MÄLARDALENS HÖGSKOLA
Institutionen för ekonomi och informatik

ASP 3 för komponenter

EI0230 – Komponentbaserad applikationsutveckling

Björn Persson, december 2000

(Denna sida har avsiktligt lämnats tom.)

Innehålsförteckning

Om detta kompendium

Detta kompendium är inte uttömmande i ämnet ASP eller komponenter i ASP. Se litteraturförteckning för böcker i ämnet.

Exempelkod i detta kompendium kan ha förkortats jämfört med de exempel som kan granskas eller laddas ner från kurshemsidan för kursen. Detta har gjorts för att göra koden mer lättläst och kompendiet kortare. HTML-taggar har ibland även skrivits med småbokstäver istället för stora (som man brukar göra som standard) då koden har kopierats direkt från FrontPage 2000.

För att lättast hitta information om objekt/metoder i VBScript och JScript så söker man på VBScript eller JScript och väljer ”undergruppen” *language reference* i hjälpen för Visual Studio (MSDN).

Vi är givetvis tacksamma för alla konstruktiva synpunkter på kompendiets utformning och innehåll.

Eskilstuna, december 2000

Björn Persson, e-post: `bjorn.persson@mdh.se`
Mälardalens högskola
Institutionen för ekonomi och informatik

1. Active Server Pages

1.1. ASP-kod och ASP-filer

ASP-sidor bygger på att man skriver ASP-kod som tolkas (interpreteras) av webbservern och som oftast returnerar vanlig HTML-kod (ett HTML-dokument) till besökarens webbläsare. Om besökaren använder Microsoft Internet Explorer (version 4.0 eller senare) kan man även utnyttja sig av fler objekt i HTML-dokumentet, t.ex. tabeller i form av ADO-objektet Recordset. Koden kompileras inte förrän den exekveras (tolkas), d.v.s. när ASP-filen begärs av en webbläsare, vilket kallas **sen bindning**[1]. Eventuella fel i koden upptäcks därför inte förrän vid exekvering.

Filer med ASP-kod använder oftast ändelsen `.ASP` för att webbservern (IIS) ska veta att den ska tolka ASP-koden innan den skickar HTML-dokumentet (d.v.s. resultatet av att ASP-koden exekverat) till besökarens webbläsare. ASP-filer innehåller vanlig HTML-kod där ASP-koden bäddats in mellan det inledande ”`<%`” och avslutande ”`%>`”.

IIS stödjer vid installation de två skriptspråken VBScript och JScript för ASP-kod m.h.a. inbyggda tolkar för dessa språk. Standardspråket är VBScript men det kan ändras m.h.a. inställningar i IIS eller med s.k. direktiv (se längre ner). Bägge dessa språk är enklare varianter av språken Visual Basic respektive Java. Det ska nämnas att JScript inte riktigt har samma syntax som JavaScript som används för skript på klienter såsom Netscape Navigator. I detta kompendium kommer vi att använda oss av främst VBScript, d.v.s. om inget annat anges så avses VBScript.

1.2. Ett inledande exempel

Nedan visas ett enkelt exempel (i filen `aspex01.asp`) hur man bäddar in ASP-kod i HTML-kod. Exemplet syftar till att visa hur ASP-kod kan se ut och i exemplet är koden skriven i VBScript. Först deklareras en variabel `i` som räknare för en loop. Loopen utförs 10 gånger och i varje loop skrivs räknaren ut i HTML-dokumentet. För att skilja på resultatet från ASP-kod och vanlig HTML-kod används två horisontella linjer (`<hr>`) i exemplet.

```
<html>
<head><title>Ett inledande exempel med ASP</title></head>
<body>
<h1>Ett inledande exempel med ASP</h1>
<p>Denna ASP-sida loopar från 1 till 10 och skriver ut talen en per rad.</p>
<hr>
<%
  Dim i
  For i = 1 To 10
    Response.Write i & "<br>"
  Next
%>
<hr>
</body>
</html>
```

1.2.1. Skriva till det resulterande HTML-dokumentet

För att skriva text, eller text sammanfogad med variabler, till ett HTML-dokument använder man metoden **Write()** i ASP-objektet **Response** (mer om objektet längre fram). I exemplet

[1] Motsatsen till sen bindning är **tidig bindning** vilket innebär att fel i koden kontrolleras vid kompliering.

ovan skriver vi värdet på variabeln och HTML-taggen
 (som sträng) genom att sammanfoga dem med &-tecknet.

```
   'Början av kod
Response.Write i & "<br>"   'Skriver ut variabel och HTML-taggen för ny rad
   'Resten av koden
```

I exemplet nedan visas hur man skriver ut endast en variabel (i loopen) som text samt hur man kan sammanfoga stränglitteraler och variabler till en sträng som skrivs ut.

```
Dim i, strFornamn, strEfternamnBPN, strEfternamnBAN
i = 0
strFornamn = "Björn "
strEfternamnBPN = "Persson"
strEfternamnBAN = "Andersson"

For i = 1 to 2
  Response.Write i    'ASP konverterar till text automatiskt
Next

Response.Write "Lärarna heter " & strFornamn & strEfternamnBPN " och " strFornamn
Response.Write strEfternamnBAN & "."
```

Resultatet blir

```
12Lärarna heter Björn Persson och Björn Andersson.
```

Om man bara vill skriva ut värdet på en variabel, t.ex. inbäddat i HTML-kod, så skriver man

```
<%
  Dim strFornamn
  strFornamn = "Björn "
%>
<!-- Annan HTML-kod -->
<p>Vi har två lärare som heter: <%=strFornamn%> i förnamn.</p>
```

Resultatet blir

```
Vi har två lärare som heter: Björn i förnamn.
```

1.3. Variabler

Variabler i ASP-kod är löst "typade" och kan endast vara av datatypen Variant. Man kan m.a.o. använda samma variabel för att t.ex. lagra både tal och strängar. Variabler deklareras, precis som i Visual Basic, m.h.a. det reserverade ordet `Dim` men utan någon datatyp. Variablerna bör deklareras så att ASP-tolken kan utföra felkontroller, t.ex. så att man inte av misstag stavat fel på en variabel.

För att tvinga ASP-tolken att kontrollera variabelnamn så använder man sig av direktivet `Option Explicit` (samma som i VB). Direktivet måste placeras i början på ASP-filen, innan öppnande HTML-tagg.

```
<% Option Explicit %>  <!--Tolken ska kontorllera att variabler deklarerats -->

<HTML>
<BODY>
<%
```

```
    Dim i           'En variabel av typen Variant
    'resten av dokumentet...
%>
```

1.3.1. Undertyper

Variabler av datatypen Variant har något som kallas undertyper, d.v.s. motsvarande det som vi i Visual Basic kallar för de ”vanliga” datatyperna. Utan konvertering av ett värde till en viss undertyp kan vi inte styra vilken undertyp som kommer hamna i en variabel vid tilldelning till variabel – VBScript kommer avgöra undertypen åt oss. Metoder för konvertering behandlas senare.

- Numeriska – Integer, Byte, Long, Single, Double och Currency.
- Sträng – String (”En sträng”)
- Datum – Date (#00-12-13#)
- Boolesk – Boolean (True eller False)
- Övriga – Empty (ej initierad), NULL (ingenting) och Object (t.ex. komponenter) och Error.

```
Dim intTal, datDatum, sngFlyttal, objEmpty, objNull
intTal = 5
datDatum = #00-12-13#
sngFlyttal = 1.5
objNULL = Null

Response.Write "<p>Talet " & intTal & " är av datatypen " & TypeName(intTal) _
   & "</p>"
Response.Write "<p>Talet " & sngFlyttal & " är av datatypen " _
   & TypeName(sngFlyttal) & "</p>"
Response.Write "<p>Summan av talen " & intTal & " och " & sngFlyttal & " är "
Response.Write (intTal + sngFlyttal) & "</p>"
Response.Write "<p>Datumet " & datDAtum & " är av datatypen " _
   & TypeName(datDatum) & "</p>"
Response.Write "<p>Den tomma variablen " & objEmpty & " är av datatypen " _
   & TypeName(objEmpty) & "</p>"
Response.Write "<p>Den tomma variablen " & objNull & " är av datatypen " _
   & TypeName(objNull) & "</p>"
```

