# ASP noter til eksamen

(Af Morten Rasmussen, januar 2003)

# 1 Active Server Pages

- ASP er en platform hvorpå det er muligt at lave interaktive (/dynamiske) hjemmesider (radikal forskel fra HTML) - Microsoft
- Kan også betegnes som et script miljø, **ikke et sprog eller program men en teknologi**
- Mulige script til rådighed: JavaScript og VBScript (mest anvendte)
- Adgang til databaser
- Først generere output (HTML) når burger vil se det – altså efter en forespørgsel
- Hvilken browser er underordnet (!?)
- Siderne kan være afhængig af tid, sted og brugervalg
- ASP blev annonceret i 1996
- ASP muliggøre individualiserede sider
- ASP er ikke et sprog
- Mere arbejde for server, men fordele overskyder bagdele
- ASP kode kan kun vises på serveren – brugbart at det ikke kan ses af browser, følsomme informationer kan beskyttes ex. med brugerid og password. Man kan også skjule beregninger.
- Meget sikrere at skrive sin kode i ASP
- ASP sider er en blanding af HTML og ASP script

# 2 Sprog

- Et script sprog er et programmeringssprog i gængs forstand (med sproglige konstruktioner)
- To ting adskiller ASP-programmeringens scriptsprog fra traditionelle sprog
  - Scriptsprog er fortolkede (skal ikke kompileres, langsommere m. netkomm. er flaskehals)
  - Scriptsprog er indlejrede i HTML (ASP koden erstattes under fortolkning til HTML)

- Http-serveren fortolker den del af scriptkoden som er serverscript, resten (klientscript) fortolkes i browseren
- Java-Script er den mest udbredt clientside scriptsprog (en slags standard!)
- Serverscript er forholdsvis ny og få servere stiller den til rådighed (?) (Ingen standard endnu derfor understøtter MS JavaScript til VBScript ASP programmering

# 3 Struktur

- Browser og server er to program (**tow-tier system**) (/ 3. kan være databaser)
- Bruger laver forespørgsel via URL
- Serveren opbygger svar som typisk er et HTML dokument
- Kommunikation foregår vha. standarder (regler for kommunikation: TCP/IP, HTTP
- Browser opfatter HTML og ASP ens – forespørg server og vis resultat

## 3.1 Installation af web server

- Da vi skal genskabe ovenstående struktur
- Personal Web Manager, eller Internet Information Services (fortolkere!)

## 3.2 Forskel på CGI og ASP

- CGI skal afvikles i en selvstændig operativsystem proces (tungt)
- ASP afvikles internt i serveren
- CGI kan skrives i et vilkårligt programmeringssprog
- ASP fortolkes via JavaScript eller VBScript
- CGI er mere standardiseret (ikke tvang med bestemt http-server)

## 3.3 Første termer

- <% %> er måden hvorpå ASP indsættes i HTML (= inline)
- Inline ikke altid smart men fantastisk praktisk
- <%= %> resultat oversættes til HTML
- Response.write dette generere HTML
- **<%=date%> og <%=time%> Viser dato og tid (server kører VBScript tim funktion!)**

### 3.4 Kommunikations protokoller (kap. 2, s. 59)
- TCP/IP (metode til at styre pakker rundt i et netværk) – indeholder data
  HTTP er en protokol til adressering via HTTP requsts og response)

### 3.5 Scripting (kap. 2, s. 63)
- Vi burger script for at muliggøre dynamiske sider
- Server-Side Scripting og Client-Side Scripting
  - eneste forskel er forberedelsen
  - vigtig mulighed
- Script identifiseres med <% %> samt <SCRIPT>

### 3.6 Server-Side Scripting (s. 66)
- To muligheder: <% samt <SCRIPT> ... **RUNAT=SERVER**
- Filnavne skal være af typen .asp
- **Caching:** Resultatsiden kan være gemt i memmory (s. 71)
- <SCRIPT RUNAT=SERVER> udføres til sidst!!! Mens <%%> udføres inline

### 3.7 Client-Side Scripting (s. 71)
- Overhovedet ikke relateret direkte til ASP
- Udfører ikke scriptet med downloader der og overlader det til browseren
- Fordele: Hastihed hurtigere, reducere belastning af server
- Bagdele: Afhængig af browserens funktionalitet – browser specifik, koden ikke skjult
- Mulighed: <SCRIPT> uden angivelse af RUNAT
- Filnavne skal være af typen .htm el. .html
- Placering (s. 73)
- Fejlmuligheder <SCRIPT> <!—Document.Write Date --> …

### 3.8 Order of execution (s. 79)
- Clent-side: Resultat bliver indsat hvor scriptet er indsat
- Server-side: s.81-84 <SCRIPT> svær at styre alt efter sprog
- <SCRIPT ...> bruges mest til procedurer og funktioner

## 4 Hvad muliggøre ASP (s. 85)
- ASP tilbyder objekter og komponenter vi kan bruge
- Objekter (7 stk.): Requst, Response, Server, Application, Session, ObjectContext, ASPError
- Komponenter: (s. 87) 10 med IIS 5.0 flere og andre i tree-tier systems
- Data Access component: MDAC ... Access
- Scripting Objekter: Dictionary, FileSystemObject og TextStream

### 4.1 Alternativer til ASP
- Client-Side: Java (applet), ActiveX og DHTML
- Server-Side: CGI, ColdFusion, JSP og PHP
  - PHP: Nyere server side scripting sprog … open-source

## 5 Oplysninger fra form (kap. 3, s. 97)
- En form oprettes med en ACTION (hvorhen ved SUBMIT?) og en METHOD (hvordan overføres dataene? post/get – kap.7)
- Indeni er der 3 forskellige <INDPUT> tags ved TYPE=: SUBMIT, RESET og TEXT (el. BUTTON)
- Editfelt – TYPE=”TEXT”
- Ved kørsel: 1. Browser tager information kendetegner ved navn, sender det til server sammen med request
- Informationerne gemmes i et Request objekt

- To teknikker for at vise værdier: HTML og ASP <% %> / dynamisk fra Request objektet:

Indput form:
<FORM ACTION=”xxx.asp” METHOD=POST>
<INDPUT TYPE=”TEXT” NAME=”strNavn
<INDPUT TYPE=”SUBMIT” VALUE=”Ok”>
<INDPUT TYPE=”RESET” VALUE=”Nulstil”>
</FORM>

Output form:
<%
Dim sNavn
SNavn = **Request.Form**(”strNavn”) ’hænger sammen med POST!!
Response.Write Snavn
%>
- Shortcut til Response.Write:
<%= Snavn%> ‘kun et variabelnavn
- Men ... preformance ... jo flere <% ... %> jo langsommere, flere kald til ASP DLL

