

## Forelæsning 8

### Databasetransaktioner via web-forms

- Konstruktion af sites som er databaser
- Datamanipulation via AOLserverns cgi-API
- Eksempel: mailing-list
- ns\_db gethandle
- ns\_db dml
- set\_the\_usual\_form\_variables
- catch
- ns\_ora resultrows
- database-to\_cgi\_string
- ns\_db select
- ns\_db getrow
- ns\_returnredirect

### Eksempel: mailing-list

Vi skal kunne vedligeholde en liste af navne og emails.

Den samme fælles liste vedligeholdes af alle der anvender systemet.

Den eneste information som gemmes er navne og emails.

Man skal kunne oprette, slette og opdatere email-information.

### Step 1: Datamodelen (mailing\_list.sql)

```
create table mailing_list (  
  email      varchar(100) primary key,  
  name      varchar(100) not null  
);
```

Vi anlægger at der ikke er to personer som anvender den samme email, dvs email er primærnøgle.

Vi påtvinger navneoplysninger.

### Konstruktion af sites som er databaser

#### Step 1: Konstruktion af datamodel

- hvilken information skal gemmes og hvordan skal den repræsenteres
- dette er den svære del!

#### Step 2: Udvikling af legale data-transaktioner

- hvordan indsættes data i databasen
- hvordan udtrækkes data fra databasen

#### Step 3: Konstruktion af web-forms og site-map

- brugergrænsefladen er html-kode (forms)

#### Step 4: Konstruktion af database-transaktioner via cgi-scripts

- SQL (Structured Query Language) bruges til de egentlige database-transaktioner

### Eksempel: mailing-list

#### Step 2: Legale transaktioner (mailing\_list.sql)

- Indsættelse af nye emails og navne:

```
insert into mailing_list (email,name) values  
  ('philg@mit.edu', 'Phillip Greenspun');  
  
insert into mailing_list (email,name) values  
  ('bjergoe@it.edu', 'Frank Bjergø');  
  
insert into mailing_list (email,name) values  
  ('nh@it.edu', 'Niels Hallenbergh');
```

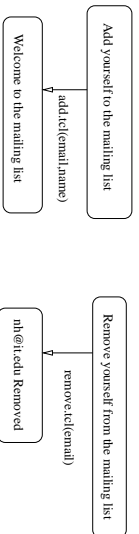
- Sletning af email og tilknyttet navn

```
delete from mailing_list where email = 'philg@mit.edu';
```

- Opdatering af navn for tilknyttet email

```
update mailing_list set name = 'Kurt Nielsen' where email = 'nh@it.edu';
```

### Step 3: Konstruktion af web-forms og site-map



Kasserne i diagrammet repræsenterer tilstande for hvilke HTML-kode vises i en brugers browser.

Unavngivne pile repræsenterer links til en ny tilstand, evt. genereret via etTcl-script.

Navigeringne pile repræsenterer transaktioner for hvilke databasen opdateres gennem etTcl-script.

I parantes angives de formvariable som overføres tilTcl-scriptet, dvs email og name.

Man kan vælge, at skrive Tcl-scriptets navn på pilene.

### Eksempel: mailing-list—filen http://bug.it.edu:8002/lec7/add.html Step 3: Opbygning af web-forms

```
<html>
<head>
<title>Add yourself to the mailing list</title>
</head>
<body bgcolor=#ffffff text=#000000>
<h2>Add yourself to the mailing list</h2>
<form method=post action=add.tcl>
<table>
<tr><td>Name<td><input name=name type=text size=35>
<tr><td>email<td><input name=email type=text size=35>
</table><p>
<input type=submit value="Add Me">
</form>
</body>
</html>
```

### Eksempel: mailing-list—filen add.tcl

#### Step 4: Konstruktion af database-transaktioner via tcl-scripts

```
# set tcl-variables to what the user typed into the form, that is,
# name and email.
set_form_variables 0

if { [info exists email] || [regexp {.+@\.+\.+} $email] } {
# the REGEXP didn't match
ns_return 200 text/html "Your email address doesn't look right to
us. We need your full Internet address ...."
return
}

# if we got here, that means the email address was OK
if { [info exists name] || [string compare $name "" ] == 0 } {
ns_return 200 text/html "You didn't give us your name...."
return
}

# Error checking complete; ready to do real work
ns_return 200 text/html "Here we should (1) insert into database and
(2) return new HTML page."
```

#### Datamanipulation via AOLservers tcl-API

Vi har brug for en forbindelse til databasen.