1.4. Operatorer

1.4.1. Aritmetiska

Operator	**Förklaring**
`+, -, *, /`	de fyra räknesätten fungerar precis som vanligt
`^`	exponent (`2^2` → `4`)
`-`	negering (`-1`)
`\`	heltalsdivision – resten kastas (`5/2` → `2`)
`mod`	modulus operatorn – resten av heltalsdivision (`5 mod 2` → `1`)

```
    Dim x, y
    x = 5
    y = 2
    Response.Write x & " + " & y & " = " & (x+y) & "<br>"
    Response.Write x & " - " & y & " = " & (x-y) & "<br>"
    Response.Write x & " * " & y & " = " & (x*y) & "<br>"
```

```
Response.Write x & " / " & y & " = " & (x/y) & "<br>"
Response.Write x & " ^ " & y & " = " & (x^y) & "<br>"
Response.Write x & " \ " & y & " = " & (x\y) & "<br>"
Response.Write x & " mod " & y & " = " & (x mod y) & "<br>"
```

1.4.2. Jämförelse

Operator	Förklaring
=	lika med
<>	skilt från
<	mindre än
>	större än
<=	mindre än eller lika med
>=	större än eller lika med

```
Dim x, y
x = 5
y = 4
Response.Write x & " = " & y & " =&gt; " & (x = y) & "<br>"
Response.Write x & " <> " & y & " =&gt; " & (x <> y) & "<br>"
Response.Write x & " < " & y & " =&gt; " & (x<y) & "<br>"
Response.Write x & " > " & y & " =&gt; " & (x>y) & "<br>"
Response.Write x & " <= " & y & " =&gt; " & (x<=y) & "<br>"
Response.Write x & " >= " & y & " =&gt; " & (x>=y) & "<br>"
```

1.4.3. Logiska

Operator	Förklaring
AND	logiskt och
OR	logiskt eller
NOT	logisk negering

```
Dim x, y
x = True
y = False
Response.Write x & " AND " & y & " =&gt; " & (x AND y) & "<br>"
Response.Write x & " OR " & y & " =&gt; " & (x OR y) & "<br>"
Response.Write x & " AND NOT " & y & " =&gt; " & (x AND NOT y) & "<br>"
```

1.4.4. Sammanfoga strängar

Operator	Förklaring
& (eller +)	Sammanfogar strängar. Undvik + dådet kan misstolkas om tal.

Påsista raden i exemplet nedan kommer ett fel att genereras. För att inte visa felmeddelandet struntar vi i eventuella fel genom att skriva `On Error Resume Next` påraden ovanför.

```
Dim x, y
Dim z
x = "Hej "
y = "hå!"
z = 4
Response.Write x & " &amp; " & y & " =&gt; " & (x & y) & "<br>"
Response.Write x & " + " & y & " =&gt; " & (x + y) & "<br>"
```

```
On Error Resume Next  'Det kommer bli fel på nästa rad
Response.Write z & " + " & x & " + " & y & " =&gt; " & (z + x + y) & "<br>"
```

1.4.5. Strängmanipulation

Funktion	Förklaring
`Ucase(str)`, `LCase(str)`	Returnerar strängen `str` konverterad till stora resp. småbokstäver
`Len(str)`	Returnerar längden på strängen `str`
`Left(str, antal)`	Returnerar antal tecken från början på strängen `str`
`Right(str, antal)`	Returnerar antal tecken från slutet på strängen `str`
`Mid(str, start, antal)`	Returnerar antal tecken från positionen start i strängen `str`
`InStr(str, sökstr)`	Returnerar startpositionen av strängen `sökstr` i strängen `str`
`LTrim(str)`, `RTrim(str)`, `Trim(str)`	Returnerar strängen `str` utan mellanslag i början, slutet resp. både början och slutet.

```
Dim x, y, z, w, v
x = "Björn     "
y = "En lite längre sträng som vi ska söka i för lite skoj"
z = 7
w = 9
v = "lite"
Response.Write "UCase(" & x & ") = " & UCase(x) & "<br>"
Response.Write "LCase(" & x & ") = " & LCase(x) & "<br>"
Response.Write "Len(" & x & ") = " & Len(x) & "<br>"
Response.Write "Left(" & y & ", " & z & ") = " & Left(y,z) & "<br>"
Response.Write "Right(" & y & ", " & z & ") = " & Right(y,z) & "<br>"
Response.Write "Mid(" & y & ", " & w & ", " & z & ") = " & Mid(y,w,z) & "<br>"
Response.Write "InStr(" & y & ", " & v & ") = " & InStr(y,v) & "<br>"
Response.Write "Trim(" & x & ") = " & Trim(x) & "!<br>"
```

1.4.6. Konvertering

Nedan listas några konverteringsfunktioner som kan vara användbara

- `Abs(tal)` – returnerar talet som positivt.
- `Asc(tkn)`, `AscB(tkn)`, `AscW(tkn)` – konverterar ett tecken till dess ASCII-kod (eller ANSI).
- `Chr(tal)`, `ChrB(tal)`, `ChrW(tal)` – konverterar en ASCII-kod till tecken.
- `CBool`, `CByte`, `CCur`, `CDate`, `CDbl`, `CInt`, `CLng`, `CSng`, `CStr` – konvertering till respektive datatyp (se mer i hjälpen).
- `DateSerial(år, månad, dag)` – skapar ett datum ”objekt” (`DateSerial(2000, 11, 30)`).
- `DateValue(”sträng”)` – skapar ett datum (`DateValue(”2000-11-30”)`).
- `TimeSerial`, `TimeValue` – fungerar på motsvarande sätt som datumfunktionerna ovan.
- `Hex(tal)`, `Oct(tal)` – konverterar ett tal till dess hexadecimala resp. oktala representation.

- `Fix(tal), Int(tal)` – returnerar heltalet av ett tal (se mer i hjälpen).
- `Sgn(tal)` – returnerar –1 om tal negativt, 0 om talet är noll och 1 om talet är positivt.

```
Dim x, y, z, w, v
x = -5
y = "a"
z = 97
w = "97"
v = "34,56"
Response.Write "Abs(" & x & ") = " & Abs(x) & "<br>"
Response.Write "Asc(" & y & ") = " & Asc(y) & "<br>"
Response.Write "AscB(" & y & ") = " & AscB(y) & "<br>"
Response.Write "AscW(" & y & ") = " & AscW(y) & "<br>"
Response.Write "Chr(" & z & ") = " & Chr(z) & "<br>"
Response.Write x & " är av typen " & TypeName(x)
Response.Write " och kan konverteras med CSng(" & x & ") till "
Response.Write "typen " & TypeName(CSng(x)) & " (" & CSng(x) & ")<br>"
Response.Write "Genom att skriva DateSerial(2000, 11, 30) så skapas ett "
Response.Write "datum 'objekt' (" & TypeName(DateSerial(2000, 11, 30)) & ")<br>"
Response.Write "Ett annat sätt att skapa ett datum är med funktionen "
Response.Write "DateValue(""2000-11-30"") - " & DateValue("2000-11-30") & "<br>"
Response.Write "Hex(" & z & ") = " & Hex(z) & "<br>"
Response.Write "Oct(" & z & ") = " & Oct(z) & "<br>"
Response.Write "Int(" & v & ") = " & Int(v) & "<br>"
Response.Write "Fix(" & v & ") = " & Fix(v) & "<br>"
Response.Write "Sgn(" & x & ") = " & Sgn(x) & "<br>"
```

1.5. Funktioner för tid och datum

Förutom funktionerna DateSerial, DateValue, TimeSerial och TimeValue så har VBScript en del ytterligare funktioner för tid och datum.

