F8
Webbteknologier 1

EDA095 Nätverksprogrammering

Roger Henriksson
Datavetenskap
Lunds universitet

Dynamiska webbsidor

- HTML är statisk. En sida får sitt utseende bestämt när en webbdesigner skapar den.
- Ofta vill man ha mera dynamiska webbsidor:
  - Svar på en databasförfrågan.
  - Konstant uppdaterade webbsidor.
  - Dialog med användaren.
  - Animeringar.
  - Kontroll av inmatad information i ett formulär.

Serversidan eller klientsidan?

Klientsidan

- Applets
- JavaScript
- Flash

Serversidan

- CGI – CommonGateway Interface
- JSP (Java Server Pages) och Servlets
- ASP – Active Server Pages
- PHP – "PHP:Hypertext Preprocessor"

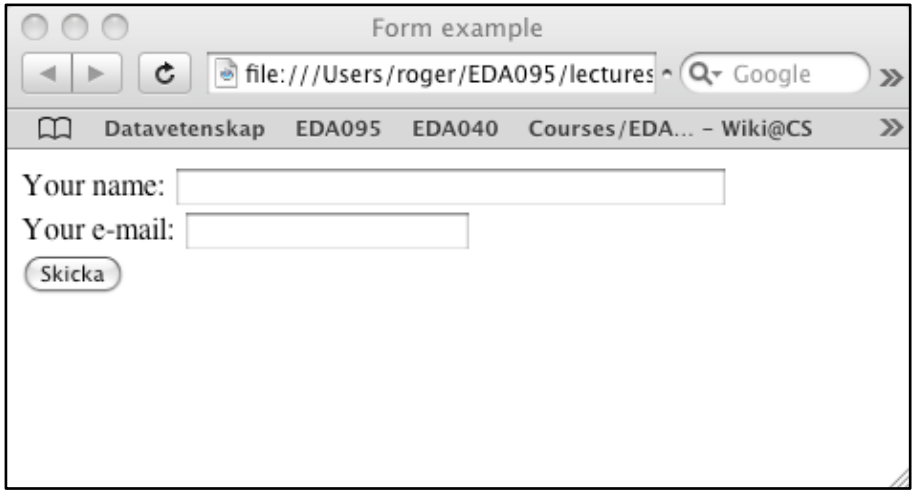
Varför inte en helt specialskriven webbserver?

Common Gateway Interface

1. När webbservern får en begäran om en webbsida med en särskild URL startar servern ett externt program – ett “CGI-skript”.
2. Det externa programmet läser in eventuella parametrar i form av en “query string” antingen via standard input eller s.k. “environtment-variabler”.
3. Programmet genererar en HTML-sida baserat på parametrarna och skriver HTML-koden till standard output.
4. Programmet avslutas.

HTML och formulär

```
<html>
<head><title>Form example</title></head>
<body>
<form method="get" action="/cgi-bin/storeaddress.pl">
Your name: <input name="name" type="text" size=40>
<br>
Your e-mail: <input name="email" type="text" size=20>
<br>
<input type="submit">
</form>
</body>
</html>
```



HTTP-förfrågan

Föregående exempel genererar en TCP-uppkoppling till servern och en HTTP-förfrågan sänds:

```
GET /cgi-bin/storeaddress.pl?name=Roger+Henriks
son&email=roger%40cs.lth.se HTTP 1.0
```

1. Serven startar skriptet “storeaddress.pl”.
2. Frågesträngen (“query string”) överförs via en environment-variabel.

POST istället för GET

```
POST /cgi-bin/storeaddress.pl HTTP 1.0
Content-type: application/x-www-form-urlencoded
Content-length: 49

name=Roger+Henriksson&email=roger%40cs.lth.se
```

Begäran består av ett huvud (avslutat med dubbla radslut (CR+LF+CR+LF) och en frågesträng.

CGI-skriptet läser frågesträngen via standard input.

Lämpligt för stora datamängder.

Frågesträngen syns ej i URL:en.

Svar

CGI-skriptet skriver till standard output:

1. MIME-typ, typiskt “Content-type: text/html”.
2. Blankrad
3. HTML-kod för den genererade sidan.

Exempel

```
Content-type: text/html

<html>
<head><title>Registration completed</title></head>
<body>
<h1>Registration completed</h1>
Roger Henriksson (roger@cs.lth.se) has been added to the
database.
</body>
</html>
```

CGI – Exempel

Skriv ett CGI-skript som visar temperaturdata.

DEMO – temperatured.cgi

CGI: fördelar och nackdelar

Fördelar

- Möjlighet att välja mellan många olika implementationsspråk.
- Väl beprövat och allmänt tillgängligt.

