

### Programmering af Web Applikationer med SMLserver

- ▶ Hvad er SMLserver
  - ▶ Motivation
  - ▶ Et SMLserver script eksempel
  - ▶ SMLserver køretidsmodel
- ▶ Standard ML (SML) for PHP programmører på 20 minutter
- ▶ Eksempel: **Dynamisk opskrift**
- ▶ RDBMS Integration
- ▶ Eksempel: **Gæstebog**
- ▶ Konsekvenser af køretidsmodellen
- ▶ Effektivitet og skalérbarhed



# Hvad er SMLserver?

SMLserver er en webserver platform for scripts skrevet i programmeringssproget **Standard ML**.

## **SMLserver har et rigt web-API indeholdende:**

- ▶ Nem **adgang til form data** og request-info, som ex. cookies.
- ▶ Adgang til **RDBMS'er** gennem et **generisk API** som tillader **genbrug af databaseforbindelser**.
- ▶ Support for “data-caching” og indhentning af fremmede websider.

## **Andre Features**

- ▶ Programmer oversættes til **bytecode**, som kun loades en gang, men kan køres mange gange.
- ▶ En multi-trådet køretidsmodel tillader at mange forespørgsler kan behandles samtidigt.

## Statisk typning — Statisk typning — Statisk typning

- ▶ Web-applikationer eksponeres til brugere tidligt
- ▶ Web-applications opdateres hyppigt
- ▶ **Statisk typning** kan lette vedligeholdelsen og øge robustheden

## Højere-ordens funktioner

- ▶ Høj grad af kodegenbrug og “smalle” API’er.
- ▶ Gode til at separere udviklings-tasks (design vs. applikationslogik)

## Et rigt modulsystem

- ▶ Name-space håndtering
- ▶ Høj grad af kodegenbrug (abstraktion)

# Hvordan ser SMLserver scripts ud?

## Scripts

```
val time_of_day =  
  Date.toString (Date.fromTimeLocal(Time.now()))  
val _ = Ns.Conn.return  
  '<HTML>  
    <BODY><H1>Hello world!</H1>  
    The current date and time is ^time_of_day  
  </BODY>  
</HTML>'
```

## ML Server Pages

```
<HTML>  
  <BODY><H1>Hello world!</H1>  
  The current date and time is  
  <?MSP= Date.toString (Date.fromTimeLocal(Time.now())) ?>  
  </BODY>  
</HTML>
```

## Eksempel på projekt-fil tod.mlb:

```
local $SML_LIB/basis/basis.mlb
    ../lib/lib.mlb
in
    scripts
        demo/time_of_day.sml
        demo/time_of_day.msp
    end
end
```

## Kørsel af SMLserver Scripts

- ▶ Projekter oversættes til bytecode fragmenter med kommandoen `smlserverc tod.mlb`
- ▶ Et script loades ind i webserveren når scriptet forespørges første gang (scriptet caches i webserveren for fremtidige forespørgsler)

- ▶ Aritmetik, variabler og strenge (*som i PHP*)
- ▶ **Typer** og adskildelsen mellem oversættelsestid og køretid
- ▶ Erklæring af funktioner (*som i PHP*)
- ▶ Funktioner der tager **funktioner som argumenter**
- ▶ Strukturer og signaturer til håndtering af navnerum (name space management) og **modularitet**

- ▶ Standard ML supporterer de sædvanlige aritmetiske operationer og variabler kan benyttes til at holde på temporære værdier, som i PHP.
- ▶ Strengkonstanter i Standard ML skrives i double-quotes og strengte sammensættes med funktionen `^`.
- ▶ Betingede udtryk ligner betingede sætninger i PHP.

```
val weight = 90.0
val height = 190.0
val bmi = weight / ( (height/100.0) * (height/100.0) )
val msg = if ( bmi > 30.0 ) then "too high!"
           else if ( bmi > 20.0 ) then "normal"
           else "too low"

val _ = print( "Your BMI is " ^ (Real.toString bmi)
              ^ ", which is " ^ msg )
```

## Oversættelse med MLKit Standard ML oversætteren:

```
doobie $ mlkit bmi.sml
[wrote executable file: run]
doobie $ ./run
Your BMI is 24.9307479224, which is normal
```

