

Database-baseret Web-publicering, efterår 2001

Forelæsning 4, tirsdag den 18. september 2001.

Web-programmering, tilfældige tal, og tilstand

- Grupperinger og indlejrede kommandoer
- Valutæksempel
- Formulærer
- Tekstområder, indtastingsfelter og menuvalg
- Generering af tilfældige tal på web-serveren
- Web-programmer med tilstand

Valutahandel (valutahandel.html)

For at bygge en valutahandelservice kræves kun to filer.

```
<html>
<head>
  <title>Valutahandel</title>
</head>
</body>
<form method=post action=valutaberegning.cgi>
  Hvor mange kroner vil du veksle til dollars? <p>
  <input type=text size=10 name=kroner>
  <input type=submit value="Beregn">
  </form>
</body>
</html>
```

Grupperinger og indlejrede kommandoer

Der er to måder at gruppere argumenter til kommandoer på:

- % set saldo 100  
100
- % puts "Saldoen er \$saldo"  
Saldoen er 100
- %  
Double-quote grupperinger fortolkes inden strengen sendes som argument til kommandoen (vi står en variabel op i lagret)
- % puts {Saldoen er \$saldo}  
Saldoen er \$saldo

Brace-grupperinger sendes ufortolket som argument til kommandoen.

Vikan kun lave indlejrede kommandoer i double-quote grupperinger:

```
% puts "Saldoen er [set saldo]"
Saldoen er 100
% puts {Saldoen er [set saldo]}
Saldoen er [set saldo]
```

Valutahandel - fortsat (valutaberegning.cgi)

```
# set the form-variable 'kroner'
set_form_variables

# compute dollar amount
set dollarkurs 7.34
set dollars [expr ($kroner - 20.0) / $dollarkurs]

ns_return 200 text/html "<html>
<head>
  <title>Valutahandel - resultat</title>
</head>
<body bgcolor=white>
  <center>
    <h1>Valutahandel - resultat</h1>
    For $kroner kroner modtager du $$dollars <p>
    <a href="//hug.itu.dk:8077/F2001/lec3/valutahandel.html">Tilbage</a>
  </center>
</body>
</html>"
```

## Valutahandel - test af inddata

Hvilke problemer er der med valutahandel-serveren?

- Hvad sker der når brugeren indtaster et beløb på mindre end kr. 20?
- Hvad sker der hvis brugeren ikke indtaster et tal?

### Afsending af email fra Tcl

Kommandoen `ns_sendmail` bruges til at sende email fra Tcl programmer på web-serveren:

```
ns_sendmail to from subject body
```

Eksempel (email.tcl):

```
set manglende_afleveringer 7
set oevelse 1
ns_sendmail "nh@it.edu" "kenneth@it.edu" "Oevelse $oevelse" "
Hej Niels,
```

Der er stadig \$manglende\_afleveringer studerende der ikke har afleveret oevelse \$oevelse.

Mvh. Kenneth"

## En procedure `proc` i forhold til andre Tcl-kommandoer

Et Tcl program kan opdeles i en række procedurer.

Vi gør dette, for at genbruge kode og skabe overblik.

En procedure navngiver en række Tcl-kommandoer, f.eks. `puts`, `return`, `for`, `while` og `if`.

En procedure kan indeholde alle Tcl-kommandoer, undtagen kommandoen `proc`.

### Tcl-kode kontra HTML-kode

Vi anvender `pr` to programmeringssprog, Tcl og HTML.

HTML er godt til oplysning af dokumenter. HTML er ikke velegnet til at udføre beregninger og behandle data.

Tcl er velegnet til at udføre beregninger og behandle data. Tcl er ikke velegnet til oplysning af dokumenter.

Derfor, anvender vi HTML til browseren og Tcl på serveren!

Da Tcl-programmer genererer data til browseren, så genererer vi HTML i vores Tcl-programmer – denne sammenblanding kan virke forvirrende.

### `puts` kontra `return`

Nogle retningslinjer:

- En procedure bør slutte med kommandoen `return`
- `puts` giver kun mening for programmer der afvikles i Tcl-fortolkeren – ikke i web-serveren.
- `ns-return` anvendes kun i web-serveren.