**5.1 Request og Response Objects (kap. 7, s. 245)**
- To måder at håndtere kommunikation mellem browser og server på
- Request muliggøre at ASP kan bruge alle informationer sendt fra klienten (data, request, URL’s, cookies ...)
- Når server har brugt alle disse data vender vi tilbage til browser via Response
- Brugeren sender packaged med Request objektet
- Sende information fra form til server: FORM, name/value
- Sende information tilbage til klient: Response objekt
  - To måder: write og shoutcut Response.Write xxx / <%=xxx%>
- Preformance falder ved brug af shortcut pga. hop ind og ud af script

## 6 Request (s. 250)
- Indeholder: QueryString, Form, ServerVariables, Cookies og ClientCertificate

- **QueryString:** xxx.asp**?navn=Hansen&email=aa@bb.dk** (navne og værdier)
- ASP modtager værdier i et HTTP request og gemmer dem i en QueryString
  <%= Request.Querystring%>
  <%= Request.ServerVariables(”QUERY_STRING”)%>
- 3 måder hvorpå QueryString genereres:
1) <A HREF=”file.ext?name=value&email=a@b.dk”>et eksempel</a>
2) Når en FORM sendes til en server med GET
3) Når en FORM sendes til en server med POST
- GET: Sender informationer ved at tilføje til URL’en
  <%= Request.QueryString(”name”)%>
- POST: Sender informationer som en del af request body (muliggøre ikke QueryString se FORM!!)
- Antal parametre: <%=Request.QueryString(”name”).Count%>
- Benytter en bestemt: <%=Request.QueryString(”name”)(index_val)

- **FORM:** Indeholder værdier og variabler sendt via POST metoden
- POST: Sender informationer som en del af request body (Form collection oprettes kun ved POST metoden)
- GET/POST: Flere forskele ved **send**, den eneste forskel ved at **modtage** informationerne hos serveren er brugen af request objekt: QueryString eller Form
  <%= Request.Form(”namex”)%>

- Ved brug af FORM / POST får man ikke noget tilføjet URL'en!!

- **ServerVatiables:** Indeholder alle HTTP headers
- Ekstra information vedr. "folholdene" hvorfra de er afsendt (s. 262)

- **ClientCertificate:** Noget med sikkerhed

- Request("xxx") gennemsøger alle indholdsgrupper for noget af det navn (altså QueryString, Form, ServerVariables, Cookies og ClientCertificate)
- TotalBytes og BinaryRead – se s. 268

## 7 Response

- Bruges til at sende serverens output til klienten
- Request har mange informationer – Response har mange properties og metoder
- Hvad sendes tilbage?, hvornår?, hvor længe?, gå til anden side!, lav cookie!
  - Response.Write tilføjer til HTML output string
- **Buffed** kan angives for at fortælle sig selv at vi bestemmer hvornår data sendes tilbage
- **Flush** sender hvad der står i buffer til browser og arbejder videre med scriptet
- **Clear** tømmer bufferen
- **End** får serveren til at stoppe og sender det der er i bufferen til browseren
- Eksempel s. 274
- Kan også kontrolere hvad browseren gør!: Cache siden i en periode eller fortælle browser at den skal gå til en anden side
- Cache gemmes i temp, man kan angive for hvor lang tid, så siden ikke downloades hver gang!
- Man kan angive hvor lang tid informationen bliver i cache
- Redirection kan gøres på forskellige måder så det kan ses eller ikke ses!! (s. 279)

## 8 Variables (kap. 4, s. 127)

- Bruges til at gemme værdier
- Definer variabel, assign værdi til variabel, test på variabel, brug værdi i variabel
- En variabel er et stykke memmory
- VBScript: **Dim** erklære variable – men ikke variabel type, der er kun en type
- Variabeltypen gemmes i **variant** (= en basal tupe!, en type definere et værdiområde)
  - Numerisk **subtype** (s. 130): integer, byte, long, single, double, currency
  - String **subtype** String = "hej"
  - Date **subtype** Date = #11/01/2003# (brug med variabler s. 190) (var x = Month(now()) s. 195)
  - Boolean **subtype** TRUE / FALSE
  - **Specielle subtype** Empty = Uinitialiseret, Null = Tom (ingenting), Object, Error
- En variabel er altid af typen variant
- VBScript kan altid klassificere en variabel af typen streng eller værdi, derudover er der andre typer kaldt subtypes

- På alle variable kan man spørge efter typer: **TypeName**(xxx)
  - Res = Empty, Null, Integer, Long Integer, Single, Double, Currency, Date, String, Object, Error, Boolean, Variant, Byte, Array
- Navngivning: vigtig, sigende, ikke for kort, ikke for lang (max. 255 tegn), undgå symboler
  - Ikke case-sensitive I VBScript: abc = ABC
  - bln, byt, dat, dbl, err, int, lng, obj, sng, ste se s. 137

- Variabler skal erklæres med **dim** før de bruges
  - Explicit erklæring: 2 linier

    ```
    Dim strAbc
    strAbc = "hej"
    ```

  - Implicit erklæring: 1 linie, uden dim

strAbc = ”hej med”

- For at sikre at alle variable erklæres explicit kan skrives: **Option Explicit** (s. 138)

- Manipulation med variable:
  - Assignment ‘=’
  - Sammenlign = <> < > <= >=
  - Arithmetic beregning + - * / ^ MOD

- Logiske operatorer: AND OR NOT (der findes flere men kun I specielle situationer)
- Konkatinering af strenge: string1 & string2
- Ikke altid opfattes typen rigtig derfor benyttes typecasts (s. 145)

- Konstanter kan ikke ændres
- Const strBygning = ”Hus” (se mere s. 149)

- Scope: Hvor kan variablen benyttes?
  - Local = procedure level variables, lever så længe proceduren er i scope (procedure level)
  - Globale = script level variables

- Stringmanipulationer (s. 154)
- Ucase, Lcase, Len, Left, Right, Mid, InStr, Ltrim, Rtrim, Trim
- ex. mid(streng, fra position, antal tegn), InStr(str1, str2) – resultat er position eller 0!