## Typer: Små programændringer kan medføre oversætterfejl

Ved at ændre den første erklæring til “val weight = 90”, vil oversætteren give følgende besked:

```
doobie $ mlkit bmi2.sml
bmi2.sml, line 3, column 10:
  val bmi = weight / ( (height / 100.0) * (height / 100.0) )
  ~~~~~
```

Type clash,

```
  operand suggests operator type: int * real->real
  but I found operator type:      real * real->real
```

- ▶ En **explicit typed** version af BMI-programmet:

```
val weight:real = 90.0
val height:real = 190.0
val bmi:real = weight / ( (height / 100.0) * (height / 100.0) )
val msg:string = if ( bmi > 30.0 ) then "too high!"
                  else if ( bmi > 20.0 ) then "normal"
                  else "too low"
val _ = print ( "Your BMI is " ^ (Real.toString bmi)
               ^ ", which is " ^ msg )
```

- ▶ Operatoren ^ har type **string \* string → string**.
- ▶ Operatoren > har type **real \* real → bool**.
- ▶ Funktionen Real.toString har type **real → string**.
- ▶ Typeinferens tillader undladelse af eksplicite typeannoteringer.
- ▶ **Hvilke typer har funktionerne / og \*?**

- ▶ Funktioner kan bruges til at give navn til en blok af kode.

- ▶ Funktioner i SML ser ud som i PHP:

```
fun italic ( s ) = "<i>" ^ s ^ "</i>"
```

- ▶ Her er den samme funktion i PHP:

```
function italic ( $s ) { return "<i>" . $s . "</i>"; }
```

- ▶ Funktionen `italic` har type **string** → **string**.

- ▶ Her er en eksplicit typet version af funktionen:

```
fun italic ( s : string ) : string =  
  "<i>" ^ s ^ "</i>"
```

- ▶ Brugerdefinerede funktioner kan bruges præcis som indbyggede funktioner:

```
val text = "I " ^ italic("really") ^ " like coffee!"
```

- ▶ I Standard ML kan funktioner tage funktioner som argument:

```
fun bold ( s : string ) : string = "<b>" ^ s ^ "</b>"  
fun format ( emph : string->string ) : string =  
  "I " ^ emph("really") ^ " like coffee!"
```

```
val _ = print ( "Text A: " ^ format(italic) ^ "\n" ^  
               "Text B: " ^ format(bold) ^ "\n" )
```

- ▶ Typen af funktionen format er (**string** → **string**) → **string**.
- ▶ Her er resultatet af at oversætte og køre programmet:

```
doobie$ mlkit format.sml  
[wrote executable file: run]  
doobie$ ./run  
Text A: I <i>really</i> like coffee!  
Text B: I <b>really</b> like coffee!
```

- ▶ Strukturer muliggør **gruppering** af relaterede funktioner.
- ▶ Strukturer bruges til **“programming-in-the-large”**.
- ▶ Vi har allerede gjort brug af basis-strukturer: **Real**, **Date** og **Time**.
- ▶ Det er ligetil at erklære en struktur:

```
structure MyFormatLib =  
  struct  
    fun italic (s) = ...  
    fun bold (s) = ...  
  end
```

- ▶ For at referere til en komponent i en struktur benyttes dot-notationen; alternativt kan strukturen “åbnes”:

```
val _ = print (MyFormatLib.italic ("Hello World"))  
open MyFormatLib  
val _ = print (bold ("Hello"))
```

- ▶ En signatur *beskriver* en struktur:

```
signature MY_FORMAT_LIB =  
  sig  
    val italic : string -> string  
    val bold   : string -> string  
  end
```

- ▶ Strukturen MyFormatLib *matcher* signaturen MY\_FORMAT\_LIB.
- ▶ Følgende erklæring garanterer at strukturen implementerer de funktioner der er specificeret i signaturen MY\_FORMAT\_LIB:

