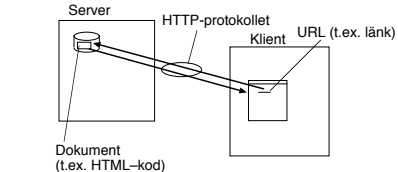


Programmeringsmetodik DV1 Programkonstruktion 1

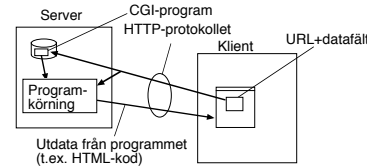
Moment 11 Om webbprogrammering

Webbsidor och webbservrar



Användaren klickar på en länk, servern lokaliserar ett dokument (oftast med HTML-kod) och skickar tillbaka till klienten.
Nackdel: Dokumenten är fasta – fungerar inte om man vill ha webben som gränssyta mot ett program – t.ex. ett bokningsystem.

Common Gateway Interface (CGI)

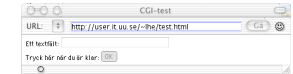


Användaren klickar på en knapp, servern lokaliserar ett CGI-program, kör det och skickar data till klienten.
Fördel: CGI-programmet kan utföra vilken uppgift som helst. Ofta är det en gränssyta mot ett bakomliggande system – bank, bokning...

HTML-formulär

CGI-programmet "anropas" från en webbsida med hjälp av formulär.

```
<form method=POST action="http://user.it.uu.se/~lhe/test.cgi">
  <p>Ett textfält:</p>
  <input type=text name=f1 value=""></p>
  <p>Tryck här när du är klar:</p>
  <input type=submit value="OK"></p>
</form>
```



När knappen trycks in anropas CGI-programmet och informationen i formuläret (textfältet i detta fall) skickas med till det.

Mer om formulär

...måste börja med HTML-taggen `<form>` och sluta med `</form>`.

```
<form method=POST action=URL-till-CGI-prog>
input-taggen definierar ett datafält eller en knapp.
```

```
<input type=typ name=namn value=värde>
```

typ kan vara bl.a.:

- text – Ett fält där man kan skriva in och redigera text.
 - password – Som text, men innehållet är dolt.
 - hidden – Ett textfält som överhuvudtaget inte syns i fönstret.
 - submit – En knapp som anropar CGI-programmet
 - reset – En knapp som återställer startvärden på fälten
- Det finns också menyer, kryssrutor, m.m.
namn är ett namn på fältet (används ej på submit/reset).
värde är ett startvärde för fältets innehåll (etikett på submit/reset).

Mer om textfält

För textfält kan man ange storleken genom att i `input`-taggen ange `size=kolumner`

eller `size=kolumner, rader`.
rader resp. *kolumner* anger storleken på textfältet. T.ex.

```
<input type=text name=f1 value="Hej" size=50,10>
```

...vilket definierar ett textfält med namnet *f1*, 50 tecken brett och 10 rader högt och med starttexten "Hej".

För stora textmängder finns specialtaggen `textarea`.

```
<textarea name=namn rows=rader cols=kolumner>
värde
</textarea>
```

Avläsning av fält i CGI-programmet

http-protokollet överför informationen i formuläret till CGI-programmet i kodad form.
 I ML-biblioteket `Mosmlcgi` finns funktioner för avkodning.
 Den viktigaste är `cgi_field_string`.

```
Mosmlcgi.cgi_field_string f
TYPE: string -> string option
POST: SOME s om f är namnet på ett fält i formuläret, annars NONE.
s är en sträng med innehållet i fältet.
EXAMPLE: cgi_field_string "f1" = SOME "Hej"
         ifall det finns ett fält f1 som innehåller texten "Hej".
         cgi_field_string "f2" = NONE
         om det inte finns ett fält f2.
```

Skapa dokument i CGI-program

Allt som CGI-programmet skriver på strömmen `stdout` skickas till webbläsaren och tolkas där som en webbsida (eller bild eller andra typer av data).

