Lisp 6: Postmodern

Eirik Alderslyst Nygaard Øystein Ingmar Skartsæterhagen

Programvareverkstedet

29. april 2010

(Postmodern)

(Et postgresql klient-bibliotek)

(Vil vise eksempler på hvordan makroer blir brukt for å forenkle kode)

(Viser bruk av DSL)

(Finnes på http://marijn.haverbeke.nl/postmodern/ eller last ved ved hjelp av clbuild)

(Postmodern: Koble til)

(Lag en permanent oppkobling:)

```
(connect-toplevel
  "database" "username" "password" "hostname")
```

(Lag en oppkobling som er gyldig så lenge man er innenfor et skop:)

```
(with-connection '("database" "username"
                   "password" "hostname")
  ...)
```

(Postmodern: Spørrespråk)

(Postmodern sitt spørrespråk heter S-SQL)

(Gir deg muligheten til å skrive SQL ved hjelp av lister)

(`sql` gir deg strengen som vil bli sendt til database-serveren)

```
(sql (:select '* :from 'table))
```

```
(SELECT * FROM "table")
```

(`query` brukes for å eksekvere en spørring mot databasen)

```
(query (:insert-into 'table :set 'id 1 'key value))
```

(Postmodern: Spørrespråk)

```
(sql (:update 'table :set 'name "Jenny"
             :where (:or (:= 'city "Liverpool")
                         (:= 'city "Manchester"))))
```

```
UPDATE "table" SET name = E'Jenny'
               WHERE ((city = E'Liverpool') or
                      (city = E'Manchester'))
```

(Postmodern: Opprette tabeller)

(Kan velge mellom DAO (ORM) eller en selvskrevet spørring)

```
(:create-table
  people
  ((id :type serial :primary-key t)
   (name :type (varchar 128))
   (hat :type (varchar 128))))
```

```
(:create-table
  friends
  ((personid :type integer :references (people))
   (friendid :type integer :references (people))))
```

(Postmodern: Kompilerte spørringer)

(`defprepared` kompilerer spørringer for deg og lar deg kalle de som funksjoner)

```
(defprepared function-name
    query)
```

(Postmodern: Kompilerte spørringer)

```
(postmodern:defprepared add-friend
    (:insert-into 'friends :set 'personid '$1
                               'friendid '$2))
```

```
(postmodern:defprepared create-person
    (:insert-into 'people :set 'name '$1 'hat '$2
                  :returning 'id)
  :single)
```

```
CL-USER> (add-friend "Bob" "Fez")
1
1
```

(Postmodern: Fyll inn databasen)

(La oss lage noen personer og sette opp noen vennskapsforhold)

```
(let ((bobid (create-person "Bob" "Fez"))
      (eveid (create-person "Eve" "Bowler"))
      (janeid (create-person "Jane "Fedora")))
  (add-friend bobid eveid)
  (add-friend eveid bobid)
  (add-friend janeid eveid)
  (add-friend janeid bobid)
```

(Stakkars Jane)

(Postmodern: Transaksjoner)

(En transaksjon lages ved hjelp av `with-transaction`)

(Når en transaksjonsblokk er ferdig vil den bli sjekket inn)

```
(with-transaction ()
  ...)
```

(Innsjekking og avbryting av en transaksjon kan og gjøres manuelt med to funksjoner:)

```
(with-transaction (name)
  ...
  (if foo
      (commit-transaction name)
      (abort-transaction name))))
```

(Postmodern: Transaksjoner, hattebytte)

(La oss gi to personer mulighet til å bytte hatt med hverandre)

(Først trenger vi to hjelpefunksjoner, en for å hente ut hatten til en person, og en for å sette hatten til en person)

```
(defprepared get-hat
    (:select 'hat :from 'people :where (:= 'id '$1))
  :single)
(defprepared update-hat
    (:update 'people :set 'hat '$2
             :where (:= 'id '$1)))
```

(Postmodern: Transaksjoner, hattebytte)

(`swap-hats` tar seg av selve byttet)

(Den henter først ut hattene til begge personene, og oppdaterer så hver av personenen med den andre sin hatt)

```
(defun swap-hats (person1 person2)
  (with-transaction ()
    (let ((hat1 (get-hat person1))
          (hat2 (get-hat person2)))
      (update-hat person1 hat2)
      (update-hat person2 hat1))))
```