```
structure MyFormatLib : MY_FORMAT_LIB =  
  struct  
    fun italic (s) = ...  
    fun bold (s) = ...  
  end
```

# “Quotations” til HTML/SQL indlejring

- ▶ En konstant-quotation skrives med backquotes: ‘Hello World’.
- ▶ Quotations sammensættes med `^^` : **quot \* quot** → **quot**.
- ▶ Quotations kan strække sig over flere linier (som strenge i PHP)
- ▶ **Ex:** struktur der implementerer funktionen `Page.return`, som tager to argumenter, en streng `head` og en quotation `body`:

```
structure Page = struct
  fun return head body = Ns.return
    (‘<html><head><title>^head</title></head>
     <body bgcolor=white>
       <h2>^head</h2>‘ ^^ body ^^ ‘<hr><i>Served by
         <a href=http://www.smlserver.org>SMLserver</a></i>,
         <i><a href="/demo/index.sml">Back to index page</a>.
       </i></body></html>‘)
end
```

- ▶ Funktionen `Web.return` sender strengen (repræsenteret som en quotation) til browseren.
- ▶ Notationen `^head` bruges til at indlejre indholdet af `head`.

# Eksempel: Æblekageopskrift (Næsten som i PHP)

- ▶ Eksemplet består af scriptet `recipe.sml` og `recipe.html`.

- ▶ Filen `recipe.html`:

```
<html>
  <body bgcolor=white><h2>Dynamic Recipe: Apple Pie</h2>
  Enter the number of people you're inviting for apple pie:
  <form method=post action=recipe.sml>
  <input type=text name=persons>
  <input type=submit value="Compute Recipe">
  </form><hr><i>Served by
    <a href=http://www.smlserver.org>SMLserver</a></i>
  </body>
</html>
```

- ▶ Se <http://www.smlserver.org/demo/recipe.html>

## Første del af `recipe.sml`:

```
fun pr_num s r =  
  if r == 1.0 then "one " ^ s  
  else if real(round r) == r then  
    Int.toString (round r) ^ " " ^ s ^ "s"  
  else Real.toString r ^ " " ^ s ^ "s"
```

## Eksempler på brug af funktionen `pr_num`

`pr_num "chair" 1.0`  $\implies$  "one chair"

`pr_num "chair" 2.0`  $\implies$  "2 chairs"

`pr_num "cup" 2.2`  $\implies$  "2.2 cups"

► Anden del af `recipe.sml`:

```
val persons = FormVar.wrapFail FormVar.getNatErr ("persons", "Persons")
```

```
val _ = Page.return "Apple Pie Recipe"
```

```
'To make an Apple pie for ^{(pr_num "person" persons)}, you  
need the following ingredients:
```

```
<ul> <img align=right src=applepie.jpg>  
<li> ^{(pr_num "cup" (persons / 16.0))} butter  
<li> ^{(pr_num "cup" (persons / 4.0))} sugar  
<li> ^{(pr_num "egg" (persons / 4.0))}  
<li> ^{(pr_num "teaspoon" (persons / 16.0))} salt  
<li> ^{(pr_num "teaspoon" (persons / 4.0))} cinnamon  
<li> ^{(pr_num "teaspoon" (persons / 4.0))} baking soda  
<li> ^{(pr_num "cup" (persons / 4.0))} flour  
<li> ^{(pr_num "cup" (2.5 * persons / 4.0))} diced apples  
<li> ^{(pr_num "teaspoon" (persons / 4.0))} vanilla  
<li> ^{(pr_num "tablespoon" (persons / 2.0))} hot water  
</ul>
```

```
Combine ingredients in order given. Bake in greased 9-inch pie pans for  
45 minutes at 350F. Serve warm with whipped cream or ice cream. <p>  
Make <a href=recipe.html>another recipe</a>.'
```

► Formvariablen "persons" tjekkes med brug af biblioteksfunktionen `FormVar.getNatErr` (and friends).

## Et Generisk Database API

- ▶ Support for Oracle, Postgresql og MySQL

## Database pooling

- ▶ Et “handle” identificerer en forbindelse til en RDBMS
- ▶ SMLserver vedligeholder at antal pools (hver med et antal handles)
- ▶ Et database handle ejes af højst et script ad gangen.
- ▶ Handles “tages” og “frigives” på en måde så **deadlocks undgås**.
- ▶ Programmøren behøver ikke kende til handles, jo mindre transaktioner bruges.

## Signaturen