#### ns\_db gethandle

- returnerer en handle til databasen
- denne handle bruges til at tilgå databasen i efterfølgende kommandoer
- • • • •
- set db [ns\_db gethandle ]
- • • • •
- • • • • \$db • • • • •

Samtidig fortolkning af to tcl-filer får derved forskellige handles til databasen

Handles gives tilbage til webserveren når fortolkning af en tcl-fil afsluttes

#### ns\_db dml

- indsættelse af data - insert

```
• • •
ns_db dml $db "insert into mailing_list
(name, email)
values ($name, $email)"
• • •
```

Samme fremgangsmåde ved opdatering af data (update) og sletning af data (delete).

#### Eksempel: mailing-list—filen addv2.tcl

```
# set tcl-variables to what the user typed into the form, that is,
# name and email.
set_form_variables 0

set db [ns_db gethandle]

if {![info exists email] || ![regexp {.+@.+} $email]} { ... }
if {![info exists name] || [string compare $name ""] == 0} { ... }

set insert_sql "insert into mailing_list (email, name)
                values ('$email', '$name')"
ns_db dml $db $insert_sql

ns_return 200 text/html "Welcome to the mailing-list."
```

Handlen til databasen gemmes i variabelen db.

SQL-kommandoen gemmes i variabelen insert\_sql.

Vi udfører SQL-kommandoen med ns\_db dml.

Læg mærke til, at vi anvender ' omkring strenge i SQL.

Læg mærke til, at vi anvender \$email og \$name når vi laver SQL-kommandoen.

**Problem:** Prøv at indsætte navnet Ken Mc'Donald

#### Eksempel: mailing-list—filen addv3.tcl

```
# set tcl-variables to what the user typed into the form, that is,
# name, QQname and email, QQemail.
set_the_usual_form_variables 0

set db [ns_db gethandle]

if {![info exists email] || ![regexp {.+@.+} $email]} { ... }
if {![info exists name] || [string compare $name ""] == 0} { ... }

set insert_sql "insert into mailing_list (email, name)
                values ('$QQemail', '$QQname')"
ns_db dml $db $insert_sql

ns_return 200 text/html "Welcome to the mailing-list."
```

Vi anvender QQemail og QQname når vi bygger SQL-kommandoen.

**Problem:** Prøv at indsætte den samme post to gange.

Kan vi indsætte en post uden email?

Kan vi indsætte en post uden navn?

#### Double-Quotes

For at indsætte strengen Ken Mc'Donald skal man "double-quote", dvs ' skrives ' ':

```
SQL> insert into mailing_list (email, name) values
      ('Ken@mac.org', 'Ken Mc'Donald');
```

ERROR:

ORA-01756: quoted string not properly terminated

```
SQL> insert into mailing_list (email, name) values
      ('Ken@mac.org', 'Ken Mc''Donald');
1 row created.
```

```
SQL> select name from mailing_list where email='Ken@mac.org';
```

NAME

-----

Ken Mc'Donald

I stedet for set\_form\_variables kan vi anvende set\_the\_usual\_form\_variables.

I addv2.tcl får vi da fire variable: email, name, QQemail og QQname, hvor QQemail og QQname har alle ' erstatet af ' ':

His name = Ken Mc'Donald, da vil QQname = Ken Mc' 'Donald.

His name = Ken Hansen, da vil QQname = Ken Hansen.

I vores SQL-kommando anvender vi QQemail og QQname.

