter med servletter: Java Servlet Development Kit (JSDK) og servletterrunner

Man kan hente JSDK fra `http://java.sun.com/products/servlet/download.html`
JSDK skal pakkes ud så man får et katalog `JSDK2.0` eller lignende.

I underkataloget `JSDK2.0\bin` findes en `servletterrunner` til eksperimenter med servletter.

Man behøver ikke være på nettet for at bruge `servletterrunner`, men PC'en skal nok være netværks-konfigureret.

I underkataloget `JSDK2.0\examples` findes nogle `servlet-eksempler`.

Start `servletterrunner` ved at skrive `JSDK2.0\bin\servletterrunner`

Start en browser og skriv `http://localhost:8080/servlet/SimpleServlet`, så køres den simpleste eksempel-servlet.

Prøv også `http://localhost:8080/servlet/SnoopServlet`

Du kan lægge dine egne eksempler (f.eks. `RandomServlet`) ned i `JSDK2.0\examples`

Derefter kan du køre dem med `http://localhost:8080/servlet/RandomServlet`

Se også `http://java.sun.com/docs/books/tutorial/servlets/servletterrunner/`

Mere om Java-servletter

Servlet-eksemplerne ligger på adressen:

```
http://www.dina.kvl.dk/~sestofc/programmering/servlets/
```

Java-servletterne udlæses af Apache JServ, som er et modul til Apache webserveren.

Suns servlet-hjemmeside:

```
http://java.sun.com/products/servlet/
```

Java servlet API (klasser og metoder der kan bruges i servletter)

```
http://www.dina.kvl.dk/~sestofc/javaservlets/apidoc/packages.html
```

På IT-højskolen kan studerende køre servletter på maskinen `mysql.it-c.dk`

Vi opretter en separat servlet-zone til hver projektyrups. Vejledning findes (snart) på GP-projekt-hjemmesiden.

Ultrakort om HTML

```
http://www.dina.kvl.dk/~sestofc/databehandling/html1.html
```

```
http://www.dina.kvl.dk/~sestofc/databehandling/html2.html
```