- `Date` – returnerar dagens datum.
- `Year(datum)` – returnerar året för datum.
- `Month(datum)` – returnerar månaden för datum.
- `Day(datum)` – returnerar dagen för datum.
- `Weekday(datum)` – returnerar dag i veckans nummer (med start på söndagen!).
- `WeekdayName(tal)` – returnerar namnet på dagen i veckan (`WeekdayName(Weekday(Date))` – se mer i hjälpen).
- `DateAdd(intervall, antal, datum)` – adderar antal intervall till datum (`DateAdd(”d”, 7, Date)` lägger till 7 dagar till dagens datum – se mer i hjälpen).
- `DateDiff(intervall, datum1, datum2)` – returnerar antal intervall mellan datum (`DateAdd(”m”, Date, datFjol)` returnerar antal månader mellan dagens datum och datum i variabeln `datFjol` – se mer i hjälpen).
- `Time` – reurnerar klockslaget.
- `Hour(tid)` – returnerar timmen i klockslaget.
- `Minute(tid)` – returnerar minuten i klockslaget.
- `Second(tid)` – returnerar sekunden i klockslaget.
- `Now` – returnerar både dagens datum och klockslaget.

```
<!-- OBS! Detta är _inte_ det mest effektiva sätt att använda ASP!
  Man bör _inte_ hoppa ut och in i skripttaggar! -->
<P>Dagens datum är <B><%=Date%></B></P>
<P>Dagens nummer i månaden är <B><%=Day(Date)%></B></P>
<P>Dagens datum är <B><%=Weekday(Date)%></B></P>
<P>Idag är det <B><%=WeekdayName(Weekday(Date))%></B></P>
<P>Tiden just nu är <B><%=Time%></B></P>
<P>Antal minuter efter hel timme är: <B><%=Minute(Time)%></B></P>
<P>Dagens datum och tiden just nu är <B><%=Now%></B></P>
```

1.6. Konstanter

Konstanter deklareras med det reserverade ordet Const och konstanternas namn brukar skrivas med stora bokstäver. Vid deklaration måste man ange värdet på konstant genom att tilldela värdet.

```
Const MINKONSTANT = "Björn"
```

En av de konstanter som kan användas för att formatera den resulterande HTML-koden (från en ASP-sida) är `vbCrLF` som lägger in en radbrytning. Även konstanten `vbTab` kan användas för att göra indrag. Användandet av dessa två konstanter påverkar endast den resulterande kodens utseende – inte hemsidans utseende.

```
Response.Write "<H2>Rubrik på en rad i HTML-koden</H2>" & vbCrLF
Response.Write "<P>Stycke på nästa rad...<P>" & vbCrLF
Response.Write "<P>Högerklicka i webbläsaren och välj 'Visa källa' "
Response.Write "(eller motsvarande i din webbläsare)."
```

1.7. Vektorer

Vektorer är, liksom i de flesta språk, något som man brukar ha nytta av. En vektorvariabel deklareras enligt exemplet nedan. Observera att antalet positioner i vektorn blir ett mer än vad som anges vid deklarationen nedan (det är alltså ingen felskrivning med nian).

```
Dim arrVektor(9)              'Vektor med 10 positioner
arrVektor(3) = ”fyra”         'Tilldelning till 4:e positionen
Dim arrVektor2(9,9)           '2-dimensionell vektor
ArrVektor2(3, 0) = ”fyrtio”   'Tilldelning till 1:a positionen i 4:e vektorn
```

1.8. If-satser

If-satser i VBScript fungerar precis som i Visual Basic. ElseIf- och Else-delarna är, precis som i Visual Basic, inte obligatoriska.

```
Dim idag
idag = Day(Date)
Response.Write "Dagen i månaden är: " & idag & ". "
If idag < 15 Then
  Response.Write "Vi är i början på månaden."
ElseIf idag = 15 Then
  Response.Write "Vi är mitt i månaden."
Else
  Response.Write "Vi är i slutet på månaden."
End If
```

1.9. Case-satser

If-satser använder man oftast då man vill göra en jämförelse mellan två alternativ. Case -satser brukar man användas då man har fler än ett alternativ att välja på

```
Dim idag
idag = Day(Date)
Response.Write "Dagen i månaden är: " & idag & ". "
Select Case idag
  Case 1,2,3,4,5,6,7,8,9,10,11,12,13,14
    Response.Write "Vi är i början på månaden."
  Case 15
    Response.Write "Vi är mitt i månaden."
  Case 16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31
    Response.Write "Vi är i slutet på månaden."
End Select
```

1.10. Testfunktioner

`IsArray(var)` – returnerar sant om variabel är en vektor.

`IsDate(var)` – returnerar sant om variabel är av undertypen Date.

`IsEmpty(var)` – returnerar sant om variabel inte blivit satt till något värde (d.v.s. 0 eller tom sträng, “”).

`IsNull(var)` – returnerar sant om variabel har ett ogiltigt värde (d.v.s. Null).

`IsNumeric(var)` – returnerar sant om variabel (endast) innehåller ett numeriskt värde. Kan vara ett tal (t.ex. 42 eller 3.14) eller en sträng som representerar ett tal (t.ex. “3.14”).

`Is` – operator som används för att avgöra om en variabel refererar till ett visst objekt (t.ex. Nothing – se exempel nedan).

`TypeName(var)` – returnerar typen av variabel som sträng.

`VarType(var)` – returnerar ett tal mellan 0 och 17 eller 8192 (`vbArray`) som motsvarar datatypen (se hjälpen för fler konstanter för dessa värden).

```
Dim arrVektor(9)
Dim datIdag
Dim strTal, intTal, dblTal
Dim objInget

Response.Write "IsArray(arrVektor) = "
Response.Write IsArray(arrVektor) & "<BR>"

Response.Write "IsEmpty(datIdag) = "
Response.Write IsEmpty(datIdag) & "<BR>"

Response.Write "datIdag = DateSerial(2000, 12, 05)<BR>"
datIdag = DateSerial(2000, 12, 05)
Response.Write "IsDate(datIdag) = "
Response.Write IsArray(arrVektor) & "<BR>"

strTal = "3.14"
intTal = 42
dblTal = 3.14
Response.Write "IsNumeric(strTal) = "& IsNumeric(strTal)
Response.Write " (VarType(strTal) = " & TypeName(strTal) & ")<BR>"
Response.Write "IsNumeric(intTal) = " & IsNumeric(intTal)
Response.Write " (VarType(intTal) = " & TypeName(intTal) & ")<BR>"
Response.Write "IsNumeric(dblTal) = " & IsNumeric(dblTal)
Response.Write " (VarType(dblTal) = " & TypeName(dblTal) & ")<BR>"

Set objInget = Nothing
Response.Write "objInget Is Nothing = " & (objInget Is Nothing)
```

1.11. Loopar

Det finns, precis som i Visual Basic, ett antal loopar att välja på Syntaxen (eller delar av syntaxen) för looparna visas i nedanstående stycken.

1.11.1. For…Next

Loopa från ett startvärde till ett slutvärde.

```
Dim i
Response.Write "<H3>Dagens nummer är följande denna månad:</H3>"
For i = 1 To Day(Date)
  Response.Write "Dag " & i & " - " & Int(Rnd()*31) & "<BR>"
Next
```

1.11.2. For Each…Next

Loopa en gång för varje position i vektorn. I varje loop kopieras elementet på den positionen i vektor till variabeln mellan Each och In. Det går även att loopa över vektorer av typen Collection.

```
Dim vektor(4), element
vektor(0) = "ett"
vektor(1) = "två"
vektor(2) = "tre"
vektor(3) = "fyra"
vektor(4) = "fem"

Response.Write "<H3>Vektorn innehåller:</H3>"
For Each element In vektor
  Response.Write element & "<BR>"
Next
```

1.11.3. Do While…Loop

Loopa så länge villkoret `villkor` är sant. Loopen utförs aldrig om villkoret är falskt vid start.

```
Dim i
Dim max
i = 0
max = 10
Do While i < max
  Response.Write i & " "
  i = i + 1
Loop
```

1.11.4. Do…Loop While

Loopa sålänge villkoret `villkor` är sant. Loopen kommer alltid att utföras minst en gång.

```
Dim i
Dim max
i = 0
max = 10
Do
  Response.Write i & " "
  i = i + 1
Loop While i < max
```

1.11.5. Do…Loop Until

Loopa tills villkoret `villkor` är sant. Loopen kommer alltid att utföras minst en gång.

```
Dim i
Dim max
i = 0
max = 10
Do
  Response.Write i & " "
  i = i + 1
Loop Until i > max
```

1.12. Procedurer och funktioner

Den största skillnaden på procedurer och funktioner i VBScript är att det inte anges någon datatyp för parametrarna eller funktionens returvärde. Annars fungerar de precis som i Visual Basic. Nedan visas syntaxen för deklaration av procedurer och funktioner.

```
<%
  Sub MinProcedur()
    Response.Write "<P>MinProcedur har anropats...</P>"
  End Sub

  Function MinFunktion(Tal1, Tal2)
    Dim Res
    Res = Tal1 + Tal2
    MinFunktion = Res
  End Function
%>
<P>Nu ska vi anropa proceduren och funktionen</P>
<%
  MinProcedur
  Response.Write "Resultatet från MinFunktion är: " & MinFunktion(1,2)
%>
```

Här nedan visas hur man anropar procedurer och funktioner. Det reserverade ordet `Call` är inte obligatoriskt men kan göra koden tydligare, t.ex. för att ett proceduranrop utan parametrar inte ska tolkas som en variabel.