## Web-server fejlhåndtering

- Ved "Server Error" ser du en fejlmeddelelse i din browser:

```
http://hug.itu.dk:8077/E2001/lec3/fej1.tcl.
```

- "Uendelig løkke" på web-serveren (uljytkke3.tcl):

```
set i 10
while { $i >= 0 } {
  set i 1
}
```

Genstart web-serveren ved brug af "Web Server Services":

```
http://hug.itu.dk:8002/webserverer.html
```

## Tcl på web-serveren

- Sider med endelsen `.tcl` bliver fortolket af web-serveren
- Tcl-programmer på web-serveren må **ikke** gøre brug af `puts` kommandoen
- I stedet bruges kommandoen `ns-return` til at sende en HTML side tilbage til en browser:

```
ns_return 200 text/html " ... "
```

### Et simpelt web-server program (simple.tcl)

Følgende program returnerer en simpel HTML-side til browseren:

```
set name "Niels Hallenberg"
set email "nh@it-c.dk"
ns_return 200 text/html "<html>
<head>
<title>Home page for $name</title>
<body>
Home page for $name <p> <hr>
<a href=\"mailto:$email\">$email</a>
</body>
</html>"
```

## Formularer

En *HTML-formular* benyttes ved indtastning af oplysninger.

Oplysningerne sendes via nettet til et Tcl program.

Tcl programmet behandler oplysningerne, gemmer dem i en database og returnerer en ny HTML side.

Syntaks for en simpel formular:

```
<form action=tcl-program method=post>
  html-kode
</form>
```

hvor *tcl-program* er en url til det Tcl program som skal udføres og *html-kode* er almindelig HTML kode inklusive de inddatafelter som formularen indeholder.

Der er tre grupper af inddatafelter:

- tekstområder (`<textarea>`)
- indtastningsfelter (`<input>`)
- menuvalg (`<select>`)

## Tekstområder

Brugeren kan indtaste vilkårlige tekster i et tekstområde.

Du kan angive tekstområdets størrelse, dvs. antal tegn per linje og antal linjer.

```
<textarea name=navn rows=r cols=c>
  tekst
</textarea>
```

hvor

- *navn* er navnet på tekstområdet. Navnet anvendes i Tcl-programmet.
- *r* er antallet af rækker i tekstområdet. Hvis brugeren skriver en tekst der er større end tekstområdet, så vil browseren vise en skyder.
- *c* er antallet af tegn per række i tekstområdet. Browseren viser en skyder, hvis brugeren laver linjer på mere end *c* tegn.
- *tekst* er den tekst som vises i tekstområdet. Brugeren kan slette denne tekst.

Eksempel på et tekstområde til kommentarer:

```
<textarea name=comments rows=7 cols=50>
  Skriv dine kommentarer her.
</textarea>
```

## Eksempel på tekstområde (kursiv.html)

```
<html>
<head>
  <title>Kursiv</title>
</head>
<body>
  <h2>Eksempel på et tekstområde.</h2>
  <form action="kursiv.tcl" method=post>
    Den tekst du skriver nedenfor bliver returneret i kursiv.<p>
    Denne tekst bliver sat i kursiv.
  </textarea><p>
  <input type=submit value="Kursiver">
  </form>
</body>
</html>
```

## Eksempel på tekstområde - fortsat (kursiv.tcl)

```
set_form_variables
ns_return 200 text/html "
<html>
<head>
  <title>Kursiver Service</title>
</head>
<body>
  Teksten i kursiv:<p>
  <blockquote>
<pre>
<i>$tekst</i>
</pre>
</blockquote>
</body>
</html>"
```

### Indtastningsfelter

Indtastningsfelter findes i flere varianter angivet ved attributen *type*:

```
<input type=type name=navn value="value">
```

hvor

- *navn* er navnet på inddatafeltet. Navnet benyttes i Tcl programmet.
- *value* er valgfri og angiver den værdi som feltet som udgangspunkt antager.
- *type* angiver hvorledes indtastningsfeltet vises. Du kan vælge mellem følgende typer:

– *text*: En enkelt tekstlinje kan indtastes, f.eks.

```
<input type=text name=by>
```

– *password*: virker som *text*, men indtastningen skjules, f.eks.

```
<input type=password name=kodeord>
```

– *checkbox*: en knap der kan klikkes. Der kan være en eller flere knapper, og en eller flere knapper kan vælges, f.eks.:

```
<input type=checkbox name=cheese value="cheese">
```

```
<input type=checkbox name=olives value="olives">
```

### Indtastningsfelter fortsat

Følgende typer kan også anvendes:

- *radio*: En eller flere knapper vises, men kun en knap kan være valgt på samme tid.

Anvendes ved valg mellem gensidigt udelukkende muligheder.