```
signature NS_DB =  
  sig  
    val dml : quot -> unit  
    val fold : ((string->string)*'a->'a)->'a->quot->'a  
    ...  
  end
```

## Bemærk

- ▶ Quotations bruges til at indlejre SQL.
- ▶ Funktionen **dml** tillader eksekvering af insert og update statements
- ▶ Funktionen **fold** folder over rækkerne i en SQL forespørgsel.

## Datamodel

```
create table guest (  
  email    varchar(100),  
  name     varchar(100),  
  comment  varchar(2000)  
);
```

## Tilføjelse af ny gæst

```
val _ = Db.dml  
  'insert into guest (name,email,comment)  
    values (^(Db.qqq n),^(Db.qqq e),^(Db.qqq c))'
```

## Bemærk

- ▶ Funktionen `Db.qqq` escaper quotes (') for SQL-indlejring.

## Vis indhold af gæstebog

```
fun layoutRow (g: string->string, acc:quot) =  
  '<li> <i>^(g "comment")</i>  
  -- <a href="mailto:^(g "email")">^(g "name")</a>  
  <p>' ^^ acc  
  
val rows = Db.fold layoutRow ''  
  'select email,name,comment from guest'  
  
val _ = Page.return "Guest Book"  
  ('<ul>' ^^ rows ^^ '</ul>' ^^  
  '<form action=guest_add.sml>...</form>')
```

## Bemærk

- ▶ Funktionen `Page.return` sender en side til browseren.
- ▶ Det første argument til `fold` er en funktion som formatterer en række.

## Skalérbar Køretidsmodel

- ▶ Scripts kører i et “tom lager” når forespørgsler behandles.
- ▶ Dvs: Tilstand gemmes ikke på serveren mellem forespørgsler.
- ▶ Istedet må tilstand vedligeholdes eksplicit ved brug af en RDBMS, evt. kombineret med brug af cache-primitiver.
- ▶ Alternativ: emulér tilstand ved brug af formvariable eller cookies.

## Den Regionsbaserede Lagermodel

- ▶ SMLserver benytter sig ikke af traditionel spildopsamling (garbage collection).
- ▶ Istedet indsætter oversætterens lager-allokerings og lager-deallokerings direktiver i koden under oversættelsen.

# Effektivitetsmålinger med ApacheBench

Program	Forespørgsler / sekund			
	MosML MSP	AOLserver TCL	Apache PHP	SMLserver MSP
hello	55	724	489	<b>1326</b>
date	54	855	495	<b>1113</b>
db	27	558	331	<b>689</b>
guest	25	382	274	<b>543</b>
calendar	36	27	37	<b>101</b>
mul	50	185	214	<b>455</b>
table	21	59	0.7	<b>93</b>
log	8	12	0.4	<b>31</b>

- ▶ ApacheBench (v. 1.3d) bruger 8 tråde (i 60 sekunder hver)
- ▶ 850Mhz Pentium 3 Linux box (384Mb RAM)

# Effektivitetsmålinger med ApacheBench

Program	Forespørgsler / sekund		
	SMLserver MSP	Ingen script caching	Med biblioteks- kørsel
hello	<b>1326</b>	916	349
date	<b>1113</b>	744	337
db	<b>689</b>	516	275
guest	<b>543</b>	356	249
calendar	<b>101</b>	69	80
mul	<b>455</b>	300	241
table	<b>93</b>	84	75
log	<b>31</b>	30	28

- ▶ Caching af loaded script bytecode forøger performance 3–53 procent.
- ▶ Eksekvering af bibliotekskode degraderer performance 10–74 procent.

På ITU bruges SMLserver til at køre **mit.itu**, som bl.a. indeholder:

- ▶ Et kursusevalueringssystem
- ▶ Et alumni-system
- ▶ Et system til håndtering af online-ansøgninger
- ▶ Et HR system, inkl. **Find Person**
- ▶ Et forespørgselssystem (RSVP)

Kodebasen er på mere end 250.000 linier Standard ML.