1.13. Inkludera filer

Ibland kan det vara praktiskt att samla funktioner, procedurer och annan kod, som man använder i flera filer, i en fil för att sen inkludera denna fil i dessa flera filer. För att göra det så använder man något som kallas *server side includes* (SSI). I exemplet nedan så inkluderas en fil som finns i samma mapp som filen som inkluderar.

```
<!-- #include file="inkludera.inc" -->
```

SSI fungerar i de flesta webbservrar, d.v.s. är inte en unik ASP-funktion, men olika versioner av webbservrar kan vara olika känsliga på hur man skriver kommandot. En del webbservrar kräver att det är ett mellanslag mellan "`<!--`" och "`#`" medan andra webbservrar inte bryr sig.

1.14. Ändra språk för ASP-kod

Det finns tre sätt att ändra språket för ASP-sidor:

1. Ändra för alla ASP-filer i en webbapplikation genom ändringar i IIS.
2. Ange språk för hela aktuell ASP-fil.
3. Ange språk för del av ASP-fil.

I IIS kan man ändra standardspråket för alla ASP-filer i en s.k. webbapplikation. En webbapplikation är ASP-filer, m.m. i en mapp och dess undermappar (mer om detta senare). Ändringarna görs m.h.a. administrationsprogrammet *Internet Service Manager*.

Om man bara vill ändra för endast en ASP-fil kan man använda direktivet @LANGUAGE som brukar anges längst upp i ASP-filen.

```
<%@LANGUAGE="JScript"%>
<!-- Resten av dokumentet -->
```

Det sista alternativet för att ändra språk är att använda elementet <SCRIPT> och ange språket som värde till elementets attribut LANGUAGE. Denna form lämpar sig bäst att använda om man vill inkludera skriptkod som ska exekvera på klienten. Då använder man sig även av attributet RUNAT i elementet <SCRIPT>.

```
<-- Början av dokumentet -->

<SCRIPT RUNAT="Client" LANGUAGE="JScript">
<!-- rem Kommentera bort så att äldre läsare inte visar koden
  rem Kod som ska exekvera på klienten
-->
</SCRIPT>

<-- Resten av dokumentet -->
```

Att använda elementet <SCRIPT> rekommenderas inte då det innebär att man måste tänka på följande:

- Kod i elementet <SCRIPT> tolkas sist av allt och hamnar efter </HTML>, d.v.s. ”utanför” HTML-dokumentet.
- Kod i elementet <SCRIPT> bör kommenteras bort för att äldre webbläsare, som inte förstår taggen, visar koden som vanligt text annars. (Webbläsare är gjorda för att bortse från taggar dom inte förstår.)
- Det blir mer kod att skriva jämfört med ”<%” och ”%>” och koden mer svårläst ☺.

Nedan visas ett exempel som visa varför elementet <SCRIPT> inte bör användas för att skript på serversidan.

```
<body>
<p>Rad 1: Vanlig HTML</p>
<% Response.Write ”<p> Rad 2: HTML med Response.Write</p>” %>
<SCRIPT RUNAT=”Server” LANGUAGE=”VBScript”>
<!--
  Response.Write ”<p> Rad 3: HTML med Response.Write i SCRIPT-taggen</p>”
-->
</SCRIPT>
<p>Rad 4: Vanlig HTML</p>
</body>
```

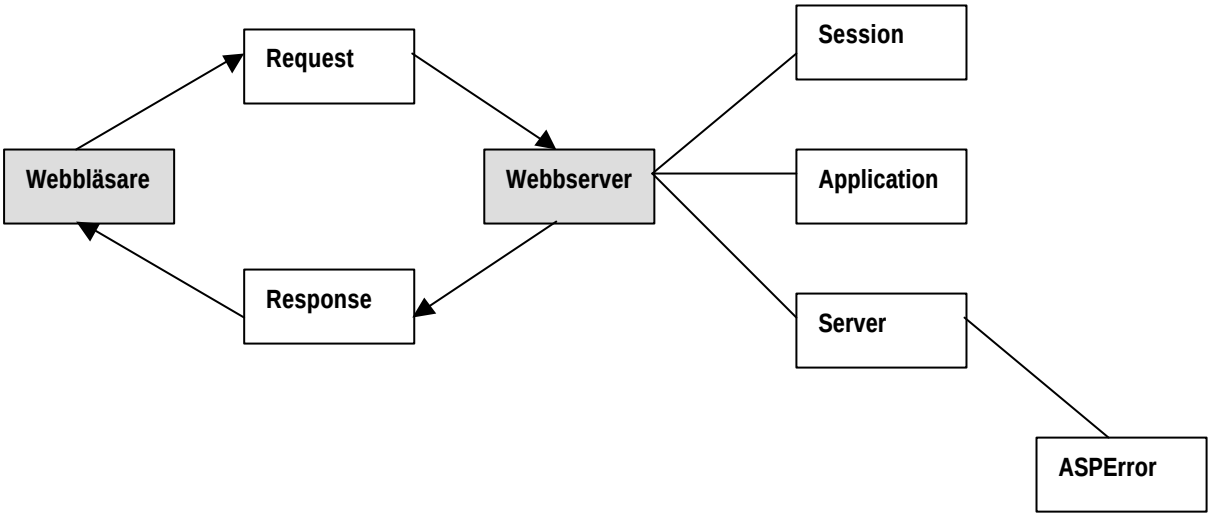
Resultatet blir

```
Rad 1: Vanlig HTML
Rad 2: HTML med Response.Write
Rad 4: Vanlig HTML
Rad 3: HTML med Response.Write i SCRIPT-taggen
```

Man bör begränsa sig till ett språk per ASP-fil för att tolkningen av ASP-koden ska ske så snabbt som möjligt.

2. Grundläggande objekt i ASP 3

ASP 3 består av 6 objekt som alltid finns tillgängligt utan att de behöver skapas: **Response**, **Request**, **Session**, **Application**, **Server** och **ASPError**. De mest frekvent använda objekten är Response och Request, som används för bl.a. formulärhantering och för att skapa de dynamiska hemsidorna som skickas tillbaka till besökarens webbläsare. Figuren nedan visar objektens förhållande till webbläsaren och webbservern. (Objektet ASPError kommer inte behandlas i detta kompendium.)



ASP innehåller även ett antal s.k. skriptobjekt för bl.a. åtkomst av webbserverns filsystem samt för att läsa och skriva till filer. Dessa objekt måste skapas för att användas.

2.1. Request

Objektet Request är det objekt som motsvarar webbläsarens begäran om att hämta ASP-dokumentet från webbservern. Innehåller i objektet består av information som ingår i denna begäran, såsom besökarens IP-adress och resultatet från ett formulär. Objektet innehåller bl.a. fyra egenskaper (vektorer av typen Collection) och är:

- `Cookies`
- `Form`
- `QueryString`
- `ServerVariables`

De två mest intressanta egenskaperna i objektet Request är Form och QueryString. I detta kapitel kommer vi endast att titta på hur de används. (För mer utförliga exempel, se *Formulärhantering* nedan.)

2.1.1. Vektorn Cookies

En *cookie* används främst för att spara information i besökarens webbläsare för senare åtkomst. När besökaren (genom webbläsaren) begär att hämta en webbsida på en webbplats så bifogar webbläsaren de *cookies* som webbplatsen sparat i besökarens webbläsare vid ett tidigare besök.[2] Dessa *cookies* blir därmed tillgängliga för ASP-sidor att använda. Ett skäl till

[2] En webbserver kan m.a.o. inte (eller bör inte kunna) läsa cookies från andra webbplatser av säkerhetsskäl.

att använda *cookies* är för att spara anpassade inställningar som besökaren har gjort på t.ex. en ASP-sidas innehåll och utseende (som man kan göra på http://www.msn.com/).