Attributen *checked* markerer den knap der som udgangspunkt vil være klikket.

```
<input type=radio name=sex value="male">
```

```
<input type=radio name=sex value="women" checked>
```

- *reset*: En knap som nulstiller alle indtastningsfelter.

```
<input type=reset value="Forfra">
```

- *submit*: En knap som anvendes til at sende formulærens data til Tcl programmet angivet i *action*-attributen.

```
<input type=submit value="Send">
```

- *hidden*: Skjulte værdier.

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

### Eksempler på indtastningsfelter (typesetting.html, del 1)

```
<html>
<head>
<title>Typesetting</title>
</head>
<body>
<h2>Eksempel på formattering af en tekstlinje.</h2>
<form action="typesetting.tcl" method=post>
Den tekst du skriver her
<input type=text name=tekst>
bliver formateret således:<p>
```

```
<blockquote>
```

```
  kursiv: <input type=checkbox name=font value="i"><p>
```

```
  fed: <input type=checkbox name=font value="b"><p>
```

```
  understrening: <input type=checkbox name=font value="u"><p>
```

```
</blockquote>
```

```
  Du kan vælge mellem sort og gul baggrund:
```

```
  Grøn <input type=radio name=farve value="Green">
```

```
  Gul: <input type=radio name=farve value="Yellow" checked><p>
```

### Eksempler på indtastningsfelter (typesetting.html, del 2)

Du skal skrive hemmeligt her <input type=password name=kodeord>, for at det virker.<p>

```
<input type=hidden name=skjult_vaerdi
```

```
  value="Denne tekst var skjult i formularen">
```

```
<input type=reset value="Forfra">
```

```
<input type=submit value="Formater tekst">
```

```
</form>
```

```
</body>
```

```
</html>
```

### Eksempler på indtastningsfelter (typesetting.tcl, del 1)

```
set_form_variables
```

```
proc home_page { title body } {
```

```
  return "
```

```
<html>
```

```
<head>
```

```
<title>$title</title>
```

```
</head>
```

```
<body>
```

```
  $body
```

```
</body>
```

```
</html>"
```

```
}
```

### Eksempler på indtastningsfelter (typesetting.tcl, del 2)

```
if { [string compare $kodeord "hemmeligt"] != 0 } {
  ns_return 200 text/html [home_page "Typesetting Service" "kodeordet
  $kodeord er forkert" ]
} else {
  ns_return 200 text/html [home_page "Typesetting Service" "
  <table bgcolor=$farve>
  <tr><td>
  Teksten $tekst ser således ud med formatering <$font>$tekst</$font><p>
  Der var også en skult værdi i formularen: $skjult_vaerdi
  </td></tr>
  </table>
  " ]
}
```

Hvad sker der hvis vi ikke vælger en formatering (kursiv, fed eller understregning)?

Variablen font eksisterer kun, hvis brugeren vælger en formatering.

Hvorledes kan vi checke, at brugeren har indtastet en værdi?

Vi kan checke, at variablen, f.eks. font eksisterer efter kaldet af procedureren set\_form\_variables.

Hvad sker der hvis vi vælger mere end en formatering?

### Procedureren set\_form\_variables (set\_form\_variables.tcl og .html)

Procedureren set\_form\_variables erklærer en variabel svarende til hvert inddata-element i HTML dokumentet.

Den fejler dog, hvis der slet ikke overføres nogen data.

Dette kan undgås ved at give argumentet 0 til procedureren

```
set_form_variables 0
```

### Findes en variabel med info exists

Med kommandoen info og første argument exists kan vi checke at en variabel findes.

F.eks. vil info exists x checke om variablen x findes.

```
% info exists x
0
% set x 42
42
% info exists x
1
```

Husk, at 0 svarer til falsk og 1 til sand.