- Array (s. 161)
- Dim NavneArray(49) ‘giver 50: 0,1,2…,49 = **fixed-size array**
- Dim ValArray() = **dynamic array**
  - Redim ValArray(5) ‘tildeler 6 pladser, eventuelt allerede udfyldte slettes
  - Redim **Preserve** VallArray(18) ‘eventuelt allerede udfyldte slettes ikke
- Multi demissionelt array Dim Marray(7,7)
  - Optil 60 mulige demissioner, kun 3 brugbare, derover er databaser brugbare

## 9 Kontrol strukturer (kap. 5 s. 169)

- Alle kodelinier kan deles op i action statement og control statement
- Control ‘kaldes’: flow control, execution order eller control structures
- Flow: rækkefølgen af udførsler
- Execution
- Action statement:
- Control statement
- Code structures: Flere liniers sammenhæng
- Control structures: Flere efterfølgende control statement

- Branching statement (if then else, select case)
- Jumping statement (sub procedure og funktioner som kan kaldes med **call**)
- Looping statement (for next, do while)

**9.1**

**9.2 If – evaluere et udtryk (s. 174)**

```
<%
if varB = “Yes” then Response.Write “Test1”
%>

<%
if varD = “hej” then
  Response.Write “Test3-1”
```

```
<%
if varC = 567 then
  Response.Write “Test2”
End if
%>
<%
if varE = 1 then
  Response.Write “Test3-1”
```

```
Else
  Response.Write “Test3-2”
End if
%>
```

```
Elseif varE = 2 then
  Response.Write “Test3-2”
else
  Response.Write “Test3-3”
End if end if
%>
```

**9.3 Case – evaluere ”flere” udtryk (s. 182)**

- If then bliver uoverskuelig ved flere end to muligheder
- Flere muligheder afgrænses med drow down box (dropbox)

```
<%
Select Case varF
  Case ”test1”
    …
  Case ”test2”
    …
  Case ”test3”
    …
end select
%>
```

```
<%
Select Case varG
  Case ”test1”
    …
  Case ”test2”
    …
  Case ”test3”
    …
  Case Eles
    …
end select
%>
```

```
<%
Select Case varH
  Case ”test1”, “test2”, “test3”
    …
end select
%>

<%
Select Case Ucase(varG)
  Case ”TEST1”, “TEST2”, “TEST3”
    …
end select
%>
```

- Boolske udtryk er også mulige

```
<%
Select Case varI
  Case 1,2,3,4,5,6,7
    …
  Case 8,9,10,11
    …
end select
%>
```

```
<%
Select Case varJ
  Case xyz <=7
    …
  Case xyz > 7
    …
end select
%>
```

**9.4 For Next – (looping) man kender antallet at gange, tjek ved start (s. 188)**

- For each udfører sætningerne for alle elementer i et array eller en collection
- Step og bagfra kan også benyttes (for i=16 to 2 step –2)

```
<%
for varI = 1 to 5
  ...
Next
%>
```

```
<%
Dim strCities(1)
StrCities(0) = “Aarhus”
StrCities(1) = “Skanderborg”
for Each item In strCities
  ...
Next
%>
```

**9.5 Do While – (looping) man kender ikke antallet at gange, tjek ved slut mulig (s. 194)**

- Loop kan ved fejl gå i selsving: infinite loop (de fleste servere vil stoppe udførslen)

```
<%
do while varI < constQ
  …
Loop
%>
```

```
<%
do
  …
Loop while varI = constQ
%>
```

```
<%
do Until varI > constQ
  …
Loop
%>
```

```
<%
do
  …
Loop until varI = constQ
%>
```

- Hvornår benyttes hvilken? Kommer an på eksempelet
- Inden i en lykke kan man også skrive **Exit do** (ex. if (a=b) then exit do)
- While wend ... benyt do while isteden

**9.6**

**9.7 Procedurer og funktioner (s. 200)**

- Når noget kode skal udføres flere gange kan man lave kodegenbrug med subprocedurer og funktioner

Subprocedure

- Mere end et muligt resultat får man ved at lade proceduren være afhængig af parametre
- Hvorfor: kodegenbrug, overskuelighed, læsbarhed, programmør kan umiddelbart se idé, position ikke vigtig og kald kan hurtig flyttes
- Brugbare for Response.Write og andre actions

```
<%
Sub ProcABC
  ...
End Sub
…
call ProcABC
…
call ProcABC
%>
```

```
<%
Sub ProcABC(navn)
  ...
End Sub
…
call ProcABC(“Hansen”)
…
call ProcABC(“Petersen”)
%>
```

Funktioner (s. 206)

- Retunere et resultat, beregner måske noget

```
<%
function BeregnX(var1, ar2)
  ...
  BeregnX = 47
End function
%>
```

## 10 Objekter, properties, metoder og events (kap. 6, s. 215)

- Objektorienteret programmering for at repræsentere den virkelige verden
- Et objekt indeholder properties, metoder og events

- Objekt: Ting vi ser på, samler op og bruger (navneord!)
- Objekt orienteret programmering
- Objekt = classes, men kan erklære og have en **instans** af en klasse
- En instans af en klasse er instansirret til af indeholde en af kopi
- En klasse kan have mange forskellige instanser
- Objekttypen er en definition af indhold (properties, collections, metoder og events)
- Objektest tilstand = givne variabelværdier, en metode kan ændre objektets tilstand.
- Objektets adfærd er defineret ved de handlinger som et objekt kan udfører

Properties

- Beskriver et bestemte element af objektet (private, public, protected, default ’ses inden pakken / java’)

Metoder

- Mulighed for at objektet kan udfører en funktion med eller uden parametre
- Metoder kan have returværdier som information til brugeren

Events

- Brugeren kommunikere med objektet vha. proparties og metoder

- Objektet kan også selv udføre noget det sker når en  event trikkes
- Når et objekt generere en event siges det at den **fire** en event, bruged må **handle** denne event
- Dette sker i stedet for hele tiden at lytte efter en event

- Synkron – alt udføres i en rækkefølge action efter action
- Asynkron – ting udføres separat når der er behov derfor (Dette sker i stedet for hele tiden at lytte efter en event)

- Objektorientering er indkapsling, bruger skal kun kende grænsefladen
- Implementation kan ændres uden bruger bemærker det

Brug af objekter
- Bruger er ansvarlig til at sætte givne properties
- Når objekter oprettes benyttes **set** xxx = ooo, da ooo returnere instans af xxx (s. 229)
- Derefter har vi en instans hvor properties fra metoder kan sætts/kaldes
  set objekt = Server.Createobjekt(”objekt”)
  objekt.property = xyz
  ...
  set objekt = nothing  ‘slut altid af med dette

Flere typer
- Indbyggede objekter: Automatisk tilgængelig i et asp script, eksistere kun i en version
- Scripting objekter: En instans af en objekttype, kan instansivres i flere konkrete objekter
- Komponent objekter: plug-in, brugeren kan udvide (forskel fra scriptting)

## 11 ASP objekt model (s. 236)
- Set af objekter og deres sammenhænge = objekt model
- 7 objekter udgør ASP: Server, Application, Session, Request, Response, ObjektContext, ASPError
- Disse udgør tilsammen ASP “systemet” (se s. 237)