Nedan visas ett exempel på hur man läser två *cookies* (`testcookie` och `nbvisits`) från besökarens webbläsare och skriver ut dem.

```
Response.Write "Cookie innehåller: " & Request.Cookies("testcookie")
Response.Write "Du har besökt hemsidan COOKIESADD.ASP "
Response.Write Request.Cookies("nbvisits") & " gånger."
```

Hur man sparar cookies i besökarens webbläsare visas då vi går igenom objektet Response.

2.1.2. Vektorn Form

Vektorn `Form` innehåller informationen som besökaren fyllt i ett formulär och som sänts med metoden `POST` (för formulär). Formulärs innehåll som sänts med metoden `POST` skickas som en del av webbläsarens begäran om en hemsida från webbservern (mer om detta under *Formulärhantering*).

För att läsa innehållet i textrutorna `txtNamn` och `txtEpost` i ett formulär så skriver man följande

```
Dim strNamn, strEpost              'Deklarera variabler att lagra formulärinnehåll
strNamn = Request.Form(”txtNamn”)    'Hämta värdet som fyllts i textrutan txtNamn
strEpost = Request.Form(”txtEpost”)  'Hämta värdet som fyllts i textrutan txtEpost
  'Behandla strängarna på något sätt, t.ex. skriv ut eller spara till databas
```

2.1.3. Vektorn QueryString

Om formuläret skickats med metoden `GET` (för formulär) så kommer innehållet från formuläret att placeras i vektorn `QueryString` istället för vektorn `Form`. Formulärets innehåll kommer även att skickas som en del av URL:en till hemsidan istället för som en del av webbläsarens begäran om hemsida. Om formuläret innehåller två textrutor som ovan (`txtNamn` och `txtEpost`) så skulle URL:en kunna se ut enligt följande

```
http://www.spam.com/spara.asp?txtNamn=Sune&txtEpost=sune@mail.com&btnSubmit=Submit
```

Koden för att hämta innehållet i dessa två textrutor skulle kunna se ut enligt följande

```
Dim strNamn, strEpost
strNamn = Request.QueryString("txtNamn")
strEpost = Request.QueryString("txtEpost")
```

2.1.4. Vektorn ServerVariables

Egenskapen `ServerVariables` är, som nämnt ovan, en vektor av typen Collection. En vektor av typen Collection kan lagra värden inte bara med index utan även med ett nyckelvärde. Nedan visas en tabell som visar värden i vektorn och nycklar för åtkomst av värdena samt en kort beskrivning av dessa värden.

Nyckel	Exempel på värde	Beskrivning
PATH_INFO URL	/asp/exempel.asp	Sökvägen till ASP-filen på webbservern som en absolut URL.
PATH_TRANSLATED	c:\inetpub\wwwroot\ asp\exempel.asp	Sökvägen till ASP-filen i filsystemet.
LOCAL_ADDR	192.168.0.1	IP-adressen till webbservern
REMOTE_ADDR	192.168.0.2	IP-adressen till besökarens dator
REMOTE_HOST	bpn2.eki.mdh.se	Adressen till besökarens dator
REQUEST_METHOD	GET (eller POST)	På vilket sätt ett formulär skickades
HTTP_ACCEPT_LANGUAGE	sv	Vilket språk som besökaren föredrar
HTTP_REFERER	http://bpn.eki.mdh.se/ asp/default.asp	URL:en till hemsida som besökaren klickade på länken till denna hemsida
HTTP_USER_AGENT	Mozilla/4.0 (compatible; MSIE 5.01; Windows NT)	Vilken webbläsare som besökaren har

```
    Response.Write "Request.ServerVariables(""PATH_INFO"") = "
    Response.Write Request.ServerVariables("PATH_INFO")

    Response.Write "Request.ServerVariables(""PATH_TRANSLATED"") = "
    Response.Write Request.ServerVariables("PATH_TRANSLATED")

    Response.Write "Request.ServerVariables(""REMOTE_ADDR"") = "
    Response.Write Request.ServerVariables("REMOTE_ADDR")
```

2.2. Response

På liknande sätt som för objektet Request så motsvarar objektet Response det svar som skickas tillbaka till webbläsaren.

Objektet Response innehåller endast en vektor (`Cookies`) men ett antal metoder och andra egenskaper av intresse. Dessa är:

- `Buffer`
- `Expires`
- `ExpiresAbsolute`
- `.Clear()`
- `.End()`
- `.Flush()`
- `.Redirect()`
- `.Write()`
- `Cookies`

2.2.1. Egenskapen Buffer

När man skriver ut till den resulterande hemsidan så kan man placera allt i en buffert (på webbservern). På detta sätt kommer inte den resulterande hemsidan skickas till webbläsaren förrän hela ASP-sidan exekverats färdigt. För att ändra om webbservern ska buffra eller inte så ändrar man egenskapen `Buffer` till `True`. Detta måste ske **innan** öppnande HTML-tagg.

```
<% Response.Buffer = True %>
<HTML>
<HEAD>
<!—Resten av dokumentet -->
```

Som standard i IIS5 så sparas allt i bufferten medan man i tidigare versio ner av IIS var tvungen att själv ändra egenskapen `Buffer` till `True`.

2.2.2. Egenskaperna Expires och ExpiresAbsolute

Om besökares webbläsare hämtat en ASP-sida och besökaren sen återvänder till sidan så kommer webbläsaren begära sidan igen. Om man har någorlunda s tatisk information som visas i ASP-sidor så kan man bespara besökaren väntetiden (d.v.s. tiden det tar att ladda hemsidan igen) genom att låta webbläsaren buffra den resulterande hemsidan. [3] Tiden som webbläsaren ska buffra tilldelas egenskap `Expires` och anges i minuter. (Om man vill vara säker på att ASP -sidan buffras av webbläsare i hela världen så kan man ange 1442 minuter, 24 timmar * 60 minuter + 2 minuter för att vara säker.)

Vill man istället ange ett datum och/eller en tid när webbläsaren först ska hämta en ny version av ASP-sidan så kan man använda egenskapen `ExpiresAbsolute`.

```
<% Response.Expires = 1442   '24 timmar + 2 minuter  %>

<% Response.ExpiresAbsolute = Date + 1 'Dagens datum + 1 dag  %>
```

2.2.3. Metoden Clear

Om t.ex. ett visst villkor uppfylls (eller inte uppfylls) en bit ner i koden så kan man rensa bufferten genom att anropa metoden `Clear()`. Information som redan skickats till webbläsaren kommer inte att rensas! Metoden tar inga parametrar.

2.2.4. Metoden End

Genom att anropa metoden `End()` så avslutas e xekveringen av all kod i en ASP-sida och webbservern meddelar webbläsaren att hemsidan är färdig. All information som finns i bufferten kommer att skickas till webbläsaren. Metoden tar inga parametrar.

2.2.5. Metoden Flush

Om ASP-sidan t.ex. innehåller tidskrävande exekvering så kan man använda sig av metoden `Flush()` för att skicka hemsidan som mindre delar – lite åt gången. Metoden tömmer bufferten vilket leder till att webbservern skickar det som finns i bufferten till webbläsaren.

---

[3] De flesta webbläsare buffrar hemsidor (på besökarens hårddisk) som besökaren titta på för att det ska gå snabbare att ladda hemsidan nästa gång besökaren fyller i en URL till en hemsida som besökaren redan varit på. Webbläsaren raderar hemsidor i bufferten som besökaren inte varit på inom en viss tid.

Genom att skicka delar av en hemsida (t.ex. med texten ”Bearbetning pågår…”) så är det mindre chans att besökaren tror att webbservern inte är tillgänglig. Metoden tar inga parametrar.