Utskriften *måste* börja med en rad som beskriver typen av data följt av en blank rad. Därefter kommer det "egentliga" webbsideinnehållet.

- Typen av data kan vara t.ex.
- `text/html` HTML-dokument
 - `text/plain` oformatterat textdokument
 - `image/gif` bild i gif-format.

Exempel på utdata från CGI-program

Content-type: text/html

```
<HTML>
<HEAD>
<TITLE>Exempel</TITLE>
</HEAD>
<BODY>
Hej, världen
</BODY>
</HTML>
```



Skapa CGI-program

ML program måste *separat kompileras* för att få körbara filer som fungerar utanför ML-systemet.

Under UNIX kan man ge kommandot:

```
mosmlc MLprogramfil -o körbarfil
t.ex.
mosmlc test.sml -o test.cgi
```

`mosmlc` accepterar bara deklARATIONER (`open`, `fun`, `val` etc.) och inte uttryck. Bibliotek laddas automatiskt – *load får* inte användas!
 Eftersom uttryck är förbjudna får man använda ett trick för att starta programmet. Avsluta filen med:

```
val _ = uttryck;
T.ex.
val _ = mainfunction();
```

Installation av CGI-program

Det kompilerade CGI-programmet måste läggas där webbservern hittar den. På institutionen för IT är det katalogen

```
public_html/cgi-bin
```

i respektive användares toppnivåkatalog, t.ex.

```
~lhe/public_html/cgi-bin
```

Programmet anropas sedan med URLen:
`http://user.it.uu.se/cgi-bin/cgiwrap/ användare/ prog`
 T.ex.
`http://user.it.uu.se/cgi-bin/cgiwrap/lhe/test.cgi`

Spara data i CGI-program

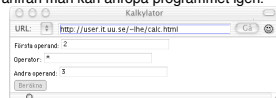
När ett formulär skickas till webbservern så körs CGI-programmet en gång och avslutas sedan. Vill man spara data från en körning av CGI-programmet till en annan finns olika metoder:

- skriva data på en fil som CGI-programmet läser in nästa gång det körs.
- spara data i ett fält (t.ex. `text` eller `hidden`) i ett formulär i webbdokumentet som CGI-programmet skapar.
- med s.k. *cookies* (vilket jag inte tar upp här).

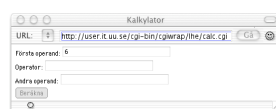
OBS! Om du sparar på fil måste du tänka på att CGI-programmet körs som om du själv var inloggad. Det skriver och läser dina filer med dina rättigheter. Detta kan leda till säkerhetsproblem (t.ex. om användaren kan påverka filnamnen).

Exempel: en webbkalkylator

CGI-programmet anropas ifrån en webbsida, körs och skapar en ny webbsida varifrån man kan anropa programmet igen.



(Klick)



calc.html

`calc.html` inleds med "magisk" text som HTML-standarden kräver skall finnas där:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<HTML>
<HEAD>
<TITLE>Kalkylator</TITLE>
<META http-equiv="Content-Type" content="text/html"; charset=iso-8859-1">
</HEAD>
<BODY>
```

calc.html (forts.)

```
<form method=POST action="http://user.it.uu.se/cgi-bin/cgiwrap/lhe/calc.cgi">
Första operand:
<input type=text name=a value="0"><br>
">
Operator:
<input type=text name=op value=""><br>
Andra operand:
<input type=text name=b value=""><br>
<input type=submit value="Beräkna">
</form>
</BODY>
</HTML>
```

Webbkalkylatorns huvudfunktion

Tänk på att citationstecken (") och bakåtsveck (\) måste föregås av bakåtsveck i ML-strängar (\") och (\\)!)