### 11.1 Serverlevel:
Server
- Properties og metoder fra serveren selv
- Time out, virtuel path/fysisk path, url

Application
- Startes når første besøg kommer
- Lukker når sidste besøgende forlader siden
- Der kan være mange klienter til den samme applikation

Session
- Hver client til application har sin egen session
- Gør det muligt at gemmer information for brugeren
- Dette er den ”største” til brug ved web baserede applikationer

### 11.2 Individual page level:
Request
- Besked fra client til server (s. 240)

Response
- Den primære funktion for ASP er at generere dynamiske sider dette kan ske med Respons

ObjectContext

- Gør det muligt at bruge transaktioner
- Enten udføres det hele uden fejl eller også udføres intet
- Sikre ”samtidige transaktioner”
- Styre afviklingen af ASP script

ASPError
- Dette objekt indeholder detajler om fejl genereret af ASP sciptet eller asp.dll

## 12 Applications, Sessions and Cookies (kap. 8, s. 290)

- Indtil fornylig kunne informationer over flere sider kun gemmes som cookies, nu er der også application og session, alt dette tilføjer power og fleksibilitet til asp

Web Applikation
- Ligner en traditionel applikation
- Men web udgaven er **stateles** serveren fører ikke brugeren fra request til request
- HTTP er client drevet, clienten fortæller hvad den ønsker
- Serveren ved ikke hvad en given client har gjort før
- Så brug af HTTP er hurtig, der er ingen historie, serveren kan bare udfører opgaven hurtigt

- Men historik af tidligere hændelser kan være ønskværdigt (fx. ved login)
- Dette kan gøres på flere måder – cookies (”outdates ASP itself)

### 12.1 Cookies
- Blev introduceret som en metode til at identifficere hver forskellig bruger
- Tekstfiler som dannes af browser men informationer fra serveren gemmes på client
- Kan bruges til at holde brugeren informeret med relevant information
- Hver server kan sende cookies men browser må tillade det
- Cookies er ikke ”farlige”, de kan kun gemme informationer som bruger har indtastet frivilligt

- Requese objekt har cookies collection, med alle besøgende
- Der indlæses når en client kommer
- Cookien kan indeholde flere med samme navn ved brug af key
- Hver sit domæne kan IKKE aflæse cookies fra et andet domæne kun sit eget (el. Ved sub domæne)
- Browser sender cookie med til server

Brug (s. 293)
- Inden hvar forespørgsel afgør browser om cookie skal sendes med
- Er der cookies med samme navn (bruges den nederst i hieratiet) men derfor benyttes key
- Request.cookies(”cookiename”)[(”key”)].attributes ‘generel syntakt
  - Af attributer er der: Domain, expires, haskeys, path, secure (se note s. 2)
- Response.write Request.cookies(“**c**ookie**n**ame”) ‘vis en cookie
- <% Response.cookies(“cn”) = value %> ‘sætter en cookie
  - Følgende HTTP header genereres: Set-Cookie:CN=value evt. m. &... hvis flere
  - Derfor ’Response.Cookie’ før response body data anviges
- <% Response.cookies(“cn”)(“key”) = value %> ‘sætter en cookie med en key
- Hvis derefter: <% Response.cookies(“cn”) = value %> overskrives alle key værdier
- Hvis igen: <% Response.cookies(“cn”)(“key”) = value %> slettes ovenstående
- If Request.cookies(”cn”).HasKeys then ‘s. 294
- En cookie slettes igen når session stopper eller browser lukker, hvis ikke:
  - <% Response.cookies(“cn”).Expires = “July 4, 2001” ‘angiver udløbsdato
  - <% Response.cookies(“cn”).Expires = Date + 1 %> ‘angiver udløbsperiode
  - <% Response.cookies(“cn”).Expires = Date - 100 %> ‘sletter cookies

Server
- Properties og metoder fra serveren selv
- Time out, virtuel path/fysisk path, url

### 12.2 Application (s. 301)
- Application = alle filer I et givet virtual directory. Kun et application object for en applikation
- Startes når første besøg kommer
- Lukker når sidste besøgende forlader siden
- Der kan være mange klienter til den samme applikation
- Gemmer informationer der kan læses af alle klienter, variable og objekter i **application-scope**
- For at holde styr på dem skal de erklæres i global.asa
- Holder også styr på alle sessions
- Applications object har to collections:
  - Contents: variable ‘Application.Contents(“key”) eller Application(“key”)
    - Se liste af contents variable s. 305
    - Har to metoder: Remove / RemoveAll (se s. 306)
  - StaticObjects: objekter ‘alt oprettet som burger tag <OBJECT>
- Applications object har to metoder:
  - Lock og UnLock: Kø, kun kører en ad gangen kritiske systemer

### 12.3 global.asa
- Script blok
- Ikke omgivet af <% %> men <SCRIPT LANGUAGE=”VBScript” RUNAT=”Server”>
- Hver applikation kan kun have en global.asa
- Heri kan man inkludere event handlers til application eller session
- Kan være besværlig at bruge se s. 311
- Problem: Ikke så stabil og god på ekstern server, ved genstart tømmes oplysningerne
- Hvis VBScript benyttes har den 4 subrutiner:
  - Application_OnStart ’bliver kun affyret ved den første besøgende (EX: DB INIT!!)
  - Application_OnEnd ’når sidste session slutter. Manuel nedlukning via IIS (DEL REC)
  - Session_OnStart (s.321) ’hver gang ny bruger begynder en session
  - Session_OnEnd ’hver gang der sker time out eller abandon (ikke at browser lukkes!)

```
sub Application_OnStart
    :          ’erklære applikations variable
end sub
```

### 12.4 Session (s. 312)
- En applikation kan have et uendeligt antal sessions kun begrænset af memmory
- Muliggøre at styre brugeren gennem siderne
- Starter når en bruger uden en session åbner en asp side
- En session kan blive nedlagt på to måder: Session.Timeout (normalt 20 min.) el. Session.Abandon
- Brugbar ved en butik eller til at identificere autoriserede brugere
- Muliggøre at identificere start og slut på session samt gemme informationer
- Session object har to collections:
  - Contents: variable ‘Session.Contents(“key”) eller Session(“key”) – brug s. 314
    - Se liste af contents variable s. 314
    - Har to metoder: Remove / RemoveAll (se s. 316)
  - StaticObjects: objekter ‘alt oprettet som burger tag <OBJECT>
- Session object har fire properties:
  - SessionID: read only s. 315 <%=Session.SessionID%>
  - Timeout <% Session.Timeout=10%> ‘server kan ikke se når burger forsvinder (s. 315)
  - CodePage: s,. 316 – kun brugbar man skal bruge non-Roman alfabet
- LCID: time-zone og sprog indikator
- Session object har en metode:

- Abandon: Nedlægger session (svart ved test s. 317)

Hvorfor så cookies?
- SessionID gemmes i en cookie
- Uden Cookies vil session ikke virke
- Cookies kan også gemme information mellem sessions

## 13 Error Handling (kap.9, s. 331)

- Svært at lave fejlfri software ... umuligt
- Vigtigt: Læsbarhed, forståelighed og ensartet abstraktionsniveau
- De fleste programmeringssprog benytter sig af en debugger
- Men for at minimere må man holde sig til gode programmeringsteknikker
- Forstå koden og hvor der er potentielle fejlmuligheder

- Brug objekter korrekt (client kontra server)
- Client fejl giver en popup dialog, server fejl retunere HTML kode (s. 335 / 339)

Hvor placeres koden?

```
<%
  'server-side script here
%>

<SCRIPT LANGUAGE=VBScript RUNAT=Server>
  'server-side script here
</SCRIPT>

<SCRIPT LANGUAGE=VBScript>
  'client-side script here
</SCRIPT>
```

- Typer af fejl:
  - Syntax error: (s. 336) stavefejl, end if / endif, forkert propert, benytter properties som metoder og omvendt, skriver til en read only variabel, instiansere objekt med SET (... alt programmering skal være så præcist)
  - Logical errors: ex. div by 0, type mismatch, no error message but output wrong, kommunikation mellem projektdeltagere (cm / inct)
  - ASP errors: Noget går galt I operativsystemet

- Hvorfor fejl?
  - Programmører er tids pressede
  - Test kræver forberedelse og organisation (tidskrævende)
  - Tests garantere ikke for fejlfrie programmer

### 13.1 Gode principper (s. 341):

- Indryk koden
- Kommenter koden
- Brug 'Option Explicit', variabel erklæring på forhånd (Kun ved test pga. det belaster serveren)
- Brug subprocedures (kodegenbrug, optimere koden)
- Brug 'include files' (pas på ved tilretning, bruges flere steder?)
- Typekonverter dine variable inden brug (s. 343) – Cint, Cbool, CStr ... (logisk fejl)
  - Tjek variable – IsNumeric, IsDate ...
- Brug navne regner – strX, intX… (variable med småt, objekt med stort begyndelsesbogstav)
- Afprøv koden med ønsket indput, min/max (grænser), uønsket indput (3 kontra tree!)
  (Testere: MS – programmere om dagen og tester om natten)

- Brug af blanke linier
- Brug konstanter (Const intAntal = 7)
- Brug ikke selvdefinerede forkortelser til variable, metoder og parametre
- **Moduler: Adskil modtagelse af input, validering, beregning og behandling af output**
- Opsplit i små fungerende enheder
- (Arbejd på sprogs primisser! Ex. indryk)

## 14 Scripting Objects (kap. 10, s. 377-387)

- Objekter som application og session er en del af ASP, disse indbyggede objekter har men kommunikationen mellem server og klient at gøre
- Der også nogle objekter der ikke har med kommunikationen at gøre men som tilføjer funktion til scripting sproget selv = **scripting objects (bl.a. Dictionary og RegExp)**
- Med asp er der mulighed for at bruge VBScript og Jscript objekter
- Disse objekter tilføjer funktionalitet
- Scripting objekter kan både bruges på server og client, modsat ASP objekter som kun er mulige på serveren. Client er ting som: farve, opløsning, system (dokument), hvorimod man på serveren har fil adgang.

- To set af scripting objects der både kan bruges på klient og server (s. 379)
  - Scripting Runtime Object: **set** objDic = CreateObject("Scripting.Dictionary") – generic object
    - bruges som nem made at gemme informationer I en enkelt datastruktur (ordbog)
  - VBScript Object: **New** set objRegExp = New RegExp 'bruges til søgning og manipulation af tekststreng, stor mængde sekvens af tegn (RegExp kom først senere ASP3/4?), søg og erstat (script object benyttes med class og new)

- CreateObject går gennem COM objekt og New går via VBScript runtime engine (COM er en fordel!)
- Udvikel som script. Når færdig og gennemtestet lav DLL/COM, som er hurtigere (er "kompileret") men sværere at rette, men beskyttede

- Gør det muligt at udvikle egne komponenter, og hvis de kan installeres på webserveren så kan man tilgå dem som COM objekter vha.: SET <var> = server.CreateObject("Navn")
- Udbredt til upload og grafikmanipulering på server

- **Gør det muligt at trække COM objekter via ASP sider ... både VBScript og Tscript (.. men kun hvor de er installeret typisk serversiden**

- Problemet med komponenter er sikkerheden, kan lægge server ned, får sjældent lov
  - Det er der vist nok taget højde for i ASP.NET hvor komponenter er indkapslet

### 14.1 Brug af Dictionary s. 380-387

- **Hvorfor** gemmes dictionary ikke i en Session? Fordi det sløver den proportionalt med antal besøgende og kan lægge den ned (fejl da dictonary oprindeligt var tænkt som værende kun til klient siden)
- Gør det muligt at gemme og modtage information i en fleksibel datastruktur, refereres via key
- I stedet for 2-dim-array, id og værdi sammen
- God til forms da dictionary kan overføres mellem siderne

- (FileSystemObject: Manipulere filer på serveren, **upload formulaer**, giver adgang til hele filsystemet der hvor det afvikles ud fra sikkerhedsniveau
- (TextStream: Læse filer af ren tekst)

## 15 Databaser

- Access: God til udvikling, langsom ved mange brugere men mulighed for upload DB

  - Optimering: Dan html fil på serveren, og send den derefter færdig fil til client
- MS SQL, Oracle: Dyr og ingen fil (create table osv.)
- My Sql: Ikke stor nok, Access2000 er at foretrække (??? Fil/create)

## 16 ASP og Data Store Accass (kap. 12)

- ASP Request er ideel til at fange og samle temporere data
- Mange gemmer information i **databaser, emaile, dokumenter** hos slutbruger
- Fælles data var en drøm, Universal Data Access udtænkte en strattegi
- Drømmen påvirker ASP fordi vi kan bruge OLE-DB via interfacet ActiveX Data Object (ADO)
- Det gør hjemmesider up to date / Active

### 16.1 Databaser

- En standardiseret måde at gemme information på
- Normalisering / undgår store ene informationer
- Gemmes i record og tabeller: selve indholdet
- Hver record refereres entydigt vht. En nøgle / key
- Data Store: Struktureret data, tabel, record, key – enhver persistent samling af information

- Standardiseret måde at tilgå Database typer (Access, Oracle, ...) på?
  - Indtil for nylig var det bedste svar ODBC
  - Men bedre er OLE-DB