Syntaks for kommandoen:

```
info exists variabe
```

hvor *variabe* er et variabelnavn

### Eksempel med med info exists (info\_exists.html og info\_exists.tcl)

#### info\_exists.html:

```
<html>
...
<form action="info_exists.tcl" method=post>
  valg 1: <input type=checkbox name=valg value="Du har valgt A"><p>
  valg 2: <input type=checkbox name=valg value="Du har valgt B"><p>
  <input type=submit>
</form>
...
</html>
```

#### info\_exists.tcl:

```
set_form_variables 0

proc home_page { title body } { ... }

if { [info exists valg] } {
  ns_return 200 text/html [home_page "Eksempel med Info Exists" "$valg." ]
} else {
  ns_return 200 text/html [home_page "Eksempel med Info Exists" "Du
  har ikke klikket på en af knapperne." ]
}
```

### Eksempel med med info exists (typesetting2.tcl)

```
set_form_variables 0

proc home_page { title body } { ... }

if { [info exists font] &&
      [info exists tekst] &&
      [info exists farve] &&
      [info exists skjult_vaerdi] &&
      [info exists kodeord] } {
  if { [string compare $kodeord "hemmeligt"] != 0 } {
    ns_return 200 text/html [home_page "Typesetting Service" "kodeordet
    $kodeord er forkert" ]
  } else {
    ns_return 200 text/html [home_page "Typesetting Service" "
    <table bgcolor=$farve>
    <tr><td>
    Teksten $tekst ser således ud med formatering <$font>$tekst</$font><p>
    Der var også en skult værdi i formularen: $skjult_vaerdi
    </td></tr>
    </table>
    " ]
  }
} else {
  ns_return 200 text/html [home_page "Typesetting Service" "Du mangler
  at indtaste en eller flere værdier." ]
}
```

## Menuvalg

Syntaks for en simpel valgliste:

```
<select name=name multiple size=size>
<option selected value=vaegen> tekst
...
<option value=vaegen2>tekstn
</select>
```

hvor

- *name* er navnet på indtastefeltet. Navnet benyttes i Tcl programmet.
- *multiple* er valgfri, og indikerer, at man kan vælge en eller flere elementer fra listen. Hvis *multiple* ikke er angivet kan man kun vælge et element fra listen.
- *size* angiver antal elementer som vises på skærmen.
- *option* tags angiver elementerne i valglisten
- værdien *value*, returneres til serveren, hvis element *i* vælges.
- for element *i* vises teksten *tekst<sub>i</sub>* i valglisten.
- *selected* er valgfri og angiver, at elementet som udgangspunkt er valgt. *selected* kan være angivet for flere elementer.

## Eksempel med Menuvalg (typesetting3.html)

```
<html>
...
<form action="typesetting3.tcl" method=post>
Den tekst du skriver her <input type=text name=tekst>
bliver formateret således:<p>
<blockquote>
<select name=font multiple size=3>
<option selected value="i">kursiv
<option value="b">fed
<option value="u">understregning
</select>
</blockquote>
```

Du kan vælge mellem sort og gul baggrund:

```
<blockquote>
<select name=farve size=1>
<option value="Green">Grøn
<option value="Yellow">Gul
</select>
</blockquote>
<input type=submit value="Formater tekst">
</form>
...
</html>
```

## Eksempel med Menuvalg, fortsat (typesetting3.tcl)

```
set_form_variables 0
proc home_page { title body } { ... }
if { [info exists font] &&
    [info exists tekst] &&
    [info exists farve] } {
    ns_return 200 text/html [home_page "Typesetting Service "
    <table bgcolor=$farve>
<tr><td>
    Teksten $tekst ser således ud med formatering <font>$tekst</font><p>
</td></tr>
</table>
} else {
    ns_return 200 text/html [home_page "Typesetting Service" "Du mangler
at indtaste en eller flere værdier." ]
}
```

Hvad hvis bruger vælger flere font-indstillinger?

Samme problem som i eksemplet typesetting.tcl.

Vi har brug for listen! Vi gennemgår listen i næste uge.

## Web-programmer med tilstand (taealler.tcl)

Skulle formvariable kan bruges til at implementere tilstand i et web-program.

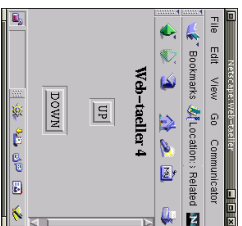
```
proc home_page { title body } { ... }
proc form_up { t } {
    return "<center><h2>Web-taealler $t</h2>
    <form method=post action=taealler.tcl>
    <input type=hidden name=tal value=\"$t\">
    <input type=submit value=UP>
    </form>
    </center>"
}
# saet formvariabel 'tal'
# fejler ikke, hvis tal ikke findes.
set_form_variables 0
if {[info exists tal]} {
    incr tal
} else {
    set tal 0
}
ns_return 200 text/html [home_page "Web-taealler" [form_up $tal]]
```

## Eksempel tæller fortsat (tae1ler2.tcl)

Lad os udvide tæller.tcl med en knap Down som tæller den skulte værdi ned.