#### Fejlhåndtering med catch

```
catch script errmsg
```

Kommandoen catch udfører kommandoerne i script, og to ting kan ske:

1. *script* fejler ikke, i hvilket tilfælde catch returnerer 0
2. *script* fejler, i hvilket tilfælde catch returnerer et tal forskelligt fra 0 og variabelen *errmsg* sættes lig en beskrivelse af fejlen.

Man anvender ofte catch sammen med en if-kommando, således at man kan reagere på fejl:

```
if { [catch script errmsg] }
```

```
    fejl_kode
```

```
 }
```

Koden fejl\_kode udføres kun, hvis der opstår en fejl når script udføres.

I koden fejl\_kode kan vi henvise til variabelen errmsg.

#### Eksempel:

```
% if { [catch {expr 4 / 0} errmsg] } { puts "Fejl: $errmsg" }
Fejl: divide by zero
```

#### Håndtering af databasefejl—addv4.tcl (http://hug.it.edu:8002/lect7/addv4.html)

Med catch kan vi håndtere fejl fra databasen:

```
if {[catch { ns_db dml $db $insert_sql } errmsg] } {
    fejl_kode
}
```

Vi kan f.eks. returnere en side til brugeren, som forklarer problemet.

```
set_the_usual_form_variables 0
set_db [ns_db gethandle]
if {[info exists email] || [regexp {.+@.+\.+} $email] } { ... }
if {[!info exists name] || [string compare $name ""] == 0 ] { ... }

set_insert_sql "insert into mailing_list (email, name)
                values ('$QOemail','$QOname');"
if {[catch { ns_db dml $db $insert_sql } errmsg] } {
    # the insert went wrong; the error description
    # will be in the Tcl variable ERRMSG
    ns_return 200 text/html "The database didn't accept your
                            insert, most likely because your
                            email address is already on
                            the mailing list...<p>
                                The error message: $errmsg"
    } else {
        ns_return 200 text/html "Welcome to the mailing-list."
    }
```

#### Stiering af data med ns\_db dml

- stiering af data - delete

```
ns_db dml $db "delete from mailing_list
                where email = 'nh@it.edu'"
```

Dette vil normalt ikke fejle medmindre vores SQL-kommando er forkert.

Der er mulighed for at 0, 1 eller flere rækker slettes.

I forbindelse med vores mailingliste

- forventer vi at slette 1 række.

- hvis vi sletter 0 rækker skyldes det at emailen ikke findes i mailinglisten.

- vi kan ikke komme til at slette mere end 1 række, hvorfor?

#### Antal rækker der slettes – ns\_ora resultrows

Med kommandoen ns\_ora resultrows og vores handle db kan vi spørge om antallet af rækker der blev

slettet:

```
if {[ ns_ora resultrows $db] == 0 } {
    # 0 rows were affected
    } else {
        # the delete affected at least one row
    }
```

#### Eksempel: mailing-list—filen http://hug.it.edu:8002/lect7/remove.html

```
<html>
<head>
<title>Remove yourself from the mailing list</title>
</head>
<body bgcolor=#ffffff text=#000000>
<h2>Remove yourself from the mailing list</h2>
<form method=post action=remove.tcl>
<table>
<tr><td>email<td><input name=email type=text size=35></tr>
</table><p>
<input type=submit value="Remove Me">
</form>
</body>
</html>
```

Bruger indlaster email som entydigt identificerer en person.

#### Eksempel: mailing-list—filen remove.tcl

```
set_the_usual_form_variables
set_db [ns_db gethandle]

# note that the dual calls to the SQL UPPER function
# ensure that the removal will be case insensitive
set_delete_sql "delete from mailing_list
                where upper(email) = upper('$QOemail');"

ns_db dml $db $delete_sql
if {[ ns_ora resultrows $db] == 0 } {
    ns_return 200 text/html "...Error..."
    } else {
        ns_return 200 text/html "
        <html><head><title>$email Removed</title>
        </head><body bgcolor=#ffffff text=#000000>
        <h2>$email Removed</h2>
        <hr>
        You have been removed from the <a href=/index.html>www.greedy.com</a>
        mailing list.
        <hr>
        <address>webmaster@greedy.com</address>
        </body></html>"
    }
```

```

Eksempel: mailing-list—filen http://hug.it.edu:8002/lec7/update.html
<html>
<head>
<title>update yourself on the mailing list</title>
</head>
<body bgcolor=#ffffff text=#000000>
<h2>update name</h2>
<form method=post action=update.tcl>
<table>
<tr><td>Name<td><input name=name type=text size=35>
<tr><td>email<td><input name=email type=text size=35>
</table><p>
<input type=submit value="Update Name">
</form>
</body>
</html>