Nackdelar

- Ineffektivt: startar en ny operativsystemprocess för varje HTTP-begäran.
- Måste avkoda frågesträngen själv.
- Besvärligt att bevara tillstånd – måste spara på disk.

Servlets

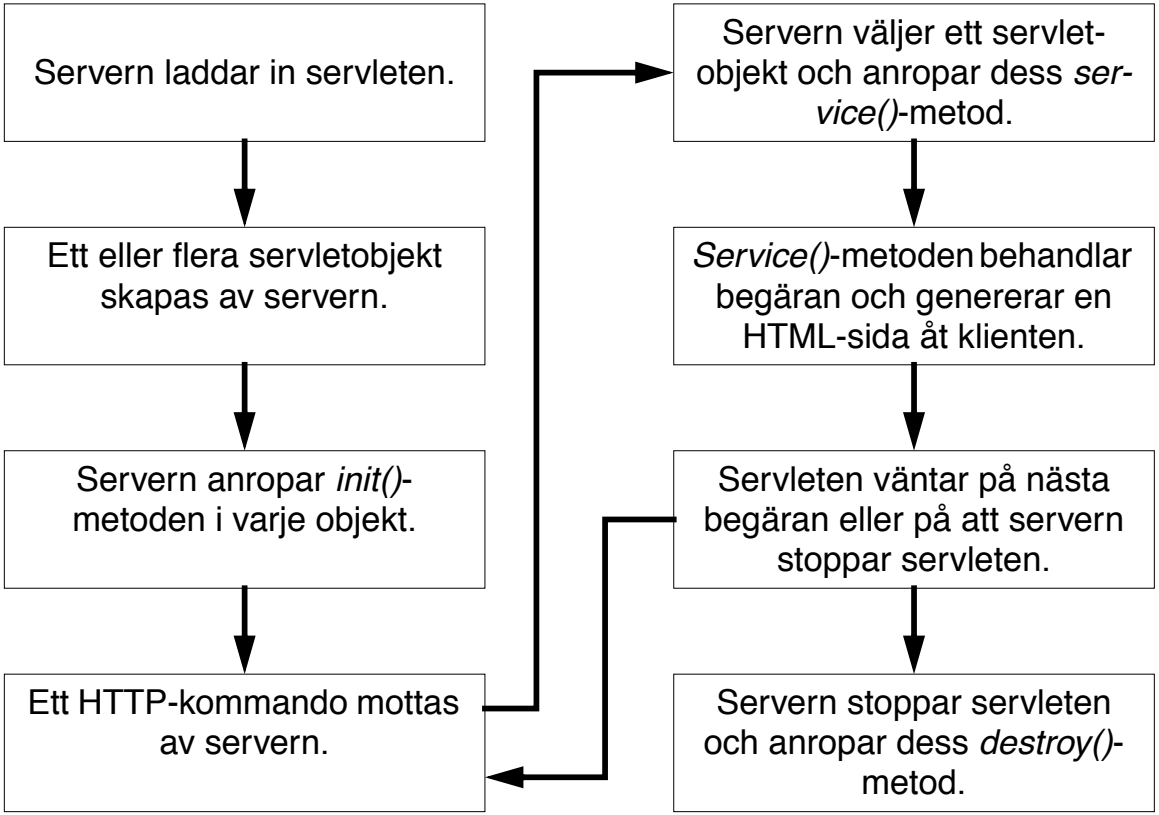
Servlet?

Applet - “liten applikation”, Servlet - “liten server”

Som CGI, fast:

- Skrivna i Java.
- Systemoberoende.
- Skapar inte ny operativsystemsprocess varje gång. Effektivare!
- Startar inte om för varje HTTP-begäran. Kan komma ihåg information från gång till gång.

En servlets liv



Klassöversikt

Paket

```
import javax.servlet.*;       // Standard i J2EE
import javax.servlet.http.*; // Standard i J2EE
import java.io.*;
```

Klasser/interface

- HttpServlet – superklass för webbservlets.
- HttpServletRequest – klientens HTTP-begäran.
- HttpServletResponse – servletens svar till klienten.
- ServletConfig – information om servern.

HttpServlet

En servlet instansieras av webbservern.

Initialisering

Implementera en av nedanstående:

```
public void init(ServletConfig config)
                                    throws ServletException;
public void init();
```

Terminering

```
public void destroy();
```

Implementera denna för att till exempel stänga databasuppkopplingar/stänga öppna filer när servleten avslutas av servern.

service()

När servern tar emot ett HTTP-kommando anropas:

```
protected void service(HttpServletRequest request,
                       HttpServletResponse response)
                  throws ServletException, IOException;
```

Denna kan implementeras för att behandla en begäran från klienten.

- request – information om klientens begäran
- response – används för att skicka svar till klienten.