Nedan är ett exempel på hur man kan använda `Flush()` för att visa text bokstav för bokstav.

```
<% Response.Buffer = On 'included for IIS4 only, default in IIS5 %>
<!-- Resten av dokumentet -->
<%
Response.Flush
strText = "Titta vad jag kan... Det går lite för snabbt bara..."
For intChar = 1 To Len(strText)
  For intWrite = 1 To 100000 : Next
    Response.Write Mid(strText, intChar, 1)
  Response.Flush
Next
%>
```

2.2.6. Metoden Redirect

Om man t.ex. flyttar en hemsida, men vill att tidigare besökare med bokmärken till den ”gamla” adressen ska hitta rätt, så kan man använda metoden `Redirect()`. Metoden tar som parameter en URL, i form av en sträng, som man vill att besökaren ska omdirigeras till. Anropet av denna metod måste ske innan utskrift till hemsida skett, d.v.s. innan öppnande HTLM-tagg.

```
Response.Redirect(”http://www.eki.mdh.se/”)
```

2.2.7. Metoden Write

Metoden `Write()` skriver ut till den resulterande hemsidan (och har behandlat tidigare).

2.2.8. Vektorn Cookies

I objektet Request kan man endast läsa från vektorn `Cookies` och i objektet Response kan man endast lägga till värden i vektorn (man får dock inget felmeddelande om man skriver till samma *cookie* två gånger). Eftersom vektorn är av typen Collection så använder man ett nyckelvärde (såsom en sträng) för att skriva och läsa *cookies* från vektorn. *Cookies* måste sättas innan utskrift sker till dokumentet, d.v.s. innan öppnande HTML-tagg.

Nedan visas ett exempel som testar om man inte redan satt *cookie* `testcookie`. Om denna *cookie* inte är satt så sätter man den till strängen ”`Hello world!`” och `nbvisits` till `1`. Skulle `testcookie` vara satt så ökar man bara på cookie `nbvisits` med 1.[4]

```
If Request.Cookies("testcookie") = "" Then       'Kontrollera om cookie saknas...
  Response.Cookies("testcookie") = "Hello world!" '... skriv då båda cookies
  Response.Cookies("nbvisits") = 1
  Response.Cookies("testcookie").Expires = Date + 1  'Gör cookie beständig
  Response.Cookies("nbvisits").Expires = Date + 1    'Gör cookie beständig
  Response.Write "Cookies set!"
Else                                              '... annars öka bara räknare
  Response.Cookies("nbvisits") = CInt(Request.Cookies("nbvisits")) + 1
  Response.Cookies("nbvisits").Expires = Date + 1
  Response.Write "Cookies already set! Only number of visits increased."
```

[4] Att öka på ett värde kan generera fel om webbläsaren inte har ett värde i den *cookie* man försöker öka på!

```
End If
```

För att radera en *cookie* från användarens webbläsare så kan man tilldela `Date - 1` till egenskapen Expires för den *cookie* som man vill radera. (Se mer i hjälpen för Visual InterDev om *cookies* med flera värden.)

2.3. Session

För varje besökare till webbplatsen skapas ett Session-objekt och objektet är endast giltigt för den besökaren. Objektet existerar så länge besökaren befinner sig på webbplatsen, vilket inte alltid kan vara lätt att avgöra då förbindelse mellan webbserver och -läsare endast etableras då en hemsida överförs. Objektet kan bl.a. användas för att lagra information så att informationen är tillgänglig för alla hemsidor som besökaren besöker. Session-objektet innehåller även två händelser, OnStart och OnEnd, förutom ett antal egenskaper och metoder, bl.a. följande:

- `SessionID`
- `TimeOut`
- `Abandon()`
- `Contents`
- `OnStart`
- `OnEnd`

2.3.1. Egenskapen SessionID

Alla sessioner får ett unikt identifikationsnummer kallat `SessionID` för att skilja de olika besökarna (eller snarare deras webbläsare) från varandra. Egenskapen kan endast läsas då den genereras av webbservern.

2.3.2. Egenskapen TimeOut

När en session börjar är ganska lätt att förstå – när besökaren begär första hemsidan på webbservern. Men eftersom förbindelsen mellan webbläsare och webbserver är tillfällig, d.v.s. endast existerar när webbläsaren begär en hemsida och webbservern skickar hemsidan, så är det svårare att avgöra när en session är slut. Ett (mycket tydligt) sätt att avsluta en session är att anropa metoden `Abandon()` eller (mindre tydligt) låta ”tiden för inaktivitet” ta slut (*time-out)*. Denna tid är tillgänglig genom egenskapen `TimeOut` samt kan både läsas och ändras från ASP-kod (eller ändras med konfigurationsprogrammet för IIS). Standardvärdet brukar vara 20 minuter.

```
Session.TimeOut = 30   'Andra till 30 minuter
```

2.3.3. Metoden Abandon

Metoden `Abandon()` används för att avsluta en session och kan vara användbar om man låter besökaren logga på webbplatsen. När besökaren loggat på så kontrollerar man på de sidor som kräver inloggning – privata sidor, om besökaren är inloggad och när besökaren loggar ut så tillås inte besökaren (eller snarare webbläsaren) åtkomst till de privata sidorna. Metoden tar inga parametrar.

2.3.4. Vektorn Contents

I vektorn Contents (av typen Collection) kan man lagra värden som kan läsas och ändras från alla ASP-sidor inom samma session. Man kan se vektorn som en lokal *cookie* som bara existerar så länge sessionen existerar. För att spara och hämta ett värde till/från vektorn kan man skriva enligt följande

```
Session.Contents("nyckel") = "värde"   'Spara ett värde till vektorn Contents

strVar = Session.Contents("nyckel")    'Hämta ett värde från vektorn Contents
```

För att radera ett eller alla värden i vektorn finns metoderna `Remove()` och `RemoveAll()`. Den första metoden tar som parameter emot nyckeln för det värde man vill ta bort. Den senare metoden tar bort alla värden från vektorn.

```
Session.Contents.Remove("nyckel")   'Ta bort ett värde från vektorn Contents

Session.Contents.RemoveAll          'Ta bort alla värden från vektorn Contents
```

2.3.5. Händelserna OnStart och OnEnd

(Se *Händelserna OnStart och OnEnd* under Application-objektet nedan.)

2.4. Application

I motsatts till objektet Session så gäller alla egenskaper för objektet Application alla besökare av webbplatsen (eller snarare webbapplikationen – se senare). Application-objektet skapas då webbservern startas och existerar tills webbservern stängs av eller startas om. Liksom Session-objektet så kan Application-objektet lagra information som är tillgänglig från alla hemsidor på webbplatsen. Även Application-objektet innehåller händelserna `OnStart` och `OnEnd` samt några fler metoder och egenskaper.

- `Lock()`
- `Unlock()`
- `Contents`
- `OnStart`
- `OnEnd`

2.4.1. Metoderna Lock och Unlock

Eftersom objektet Application är tillgängligt för alla besökares ASP-sidor så kan värden som lagras i vektorn `Contents` uppdateras av flera ASP-sidor samtidigt. För att förhindra samtidig access för skrivning så anropar man metoden `Lock()` som förhindrar uppdatering av Application-objekts egenskaper av andra ASP-sidor. När man är färdig med uppdateringen anropar man metoden `Unlock()` för att tillåta andra att uppdatera Application-objektet. Om man inte anropar `Unlock()` så kommer webbservern att anropa metoden bl.a. när ASP-sidan exekverat färdigt.

```
Application.Lock
Application.Contents("nbvisits") = Application.Contents("nbvisits") + 1
Application.Unlock
```

2.4.2. Vektorn Contents

Om man vill lagra information som är tillgängligt för alla ASP-sidor kan man lagra informationen i Application-objektets vektor `Contents`. Den fungerar på samma sätt som Session-objektets vektor `Contents`.

```
Response.Write "Antal besökare är: " & Application.Contents("nbvisitors")
```

2.4.3. Händelserna OnStart och OnEnd

När webbservern startas och Application-objektet skapas så inträffar händelsen `OnStart` för Application-objektet vilket leder till att metoden `Application_OnStart` anropas (om den finns – se mer under *Filen GLOBAL.ASA*). När webbservern stängs av så sker händelsen `OnEnd` och på motsvarande sätt så anropas metoden `Application_OnEnd` (om den finns). Man kan använda dessa två händelser (metoder) för att skapa objekt (instanser av t.ex. en komponent) respektive förstöra objekten.

Observera att komponenter som kan tänkas lagras i Application-objektet kan användas av flera samtidiga användare och bör använda trådmodellen *Free* eller *Both* för att serverns prestanda ska påverkas så lite som möjligt!