### 16.2 ODBC

- Open Database Connectivity, standart for at tilgå data
- Indeholdt rutiner til data access uanset DB typen
- ODBC fælles lag ovenpå oracle, access og sql server hvorfra ens kommunikation til program kunne foregå (man skulle ikke bekymrer sig om hosting men om brugen)
- Så havde man adgang til DB via ODBC kunne man manipulere data

- ODBC ikke nok
- ODBC kan ikke hjælpe med af samle information fra forskellig format (db, emails, doc.,...)
- OLE-DB har mulighed for at samle information på tværs af grænseflader

### 16.3 OLE-DB

- Mere generisk end ODBC, hurtigere, nemmere at bruge
- Samme ide som IDBC med kan kommunikere med flere formater (db, emails, dokumenter)
- OLE-DB introducere data providers og data consumers (s. –469)
  - Data provider: Noget der stiller data til rådighed (mange ... db, excel, email, ...)
  - Data consumers: Noget der bruger dataene (bruger dataene, vores ASP sider – ADO object)
- OLE-DB lavniveau så gøres tilgængelig med AOD (ActiveX Data Objects)

(- Så med denne frihed kan man koncentrerer sig om funktionaliteten)

### 16.4 ADO

- AOD = venligt interface (til OLE-DB) der muliggør at ASP taler med OLE-DB
- Så når vi bruger ASP til at referere Data Store, er det faktisk ASP og ADO
- ADO højere niveau end OLE-DB

### 16.5 ADO er ikke en del af ASP

- ADO er en ideel måde at access data fra ASP sider
- ADO kan også bruges med Java, c++, - et hvert COM kompatibelt programmeringssprog
- Objektmodellem til ADO og ASP er vidt forskellig
- ASP muliggøre dynamiske hjemmesider, ADO muliggøre brug af data store på dynamiske
- ADO 2.5 tilføjer objekter som Record og Stream
- ADO objekt model s. 471

- Connection – link mellem program og data store
- Command – kører komandore op imod data store
- Recordset – Indeholder et set af data
- Record
- Stream – muliggør manipulation af data

### 16.6 Hviklen database?

- Access: God til udvikling, langsom ved mange brugere men mulighed for upload DB
  - Optimering: Dan html fil på serveren, og send den derefter færdig fil til client (*.mdb)
- MS SQL, Oracle: Dyr og ingen fil (create table osv.)
- My Sql: Ikke stor nok, Access2000 er at foretrække (??? Fil/create)

### 16.7 Eksempel i ASP:

```
Dim objConn, objRS, strConnection, sPath
Dim adOpenForwardOnly, adLockReadonly, adCmdTable

adOpenForwardOnly = 0
adLockReadonly = 1
adCmdTable = 2
Set objConn = Server.CreateObject("ADODB.Connection") ‘Opretter en ADO forbindelse
‘Programmets identifier (eller ProgId) til ADO forbindelsen er ADODB.Connection (objConn)
Set objRS = Server.CreateObject("ADODB.Recordset")

sPath = Server.MapPath("/asp/uge40/Telefonliste_2000.mdb") 'For at slippe for absolut sti
Response.Write sPath

strConnection = "Provider=Microsoft.Jet.OLEDB.4.0;" & _
"Data Source=" & sPath
objConn.Open strConnection ‘aktiverer en Access database (åbner databasen)
’kun en parameter, andre antager default værdier, alle properties kan vises via eksempel s. 496. Istedetfor ... ConnectionString s. 493
objRS.Open "Personer", objConn, adOpenForwardOnly, adLockReadonly, adCmdTable

Response.Write "</br></br>"
Response.Write "<table border=""1"" width=""100%"">"
While not objRS.EOF
  Response.Write "<tr>"
  Response.Write "<td>" & TomtFelt(objRS("Fornavn")) & " " & TomtFelt(objRS("Efternavn")) &
"</td>"
  Response.Write "<td>" & TomtFelt(objRS("Adresse")) & "</td>"
  Response.Write "<td>" & TomtFelt(objRS("Postnr")) & " " & objRS("By") & "</td>"
  Response.Write "<td>" & TomtFelt(objRS("Tlfnr")) & "</td>"
  Response.Write "<td>" & VisSomEmail(objRS("Email")) & "</td>"

  Response.Write "</tr>"
  objRS.MoveNext
Wend
Response.Write "</table>"

objRS.Close
objConn.Close          ‘luk forbindelsen, fjerner ikke fra memmory så open kan kaldes igen med andet indhold af connection string
Set objRS = Nothing
Set objConn = Nothing          ‘Fjerner det fra memmory
```

1. **Opret forbindelse**

2. **Vis data**
3. **Luk forbindelse**

### 16.8 Connection

- Linker ADO med data store (telefoneksempel s. 479), overføre data imellem (ADO = ActiveX Data Objects)
- Data Store taler med OLE-DB data provider
- Det kan bruges overalt (den samme connection) så længe den er i memmory / ikke lukkes
- Kan oprettes med yderligere information vedr. data stor, hvor, hvilken part af data
  - 3 måder: Connecting string, Data Link Files, Data Source Names

#### 16.8.1 Connecting string (s. 480-484)

- Tekststreng med de vigtigste informationer, typisk: OLE-DB provider, driver (ODBC type), file name/data source (fusisk path og navn), navnet på databasen, bruger id, password
- Sværste men med flest muligheder
- Kan lægges ud i en fil dbConnect.asp (god ved flytning!, gøres med **Server-Side include** (SSI)
- Afhænger af hvilken type data man prøver at tilslutte

(- God på ekstern server)

#### 16.8.2 Data Link Files (s. 485-488)

- Undgår at angive specifikke type ting, - hvis man er usikker
- Opret UDL fil vha. fil properties

#### 16.8.3 Data Source Names (s. 488-

- DNS en anden made hvor man undgår de specifikke ting – meget simpel
- Udgået metode, bruger ODBC så man får ikke de nye muligheder som med OLE-DB
- Hurtigere og mere effektiv
- Gemmer information i en identifier som man så bruger i ASP koden – objConn.Open ”DSN=Adr”

- ADO bruger Connection til at gemme information om data store connection
- Man kan have flere connection til flere databaser
- Ikke alle data store providers er lige kraftige

Indeholder

- Propperties collection (s. 494), sættes med
  - objConn.Open strConnection, eller
  - objConn.ConnectionString = strConnection
    objConn.Open
- Error collection (s. 498)
  - single failuer
  - Når en ADO operation generere en error, tomes listen først inden ny indsættes
  - Kun flere når den samme linie generere flere fejl