Ofta vill vi göra olika saker beroende på typ av begäran:

```
if (request.getMethod().equals("GET")) {
  ...
} else {
   if (request.getMethod().equals("POST")) {
     ...
```

Alternativ till service()

Standardimplementationen av service() undersöker vilken typ av kommando klienten skickade (GET/POST/HEAD etc) och anropar en av:

```
protected void doGet(HttpServlerRequest request,
                     HttpServletResponse response)
                          throws ServletException,IOException;
protected void doPost(HttpServlerRequest request,
                      HttpServletResponse response)
                          throws ServletException,IOException;
```

Likadant för:

doHead
doPut
doDelete

HttpServletRequest

Information om klientens begäran.

Metoder

`public String getParameter(String name);`
Returnerar värdet för angiven parameter, t.ex. innehållet i ett textfält i ett HTML-formulär.

`public String getRemoteAddr();`

`public String getRemoteHost();`
IP-nummer / namn på klientdatorn.

`public String getMethod();`
Typ av begäran (GET, POST, etc.).

Med flera...

HttpServletResponse

Används för att skicka svar till klienten.

1. Ange MIME-typen för svaret:
   ```
   response.setContentType("text/html");
   ```
2. HTML-koden skickas genom en ström:
   ```
   PrintWriter output = response.getWriter();
   ```
3. Skriv HTML-koden till strömmen.
4. Stäng strömmen:
   ```
   output.close();
   ```

Servlets och HTML-formulär

- I formulär kan man specificera vilken typ av HTTP-kommando som ska användas när formuläret skickas in till servern: GET eller POST. POST vanligast.
- Om man skriver in en URL i adressfältet i en webbläsare används GET.

DEMO – GuessingGame

Tillståndsinformation

Kommunikationen mellan klient och webbserver är ofta en dialog - jmf en webbshop.

Servern måste hålla reda på vad som hänt under dialogen, dvs hålla reda på ett tillstånd.

Tillstånd kan lagras i servleten, men:

Vi måste kunna skilja på olika klienter!

- Gömda fält i formulär

```
<input type="hidden" name="number" value="42">
```

- Cookies
- HttpSession

Klassen Cookie

Namn/värdepar som lagras på klienten.

Paket

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

Konstruktor

```
public Cookie(String name, String value);
```

Metoder

```
public String getName();
public String getValue();
```

Med flera...

Skriva/läsa cookies

Metoder i klasserna HttpServletRequest/HttpServletResponse.

HttpServletRequest

```
public Cookie[] getCookies();
```

Returnerar en vektor med samtliga cookies från denna webbplats.

HttpServletResponse

```
public void addCookie(Cookie cookie);
```

Skriver ner en ny cookie eller ny version av en gammal cookie till klienten.