Webbapplikationer

En webbapplikation omfattar alla ASP-filer i en mapp och dess undermappar (och som inte ingår i en annan webbapplikation). När IIS installeras så skapas en rotwebbapplikation som omfattar alla ASP-filer på webbservern. Om man vill kan man skapa en ny webbapplikation genom att skapa en ny mapp (i t.ex. rotmappen för webbservern) och tala om för IIS att man vill att denna mapp (och undermappar) ska vara en ny webbapplikation. Detta görs lättast i Visual InterDev genom att skapa ett nytt projekt samt låta Visual InterDev skapa mappen och ordna inställningarna för den nya webbapplikationen.

Filen GLOBAL.ASA

En del av webbapplikationerna är filen GLOBAL.ASA. Filen innehåller inställningar som man alltid vill ha i Application- eller Session-objekten när webbservern startas respektive ny besökare tillkommer. I GLOBAL.ASA kan man också implementera metoder för händelserna `OnStart` och `OnEnd` för objekten Application och Session.

Nedan visas ett exempel på hur (en mindre bra) räknare av antal samtidiga personer på webbplatsen skulle kunna implementeras med metoderna `Session_OnStart` och `Session_OnEnd`.

```
<SCRIPT LANGUAGE="VBScript" RUNAT="Server">
Sub Session_OnStart
  Application.Lock
  Application.Contents("nbvisitors") = Application.Contents("nbvisitors") + 1
  Application.Unlock
End Sub

Sub Session_OnEnd
  Application.Lock
  Application.Contents("nbvisitors") = Application.Contents("nbvisitors") – 1
  Application.Unlock
End Sub

Sub Application_OnStart
  Application.Contents("start") = Date   'Initiera en global variabel
   'Resten av implementationen av metoden
```

```
End Sub

Sub Application_OnEnd
   'Implementationen av metoden
End Sub
</SCRIPT>
```

2.5. Server

I objektet Server har placerats de egenskaper och metoder som motsvarar webbserverns inställningar samt de som inte passar i de andra objekten (vilket kan få objektet Server att se ut att sakna struktur). Den mest användbara metoden i Server-objektet är `CreateObject()` och används för att skapa instanser av komponenter. Och objektet innehåller endast en egenskap (`ScriptTimeOut`) men ett antal metoder som är:

- `ScriptTimeOut`
- `.CreateObject()`
- `.Execute()`
- `.HTMLEncode()`
- `.MapPath()`
- `.Transfer()`
- `.URLEncode()`

2.5.1. Egenskapen ScriptTimeOut

Som standard är IIS konfigurerad att låta ASP-sidor exekvera i 90 sekunder innan den avbryter exekveringen. Om man har en ASP-sida som behöver längre tid på sig att exekvera så kan man ändra det med egenskapen `ScriptTimeOut`.

```
Server.ScriptTimeout = 180
```

2.5.2. Metoden CreateObject

Denna metod används för att skapa instanser av komponenter och som parameter tar metoden en sträng som motsvarar komponentens ProgID. Metoden `CreateObject()` i objektet Server bör användas framför den (i ASP) inbyggda funktionen `CreateObject()`. D.v.s. i exemplet nedan så bör man undvika den översta raden till fördel för den undre.

```
CreateObject("KompTest.Slump")          'Undvik den inbyggda funktionen

Server.CreateObject("KompTest.Slump")   'Använd metoden i objektet Server
```

2.5.3. Metoderna Execute och Transfer

Dessa två metoder har en liknande funktion, vilken påminner om metoden `Redirect()` i objektet Response (som är tänkt att eventuellt utgå i en senare version av ASP). Men de två metoderna skiljer sig från varandra på en punkt – `Execute()` exekverar en ASP-fil men lämnar sen tillbaka "kontrollen" till ASP-filen där anropet gjordes medan `Transfer()` tar över "kontrollen" och lämnar inte tillbaka den. I exemplet nedan så skrivs tre rader ut från ASP-filen samt mellan utskrifterna anropas först `Execute()` och sen `Transfer()`. Den tredje utskriften i ASP-filen kommer inte att ske.

```
    Response.Write "<P>Första raden i EXECUTE_TRANSFER.ASP</P>"
    Server.Execute("doc1.asp")
    Response.Write "<P>Andra raden i EXECUTE_TRANSFER.ASP</P>"
    Server.Transfer("doc2.asp")
    Response.Write "<P>Tredje raden EXECUTE_TRANSFER.ASP</P>"
```

Dessa två metoder är nya i ASP version 3 (d.v.s. finns inte i IIS4 eller tidigare).

2.5.4. Metoderna HTMLEncode och URLEncode

Dessa två metoder kan bespara en utvecklare av webbapplikationer (med IIS) en hel del – utvecklaren slipper koda t.ex. svenska tecken för HTML och sen sitta och läsa denna något svårlästa kod (”`Björn`” är lättare kod att läsa än ”`Bj&ouml;rn`”). `HTMLEncode()` konverterar ”vanlig” text till HTML-kod, d.v.s. kodar eventuella specialtecken (såsom < och >) till HTML-kodens motsvarighet (`<` respektive `>`).

Om man skapar länkar med ASP-kod så kan man använda sig av `URLEncode()` för att skapa en giltig URL. Om ett formulärs resultat skickas (med metoden GET) till en ASP-sida och t.ex. innehåller svenska tecken (i namnet Björn) så skulle den kunna se ut enligt följande

```
http://www.spam.com/search.asp?Fornamn=Bj%F6rn&Efternamn=Persson
```

Genam att skicka en sträng som parameter till `URLEncode()` så skulle vi i ASP-kod skapa en URL med ”specialtecken” utan att känna till alla koder.

2.5.5. Metoden MapPath

Denna metod används för att översätta en (absolut eller relativ) URL till en sökväg i filsystemet. Exemplet nedan visa hur man skapar en hemsida med länkar till alla filer i mappen och femte raden visar hur man hämtar sökväg till aktuell mapp med `MapPath()`. Komponenten `Scripting.FileSystemObject` behandlas senare.

```
Dim objFSO, objFolder, objFile, strParentFolder
  'Skapa en instans av FileSystemObject
Set objFSO = Server.CreateObject("Scripting.FileSystemObject")
  'Hämta sökväg till aktuell mapp som en sträng
strParentFolder = Server.MapPath("./")
  'Hämta en referens till aktuell mapp
Set objFolder = objFSO.GetFolder(strParentFolder)
  'Loopa över alla filer i mappen
For Each objFile In objFolder.Files
    'Visa inte denna fil (DEFAULT.ASP)
  If objFile.Name <> "default.asp" Then
      'Bygg en lista med länkar till filer i denna mapp
    Response.Write "<a href=""" & objFile.Name & """>" & objFile.Name & "</a><br>"
  End If
Next
```

2.6. ASP skriptobjekt

IIS innehåller även ett antal s.k. skriptobjekt som är tillgängliga för ASP-skript. Dessa objekt, i motsats till de inbyggda objekten ovan, måste skapas innan de kan användas. Dessa objekt är

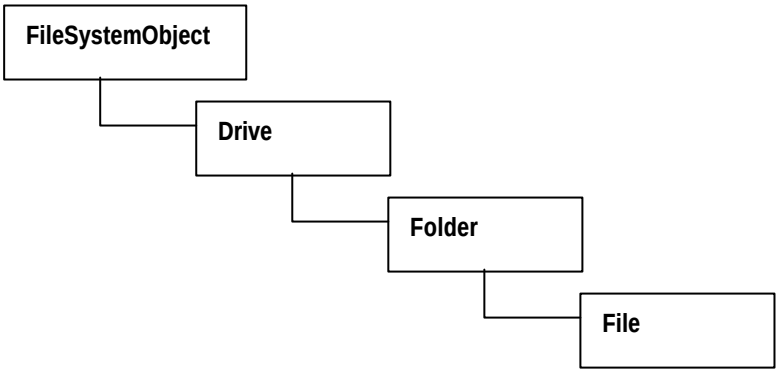
- `Dictionary`
- `FileSystemObject`

- `RegExp`
- `TextStream`

För information om Dictionary- och RegExp-objekt – se hjälpen.