## 17 Recordsets (kap. 13, s. 509)

- Data kan overføres mellem data store og ASP sider i begge retninger
- Figur af recordset objekt s. 510. **Recordset er i midten af ADO object diagram**
- Record er en stump data, recordset er flere stumper data
- Men mere end set af relaterede data: Kan manipulere dem med **add, remove, hide**
- Hver aktiv recordset har en **cursor** (pointer), peger altid på en record i settet
  - Recordset metoder: MoveFirst, MoveLast, MoveNext, MovePrevious
- Oprettes med
  Dim objRS
  Set objRS = Server.CreateObject("ADODB.Recordset")

- Når vi skal have data fra data store ind i et recordset:
  **objRS.Open ”tabelnavn”, objConn**
- At oprette objConn er dyrt så det anbefales at det kun sker en gang og så bruge den hver gang (explicit connection)
- Vi kan specificere connection detaljer direkte gennem recordset (s. 514) undgår explicit connection (Pga. ADO’s flade hierarki)
  strConnection = ”Provider=Microsoft.Jet.OLEDB.4.0;Data Source=C:\db.mdb
  **objRS.Open ”tabelnavn”, strConnection**

- Derudover kan tilføjes 3 konstanter der fortæller noget om typen af recordset vi ønsker
  - 2 instillinger af recordset: **cursor type** og **lock type**
  - 3. (og samlet 5. parameter): definere datatypen ex. tabel
- Konstanterne hertil samt sigende navne er defineret i msado15.dll, derfor

<!-- METADATA TYPE="typelib"
FILE="C:\Program files\Common Files\System\ADO\msado15.dll" -->

  - Disse kan ses I filen: adovbs.inc, denne kan også inkluderes:

<!-- #INCLUDE FILE= "C:\Program files\Common Files\System\ADO\adovbs.inc" -->

  - Ikke godt da filen bliver stor

### 17.1 objRS.Open source, connection, cursor, locking, type

**source** (hvorfra kommer data) (s. 521)
- SQL, Stored procedure / query el. ADO command
- Istedetfor 1. parameter kan objRS.Source= også angives

**connection**
- strConnection (numere of querys) eller objConn (only single query)
- Istedet kan man bruge et connection objekt og så skrive navnet på dette ex. objConn
- Istedetfor 2. parameter kan objRS.ActiveConnection= også angives

**cursor** (s. 522)
- Cursor type = pointer til en bestemt record i settet, flere forskellige alt efter funktionalitet:
( - Updateable vs. Non-updateable (non giver en preformanceforøgelse) )
  - Scrollable vs. Non-scrollable (non kun fremadsøgning, giver en preformanceforøgelse
  - Keyset vs. Non-keyset (keyset retunere kun nøglen (pref.), non-keyset også alle data
  - Dynamic va. Static (om recordsettet afsplejles (static) på kalletidspunktet eller ...)
- ADO cursor types: odOpenForwardOnly, adOpenStatic, adOpenDynamic, adOpenKeyset
- Istedetfor 3. parameter kan objRS.CursorType = adOpenForwardOnly (mest ved. ASP)

**locking**
- Tæt på angivelse om recordsettet er updateable,
  - daLockReadOnly: non-updateable derfor ingen lock
  - daLockPessimistic: updateable, låses så snart der editeres
  - daLockOptimistic: updateable, låses ikke den første der update vinder
  - daLockBatchOptimistic: ændre flere records og så opdatere dem samtidig
- Istedetfor 4. parameter kan objRS.LockType = adLockReadOnly (ved visninger)

**Type / Option**
- Fortæller recordset hvilken form data source vil have
  - adCmdText: Tekst ex, en sql kommando
  - adCmdTable: Navn på tabel
  - adCmdStoredProc: Stored procedure eller query
  - adCmdTableDirect: Navn på tabel
  - adCmdFile: Filnavn
  - adCmdURL: Url
  - Istedetfor 5. parameter kan ... nej henviser til command objektet

### 17.2 Brug af recordset objektet

- BOF og EOF properties, hvis BOF = EOF er recordsettet tomt
- 4 nemme metoder: objRS.MoveFirst, objRS.MoveLast, objRS.MoveNext, objRS.MovePrevious

- objRS.Move nummer, hvorfra
  - objRS.Move 2: 2 frem fra aktuel position
  - objRS.Move -3: 3 tilbage fra aktuel position (forsigtig s. 529)
  - objRS.Move 2, Start: 2 frem fra start

### 17.3 Bogmærker
- Huske position på bestemte record i recordsettet
- To forskellige bookmarks kan pege på to forskellige eller den samme record
- Ikke alle recordsets understøtter bookmarks, tjek efter med objRS.Supports(adBookmark)
- Kan bruges sammen med move: objRS.Move 2, bookmark
- Predefinerede bogmærker: adBookmarkCorrent, adBookmarkFirst, adBookmarkLast

### 17.4 Finde records
- objRS.**Find** eller objRS.**Filter**
- Find god når en bestemt skal finder, peger på en bestemt record
- Filter god når et set skal findes, går bestemte resords synlige og usynlige i vores set

#### 17.4.1 Find
- Kan scanne store recordsets så vi ikke skal gå gennem disse
- **objRS.Find kriterium, skip, søg, start**
  - Kiterie: Sammenlignings streng (>, <, =, LIKE)
    - objRS.Find ”Titel = ’Pulp Fiction’ ” ‘ikke case sensitive, teksten indeholder husk ’
    - objRS.Find ”Antal = 47 ” ‘ikke ’
    - objRS.Find ”Dato = #10/23/98# ” ‘husk # ved datoer
    - objRS.Find ”Titel LIKE ’Pulp*’ ”
  - Skip: Startposition, spring nogle af de første over (default: 0)
  - Søg: adSearchForward, adSearchbackward (default: forward)
  - Start: Startposition via bogmærke (default: adBookmarkCurrent)
- Hvis søgningen går godt placeres cursoren på den rigtige position, EOF hvis ingen fundet og BOF hvis ingen fundet og baglæns søgning

#### 17.4.2 Filter
- objRS.Filter = ”Director = ’Quentin Tarantino’ ” ‘Skjuler nogle record
- objRS.Filter = adFilterNone ’Gør dem synlige igen

### 17.5 Felter i en record
- objRS(”Titel”)
- while not **objRS.EOF**
  for each fld in **objRS.Fields**
  res = **fld.Value**
  Next
  ObjRS.MoveNext
  Wend

Count
- objRS.RecordCount ‘Virker kun ved Cursortype adOpenStatic

## 18 Advanced DataHandling Techniques (kap. 14, s. 563)
- Andre muligheder end recordset og connection, ex. command (i ADO 2.5)

### 18.1 Command object
- I stedetfor at instruere data store om at vi vil lave noget med data så fortæl hvad vi vil lave
  - select, insert, update, delete
- Muliggøre komplekse operationer: stored procedures og parametre