DEMO – CookieGuess

JSP – Java Server Pages

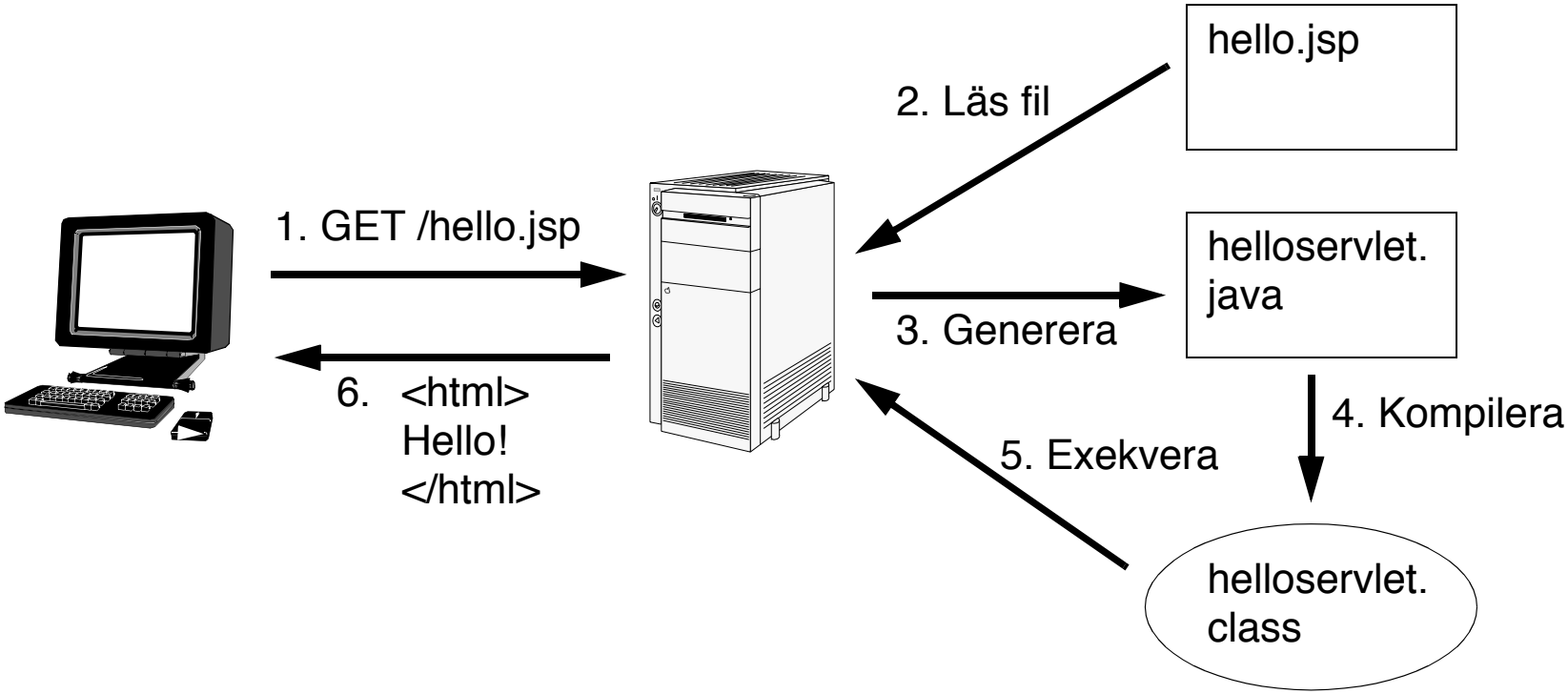
Idé

"Programkoden i HTML-koden" istället för "HTML-koden i programkoden".

Implementation

- En JSP-fil (.jsp) är en HTML-fil med några extra element i.
- JSP-elementen ("tags") anger var dynamisk HTML-kod ska infogas i dokumentet och hur den ska genereras.
- När en klient begär JSP-filen tolkas innehållet av servern och görs automatiskt om till en servlet första gången.

Översättning av JSP-fil



JSP – Tags

Direktiv

Anvisningar för översättningen till servlet. "<%@ ... %>"

Deklarationer

Deklarationer av attribut motsvarande servletattribut. "<%! ... %>"

Uttryck

Anger javauttryck vars värde stoppas in på sidan. "<%= ... %>"

Scriptlets

Block av javakod som exekveras när JSP-sidan anropas. "<% ... %>"

Kommentarer

"<%-- ... -- %>"

JSP – Direktiv

Styr översättningen till en servlet.

"page"

Styr servletens struktur: Importerar externa klasser, ändrar "content type", ändrar servletens superklass.

Exempel:

```
<%@ page import="java.util.*" %>
<%@ page contentType="text/plain" %>
```

"include"

Inkluderar andra JSP-filer vid översättningen.

Exempel:

```
<%@ include file="filetoinclude.jsp" %>
```

JSP – Deklarationer

Används för att deklarera variabler som sedan kan användas i uttryck och i "scriptlets".

Motsvarar attributen i en servletklass.

Exempel:

```
<%! int counter = 0; %>
<%! Date today = new Date(); %>
```

Flera deklarationer kan samlas:

```
<%!
  int counter = 0;
  Date today = new Date();
%>
```

JSP – Uttryck

Används för att stoppa in resultatet av en beräkning eller annat uttryck i HTML-koden.

Exempel:

```
<%= counter %>
<%= today.toString() %>

Pris (inkl. moms): <%= pris*1.25 %> kronor.
```

Kan innehålla godtyckliga javauttryck.

JSP – Scriptlets

Anger Javakod som ska exekveras när sidan hämtas.

Exempel:

```
<%
    total = 0.0;
    for(int i=0;i<myArray.length;i++) {
        total = total+myArray[i];
    }
%>

Average: <%= total/myArray.length %>
```

JSP – Implicita objekt

Ett antal standardobjekt finns alltid tillgängliga utan explicit deklaration.

`request` – HTTP-begäran från klienten.

`response` – HTTP-svaret till klienten.

`session` – HTTPSession-objekt associerat till den aktuella användaren/sessionen.

`application` – Refererar till "globala" objekt som ska delas mellan alla sessioner, t.ex. databasanslutning.

`out` – Objekt som används för att skriva till den utgående svarsströmmen (till klienten).

Med flera...

ServletContext

Ett sätt att dela information mellan servlets/JSP-sidor. Typen för den implicita variabeln "application".

Servlets: **`ServletConfig.getServletContext();`**

```
public void setAttribute(String,Object);
public Object getAttribute(String);
```

Med mera...

Gör det möjligt att dela upp en webbapplikation på flera servlets/JSP-sidor. Dock: Se upp för kapplöpning!

JSP – Exempel

DEMO – guessinggame.jsp