```

### Opdatering af data med ns\_db dml

Helt identisk med sletning af data med ns\_db dml.

- opdatering af data - update  

```

ns_db dml $db "update mailing_list set name = 'Kurt Nielsen'
where email = 'nh@it.edu'"

```

Dette vil normalt ikke fejle medmindre vores SQL-kommando er forkert.

Der er mulighed for at 0, 1 eller flere rækker opdateres.

I forbindelse med vores mailingliste

- forventer vi at opdatere 1 række.
- hvis vi opdaterer 0 rækker skyldes det at emajlen ikke findes i mailinglisten.
- vi kan ikke komme til at opdatere mere end 1 række, hvorfor?

### Antal rækker der opdateres – ns\_ora resultrows

Med kommandoen ns\_ora resultrows og vores handle db kan vi spørge om antallet af rækker der blev opdateret:

```

if { [ns_ora resultrows $db] == 0 } {
# 0 rows were affected
} else {
# the delete affected at least one row
}

```

```

Eksempel: mailing-list—filen update.tcl
set_the_usual_form_variables 0
set db [ns_db gethandle]
if { [llinfo exists email] || [lregexp {.+@.\.+} $email] } { ... }
if { [llinfo exists name] || [string compare $name "" ] == 0 } { ... }

set update_sql "update mailing_list set name = '$Qname'
where upper(email) = upper('$Qemail')"
ns_db dml $db $update_sql

if { [ns_ora resultrows $db] == 0 } {
ns_return 200 text/html "... Error ..."
} else {
# the update affected at least one row so update must
# have been successful
ns_return 200 text/html "
<html><head><title>Name for $email Updated</title>
</head><body bgcolor=#ffffff text=#000000>
<h2>Name for $email Updated</h2>
<hr>
You have been updated on the <a href=/index.html>www.greedy.com</a>
mailing list.
<hr>
<address>webmaster@greedy.com</address>
</body></html>"
}

```

```

Eksempel: mailing-list—filen http://hug.it.edu:8002/lec7/search.html
<html>
<head>
<title>Search the mailing list</title>
</head>
<body bgcolor=#ffffff text=#000000>
<h2>Search the mailing list</h2>
<form method=post action=search.tcl>
<table>
<tr><td>email<td><input name=email type=text size=35>
</table><p>
<input type=submit value="Search">
</form>
</body>
</html>

```

### Dataudrækning med `database_to_tcl_string db sql` – en celle i en række

- udrækning af en celle i en tabel
- fejler hvis der returneres andet end en celle

Eksempel:

```
...
set query "select name
          from mailing_list
          where email = 'nh@it.edu'"
set name [database_to_tcl_string $db $query]
...
```

Kommandoen `database_to_tcl_string` følger, hvis der ikke findes præcis en række med en celle.

Derfor anvender vi `catch`:

```
if { catch {set name [database_to_tcl_string $db $query]} errmsg } {
  fejl_kode
} else {
  variabel name indeholder cellen fra databasen
}
```

### Eksempel: `mailing-list`—filen `search.tcl`

```
set_the_usual_form_variables
set db [ns_db gethandle]

set query "select name
          from mailing_list
          where upper(email) = upper('$Qqemail')"

if { catch {set name [database_to_tcl_string $db $query]} errmsg } {
  ns_return 200 text/html "... Error ..."
} else {
  ns_return 200 text/html "
<html><head><title>$email Found</title>
</head><body bgcolor=#ffffff text=#000000>
<h2>$email Found</h2>
<hr>
We found the name $name for email $email on the <a
href=/index.html>www.greedy.com</a>
mailing list.
<hr>
<address>webmaster@greedy.com</address>
</body></html>"
}
```

### Dataudrækning – 0, 1 eller flere rækker med `ns_db select`

```
ns_db select db sql
```

hvor

`db` er vores database handle og

`sql` er den forespørgsel som resulterer i 0, 1 eller flere rækker.