- set objCommand = Server.CreateObject("ADODB.Command")
  objCommand.Execute action, parameter, option
  - Action: select, insert, update, delete – 1. parameter retunere int for hvor mange berørt
  - Parameteres: Mulighed for at angive parameter til generiske kommandoer
  - Option: Hvorpå? Tabel, fil, …
  - Execute retunere et fysisk ADO recordset
- Få fat i result at: set objRS = objCommand.Execute

- Command, hvordan? hvad vil vi? (tabelnavn, sql, ...
  - Specificer objCommand.CommendText: Tabelnavn
  - Specificer objCommand.CommendType: adCmdTable
- Specificer objCommand.ActiveConnection for at angive connectionen
- ex.

objCommand.CommendText = "Film"
objCommand.CommendType = adCmdTable
objCommand.ActiveConnection = strConnection
objRS.Open objCommand

- Pga. flat hierarki kan execute også udføres på et Connection objekt
  set objRS = objConn.Execute CommandText RecordAffected, Options
- Basal sprog for at lære mere om disse typer af kommandoer: SQL

SQL
- Structured Query Language, universalt sprog for at programmere databaser
- Deklarativt sprog: Siger hvad vi vil modsat procedurelt
- objRS.Open = SQL, strConnect
- Operationer: SELECT, INSERT, UPDATE, DELETE
  - Select retunere et record set, insert, update og delete er action kommandoer

Select s. 571
- Fleksibel måde at udtrække data på
- Select * from tabel ... bedere at være præcis pga. preformance Select navn from tabel
- SELECT <felter> FROM <tabel> WHERE <condition>
- Fra flere tabeller, mange måder:
- Inner join (samle to tabeller i et recordset)

**SELECT** Film.Tittel Instruktør.Navn
**FROM** Film **INNER JOIN** Person **ON** Film.filmid = Instruktør.filmid
**WHERE** (Film.Navn LIKE 'Seven' OR Instruktør.Navn LIKE 'navn')

Update s. 579
- Opdatere en records værdi
- UPDATE <tabel> SET <felt navn> = <value> WHERE <condition>
- Lykkedes det og på hvor mange kan oplyses i 1. parameter
  - objCommand.Execute <RecordsAffected>

objCommand.Execute intAntal
Response.write "Antal poster der blev berørt = " & intAntal

**18.2 Stored Procedures / queries 582-**
- Select og update giver os meget fleksibilitet, men nogle udtræk udføres tit!
- Hvis dette kan udføres i selve databasen opnår vi:
  - Vinde preformance
  - Koden bliver mere læsevenlig
  - De kan bruges på mange ASP sider

objCommand.ActiveConnection = strConnect
objCommand.CommandText = navn på queries / stored procedures

objCommand.CommandType = adCmdStoredProc
set objRS = objCommand.Execute <antal>

- Access: Queries
- SQL server: Stored Procedures
- Med disse kan man bruge: parametre
- Parametre overføres med ADO's parameter collection (kun vha. command objekt)
  objCommand.ActiveConnection = strConnect
  objCommand.CommandText = navn på queries / stored procedures
  objCommand.CommandType = adCmdStoredProc
  set objParam = objCommand.CreateParameter(name, type, direction, size, value)

### 18.3 Modifying data med RecordSet

- Tilføje data vha. recordset med AddNew og Update s. 595
  objRS.MoveLast
  intIdForNewObject = objRS("MovieID")+1
  objRS.AddNew 'og flytter cursor
  objRS("MovieID") = intIdForNewObject
  objRS.Update
  objRS.Close
- Update er næsten det samme – med AddNew kaldes ikke
- Batch kørsel kan bruges vha. LockType = adLockBatchOptimistic
- Fortryd kan ske inden kald af update eller move*
  - objRS.CancleBatch adAffectAll
- Slette: objRS.Delete

### 18.4 **Modifyint data med SQL** s. 602

- INSERT INTO <tabel> (<fields>) VALUES (<values>)
  - Vigtigt at fields og values er I same rækkefølgåe
  - INSERT INTO Medarbejder (navn, lon) VALUES ('hansen', 25000)
  - Det vil fejle hvis ikke alle nødvendige felter er udfyldt
- DELETE FROM Medarbejder WHERE Navn = 'hansen'

- Til at selecte, indsætte, slette er brugervalg brugbare

### 18.5 Non database data store

- Record object: Repræsentere en record (s. 515)
- Stream object: Repræsentere en data binært eller ren tekst (s. 623)

## 19 Objekter i ADO

| | | |
|---|---|---|
| - | Connection (Errors) | kap. 12 |
| - | Recordset | kap. 13 (/14) |
| - | Command (Parameters s. 585) | kap. 14 s. 563 |
| - | Record (Fields) | kap. 14 s. 614 |
| - | Stream | kap. 14 s. 623 |

## 20 Eksempler

**1**

s. 16 Gettings Dato og klok, select case
Punctual Time

**2**

s. 69 Dataconf1 Date, <SCRIPT>, sidst
s. 70 Dataconf2 Date, <% %> 3+4 (client)
Udførselsrækkefølge

**3**

s. 110 Spring Form, POST
4 **Typer og procedurer**
**5** if/then, select/case, do/while, …
**6** Objekter, telefon,
**7 Request og Response**
Response.write
Reguest.Querystring
8 Application, sessions and cookies
s. 295 login Loginform
s. 304 application global.ass
s. 318 sessions overføre data fra form til form
9 Error Handling (s. 331-348)
s. 347 Nummertjek (dbl / int)
s. 350 Debug
10 Scripting Object (s. 377-387)
s. 370 SimpleDic Dictionary, ændre s. 382, modtag s. 385
**11 ASP components (kusorisk)**
12 ASP og Data Store Access
**s. 475 Connect Vis data**
**Using SSI og link DNS**
s. 495 ConProp Properties Collection
s. 499 Erroes Collection
13 Using Recordset
s. 515 Opret et recordset
s. 530 Brug af recordset
s. 541 Brug af find
s. 547 Brug af Filter
s. 522 Brug af Fields Collection
s. 556 Brug af GetRows metoden (recordset kontra array)
**14 Avanceret DB**
s. 566 Brug af Command
s. 572 Command og SQL udtræk
s. 586 Parameter Collection med Access
s. 595 Adding new record (recordset)
s. 601 Deleting records (recordset)
s. 603 Insert og delete med SQL (insert, select og delete)
s. 606 Brugervalg ved arbejde med recordset (cb søg felt og indhold)
s. 615 Record object
s. 623 Backup med stream object.

Udfyld combobox fra database: s. 534
Hvilket felt er udfylde: s. 607