Vi har brug for:

- en ekstra knap af type submit
- at de to submit-knapper har forskellige værdier: UP og DOWN
- en ekstra if-kommando som kan finde frem til hvilken af de to knapper der er klikket.



Vi anvender string compare til at finde frem til hvilken knap der er klikket.

```
if {[info exists tall]} {
  if {[info exists submit] && [string compare $submit "UP"] == 0} {
    incr tal
  } else {
    incr tal -1
  }
} else {
  set tal 0
}
```

Udover ovenstående har vi også indsat følgende i form-koden:

```
<input type=submit name=submit value=DOWN>.
```

## Generering af pseudo-tilfældige tal på web-serveren

En maskingenererer pseudo-tilfældige tal/følge synes tilfældig: 91529, 15343, 42, 68836, ...

Men faktisk er hvert tal helt bestemt af det foregående. Det første tal kaldes sekvensens frø (engelsk: seed).

Pseudo-tilfældige tal

- kan bruges til at simulere komplicerede (naturlige) processer,
- eller noget så simpelt som at kaste en terning.

## Kommandoen randomRange

Kommandoen randomRange *N* genererer et ligefordelt tilfældigt tal mellem 0 og *N* - 1.

('Ligefordelt' betyder at alle resultater er lige sandsynlige.)

Eksempler:

- randomRange 100 genererer et tilfældigt tal mellem 0 og 99
- expr [randomRange 6] + 1 genererer et tilfældigt tal mellem 1 og 6, dvs. kaster en terning.
- Aid to Evaluating Your Accomplishments  
([www.photo.net/pilg/careers/Four-random-people.tcl](http://www.photo.net/pilg/careers/Four-random-people.tcl))

## Lad os kaste nogle terninger. (kast\_terning.tcl)

Opgave: lav en procedure kast\_terning, som returnerer et tilfældigt tal mellem 1 og 6.

```
proc kast_terning { } {
  return [ ]
}
```

Opgave: lav et kald til ns\_return, således at du returnerer en side med resultatet af dit terningekast.

```
Du skal anvende proceduren home_page.
ns_return 200 text/html [home_page "
  Resultat af at kaste din terning: [ ] " ]
```

## Regneservice (regneservice.tcl)

Vi har brug for følgende til vores regneservice:

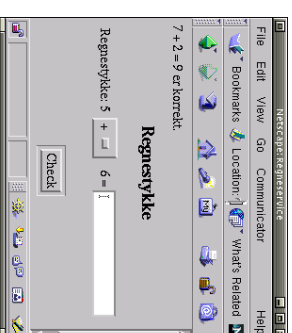
- en menu hvor man kan vælge mellem plus og minus.
- et indtastingsfelt til resultatet.
- skulte felter som husker regnestykke.
- en knap Check, som checker ens indtastning.

Opgave: hvilke indtastefelter har vi brug for?

Opgave: hvilke forvariable har vi brug for?

Kan du udfylde skitsen til formularen med regnestykket:

```
proc form_spg { } {
  set talA [ ]
  set talB [ ]
  return "<form method=post action=regneservice.tcl>
  <input type=hidden name=
  <input type=hidden name=
  Regnestykke: $talA
  <select name=
  <option value=\"minus\">- </select>
  $talB = <input type=text name=
  <input type=submit name=submit value=Check>
  </form>" }
```



## Regneservice - del 2 (regneservice.ctcl)

```
proc home_page { title body } { ... }
# saet formvariabel 'tala', 'talB', 'res', 'regneart' og 'submit'
set_form_variables 0

if {[info exists tala] &&
    [info exists talB] &&
    [info exists res] &&
    [info exists regneart] &&
    [info exists submit]} {
    if {[string compare $regneart "Plus"] == 0} {
        ns_return 200 text/html [home_page
            $res er korrekt.<p>[form_spg]]
        } else {
            ns_return 200 text/html [home_page
                "Regneservice" "$tala + $talB =
                $res er forkert.<p>[form_spg]]"]
        } else {
            if {[expr $tala - $talB] == $res} {
                ns_return 200 text/html [home_page
                    "Regneservice" "$tala - $talB =
                    $res er korrekt.<p>[form_spg]]"]
            } else {
                ns_return 200 text/html [home_page
                    "Regneservice" "$tala - $talB =
                    $res er forkert.<p>[form_spg]]"]
            }
        } else {
            ns_return 200 text/html [home_page "Regneservice" "[form_spg]"]
        }
    }
}
```