`ns_db select` returnerer en *selection*:

```
set query "select name, email from mailing_list"
set selection [ns_db select $db $query]
```

### `ns_db getrow` for at hente de enkelte rækker

```
ns_db getrow db selection
```

hvor *selection* indeholder resultatet af et kald til `ns_db select`

```
while { [ns_db getrow $db $selection] } {
  # set tcl-variables corresponding to the selected columns
  set_variables_after_query
}
```

`ns_db getrow` returnerer 0, når der ikke er flere rækker.

`set_variables_after_query` skaber variable svarende til feltene i forespørgslen, f.eks. `email` og `name`.

### Oversigt over `mailing-list`en (`http://hug.it.edu:8002/lec7/list.tcl`)

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

set db [ns_db gethandle]

set query "select name, email from mailing_list"
set selection [ns_db select $db $query]

set tab "<table>\n<tr><th>Name</th><th>Email</th></tr>\n"
while { [ns_db getrow $db $selection] } {
  # set tcl-variables corresponding to the selected columns,
  # i.e., name and email
  set_variables_after_query
  append tab "<tr><td>$name</td><td>$email</td></tr>\n"
}
append tab "</table>\n"

ns_return 200 text/html [home_page "Mailing list" "This is the
mailing list: <br>$tab"]
```

### Vi kan kode form-variable direkte i en url

Tcl-scriptet `remove.tcl` fjerner posten med den email som er angivet i form-variablen `email`. Vi kan derfor skrive, f.eks.

```
remove.tcl?email=nh@it.edu
```

Vi adskiller form-variable med `&`, f.eks.: `remove.tcl?email=nh@it.edu&username=kurt`

### Oversigt over mailinglist—med slet option (`http://hug.it.edu:8002/lec7/list.tcl`)

```
proc home_page { title body } { ... }
set db [ns_db gethandle]
set query "select name, email from mailing_list"

set selection [ns_db select $db $query]
set tab "<table>\n<tr><th>Name</th><th>Email</th></tr>\n"
while { [ns_db getrow $db $selection] } {
    # set tcl-variables corresponding to the selected columns,
    # i.e., name and email
    set_variables_after_query
    append tab "<tr><td>$name</td><td>$email</td><td>
    <a href=\"remove.tcl?email=$email\">remove</a></td></tr>\n"
}
append tab "</table>\n"

ns_return 200 text/html [home_page "Mailing list" "This is
the mailing list: <br>$tab"]
```

Tilhojskolen

Databasestøttet Webpublicering, forår 2002

Side 8-25

### ns\_returnredirect (removev2.tcl)

Fra et Tcl-script kan vi returnere resultatet af at kalde et andet Tcl-script.

Fra `listv2.tcl` kalder vi `removev2.tcl`.

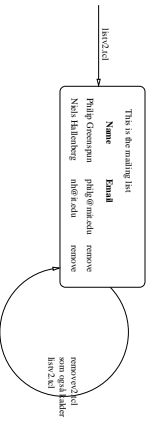
```
removev2.tcl sletter en email fra databasen.
```

Derefter ønsker vi at se mailinglisten igen, dvs. resultatet af at køre `listv2.tcl` igen.

### removev2.tcl:

```
set_the_usual_form_variables

set db [ns_db gethandle]
set delete_sql "delete from mailing_list
where upper(email) = upper('$QOemail')"
ns_db dml $db $delete_sql
ns_returnredirect listv2.tcl
```



Tilhojskolen

Databasestøttet Webpublicering, forår 2002

Side 8-26