```
(* calc x
  TYPE: unit -> unit
  SIDE-EFFECTS: Skapar kalkylatorns nya webbsida *)
fun calc() =
  (print("Content-type: text/html\n\n");
  print("<!DOCTYPE HTML PUBLIC \"-//W3C//DTD HTML 4.01
  Transitional//EN\" \"http://www.w3.org/TR/html4/loose.d
  td\">\n");
  print("<HTML>\n<HEAD>\n<TITLE>Kalkylator</TITLE>\n");
  print("<META http-equiv=\"Content-Type\" content=\"te
  xt/html; charset=iso-8859-1\">\n</HEAD>\n");
  print("<BODY>\n");
  print(calcform(cgi_field_integer("a",0),
  cgi_field_string("op"),
  cgi_field_integer("b",0)));
  print("</BODY>\n</HTML>\n"));
```

PK1&PM1 HT-05 moment 11

Side 17

Uppdaterad 2004-10-21

Hjälpfunktioner (1)

```
(* calcform (a,opnd,b)
  TYPE: int*string option*int -> string
  POST: Räknar ut värdet av operationen opnd med
  argumenten a och b. Returnerar ett HTML-formulär
  för kalkylatorn som innehåller resultatet. *)
fun calcform(a,opnd,b) =
  "<form method=POST action=\"http://user.it.uu.se/cgi-b
  in/cgiwrap/lhe/calc.cgi\">\n" ^
  "Första operand: <input type=text name=a value=\"\" ^
  Int.toString(calcresult(a,getOpt(opnd,\"\"),b)) ^
  \"><br>\n" ^
  "Operator: <input type=text name=op value=\"\"><br>\n"
  ^
  "Andra operand: <input type=text name=b value=\"\"><br
  >\n" ^
  "<input type=submit value=\"Beräkna\">\n" ^
  "</form>\n"
```

PK1&PM1 HT-05 moment 11

Side 18

Uppdaterad 2004-10-21

Hjälpfunktioner (2)

```
(* calcresult (a,opnd,b)
  TYPE: int*string*int -> int
  PRE: Om opnd är "/" måste b vara 0
  POST: Värdet av operationen opnd på argumenten
  a och b. Om opnd är en felaktig operation
  returneras istället a. *)
fun calcresult(a,"+",b) = a+b
  | calcresult(a,"-",b) = a-b
  | calcresult(a,"*",b) = a*b
  | calcresult(a,"/",b) = a div b
  | calcresult(a,_, b) = a;
```

PK1&PM1 HT-05 moment 11

Side 19

Uppdaterad 2004-10-21

calc.sml

```
open Mosmlcgi;

fun calcresult ...

fun calcform ...

fun calc ...

val _ = calc();
```

PK1&PM1 HT-05 moment 11

Side 20

Uppdaterad 2004-10-21

Felsökning av CGI-program

CGI-program kan vara svåra att felsöka eftersom de inte går att köra interaktivt.

Tips 1: Dela upp programmet i "webbdel" och "funktionsdel" där den senare inte utnyttjar några CGI-funktioner. Testa och felsök funktionsdelen först.

Tips 2: Om du använt `open Mosmlcgi`; för att slippa skriva `Mosmlcgi`. före cgi-funktionerna, så kan du definiera egna funktioner `cgi_field_string` och `cgi_field_integer` som antingen ger tillbaka konstanta värden eller läser från terminalen. Då kan du testa programmet interaktivt. (Du måste definiera dem *efter* `open Mosmlcgi`; eller ta bort denna deklaration.)

PK1&PM1 HT-05 moment 11

Side 21

Uppdaterad 2004-10-21

Testkod för kalkylatorn

I kalkylatorfallet kan du definiera:

```
fun cgi_field_string("op") = SOME ("*")
  | cgi_field_string(_) = NONE

fun cgi_field_integer("a",_) = 2
  | cgi_field_integer("b",_) = 3
  | cgi_field_integer(_, x) = x
```

så kommer kalkylatorn att försöka räkna ut 2^3 utan att någon webbkommunikation är inblandad. För att prova andra testfall ändrar du helt enkelt på definitionerna.

PK1&PM1 HT-05 moment 11

Side 22

Uppdaterad 2004-10-21