2.6.1. FileSystemObject

För att kunna läsa filsystemet från ASP-kod såkan ett objekt av typen FileSystemObject skapas. Egentligen är det inte bara ett objekt utan en hierarki av objekt, men man måste skapa en instans av FileSystemObject för att fåtillgång till de andra objekten.



FileSystemObject har bara en egenskap, `Drives`, men desto fler metoder, bl.a. följande

- `GetDrive("enhet")` – returnerar ett Drive-objekt för enheten, t.ex. "`C:`".
- `GetFolder("sökväg")` – returnerar ett Folder-objekt för mappen, t.ex. "`C:\Program`".
- `GetFile("sökväg")` – returnerar ett File-objekt för filen, t.ex. "`C:\Program\Microsoft Office\Office\Excel.exe`".
- `CreateTextFile("filnamn.txt", skriv_över)` – metoden skapar en fil. Om andra parametern (som inte är obligatorisk) är `False` såskrivs inte eventuell redan existerande fil över. I motsats till vad som står i hjälpen såverkar det som om den är `True` som standard, d.v.s. filen skrivs över om parametern utelämnas.
- samt metoder för att skapa, flytta och radera filer och mappar (se mer i hjälpen).

När man använder Get-metoderna i FileSystemObject såreturneras alltsåett objekt av denna typ. Alla dessa objekt (Drive, Folder och File) innehåller både egenskaper, med vanliga typer och vektorer, samt metoder. Nedan beskrivs kortfattat några av deras egenskaper och metoder.

Drive-objektet

Drive-objektet innehåller endast egenskaper, dock inga vektorer.

- `DriveType` – ett tal som visar typen av enhet (0 – okänd, 1 – flyttbar/diskett, 2 – lokal enhet, 3 – nätverks enhet, 4 – CD-ROM eller 5 – RAM-disk).
- `FreeSpace` – antal lediga byte påenheten.
- `IsReady` – om enheten är klar att användas eller inte (`True` eller `False`). Lämpligt att testa innan man läser från diskett eller CD-ROM.
- `Path` – sökvägen (eller snarare enhetsbokstav), t.ex. "`C:`".

- `TotalSize` – storleken på enheten i bytes.
- `VolumeName` – namnet på enheten, t.ex. ”`MinDisk`”.

Folder objektet

Detta objekt innehåller egenskaper som är av både vanliga typer och vektorer.

- `DateCreated` – datum som mapp skapades.
- `DateLastModified` – datum som mapp sist ändrades.
- `Name` – namn på mappen.
- `ParentFolder` – sökvägen till mappen som aktuell mapp finns i.
- `Path` – sökvägen till aktuell mapp.

File objektet

- `DateCreated` – datum som filen skapades.
- `DateLastModified` – datum som filen sist ändrades.
- `Name` – namn på filen (långa filnamnet).
- `ParentFolder` – mapp som filen finns i.
- `Path` – sökväg till filen, inklusive filnamn.
- `Size` – storleken på filen i bytes.

Exempel – Get-metoder och egenskaper i objekten

Exemplet visar på hur Get-metoderna i FileSystemObject fungerar samt exempel på egenskaper i det tre övriga objekttyperna (Drive, Folder och File).

```
Response.Write "<H2>Drives</H2>"
Dim objFSO, objDrive, objFolder, objFile
Set objFSO = Server.CreateObject("Scripting.FileSystemObject")
Set objDrive = objFSO.GetDrive("C:")

Response.Write "<P><B>Drive: </B>" & objDrive.Path & "</P>"
Response.Write "<TABLE>"
Response.Write "<TR><TD><B>"
Response.Write "Volume's name is:</B></TD><TD>" & objDrive.VolumeName
Response.Write "</TD></TR><TR><TD><B>"
Response.Write "Total size (in bytes):</B></TD><TD>" & objDrive.TotalSize
Response.Write "</TD></TR><TR><TD><B>"
Response.Write "Free space (in bytes):</B></TD><TD>" & objDrive.FreeSpace
Response.Write "</TD></TR>"
Response.Write "</TABLE>"

'*********************************************************************
Response.Write "<H2>Folders</H2>"
Set objFolder = objFSO.GetFolder("C:\Program")
Response.Write "<P><B>Folder: </B>" & objFolder.Path & "</P>"

Response.Write "<TABLE>"
Response.Write "<TR><TD><B>"
Response.Write "Folder's name is:</B></TD><TD>" & objFolder.Name
Response.Write "</TD></TR><TR><TD><B>"
Response.Write "Date created:</B></TD><TD>" & objFolder.DateCreated
Response.Write " (?)</TD></TR><TR><TD><B>"
Response.Write "Last modified:</B></TD><TD>" & objFolder.DateLastModified
Response.Write "</TD></TR><TR><TD><B>"
Response.Write "Parent folder:</B></TD><TD>" & objFolder.ParentFolder
```

```
Response.Write "</TD></TR><TR><TD><B>"
Response.Write "Size of files in folder <BR>and subfolders " _
   & "(in bytes):</B></TD><TD>" & objFolder.Size
Response.Write "</TD></TR>"
Response.Write "</TABLE>"

'*****************************************************************
Response.Write "<H2>Files</H2>"
Set objFile = objFSO.GetFile("C:\Program\Microsoft Office\Office\Excel.exe")
Response.Write "<P><B>File: </B>" & objFile.Path & "</P>"

Response.Write "<TABLE>"
Response.Write "<TR><TD><B>"
Response.Write "File's name is:</B></TD><TD>" & objFile.Name
Response.Write "</TD></TR><TR><TD><B>"
Response.Write "Date created:</B></TD><TD>" & objFile.DateCreated
Response.Write "</TD></TR><TR><TD><B>"
Response.Write "Last modified:</B></TD><TD>" & objFile.DateLastModified
Response.Write "</TD></TR><TR><TD><B>"
Response.Write "Parent folder:</B></TD><TD>" & objFile.ParentFolder
Response.Write "</TD></TR><TR><TD><B>"
Response.Write "File type:</B></TD><TD>" & objFile.Type
Response.Write "</TD></TR><TR><TD><B>"
Response.Write "Size of file (in bytes):</B></TD><TD>" & objFile.Size
Response.Write "</TD></TR>"
Response.Write "</TABLE>"
```

Exempel – vektorer i objekten

Detta exempel visar påhur de olika vektorerna i FileSystemObject och dess ”underobjekt” (Drive, Folder och File) kan användas för att lista objekten i vektorerna.

```
Dim objFSO, objDrive
Set objFSO = Server.CreateObject("Scripting.FileSystemObject")

For Each objDrive in objFSO.Drives
  Response.Write objDrive.DriveLetter & " - "
  If objDrive.DriveType = 1 Then
    Response.Write "Diskette"
  ElseIf objDrive.DriveType = 4 Then
    Response.Write "CD-ROM"
  Else
    Response.Write objDrive.VolumeName & " - " _
       & Int(objDrive.TotalSize/(1024*1024)) & " MB"
  End If
  Response.Write "<BR>" & vbCrLf
Next
Response.Write "</P>"
'*******************************************************
Response.Write "<H2>Folders in root of drive C</H2>"
Response.Write "<P>"

Dim objFolder, objFold
Set objDrive = objFSO.GetDrive("C:")
Set objFolder = objDrive.RootFolder

For Each objFold in objFolder.SubFolders
  Response.Write objFold.Name & "<BR>" & vbCrLf
Next
Response.Write "</P>"
'*******************************************************
Response.Write "<H2>Files in root of drive C</H2>"
Response.Write "<TABLE>"
Response.Write "<TR><TH>File Name</TH><TH>Size (Bytes)</TH></TR>"

Dim objFile
Set objDrive = objFSO.GetDrive("C:")
Set objFolder = objDrive.RootFolder

For Each objFile in objFolder.Files
  Response.Write "<TR><TD>"
  Response.Write objFile.Name & "</TD><TD ALIGN=""right"">"
```

```
    Response.Write objFile.Size & "</TD></TR>" & vbCrLf
  Next
  Response.Write "</TABLE>"
```

Exempel – läsa och skriva till textfil

```
  <P>Denna sida skapar en fil (C:\Student\TestFile.txt) för att skriva, skriver
  strängen "This is a test" och stänger filen.</P>
  <%
      'Använder två TextStream objekt för att visa att filen verkligen skapades :-)
    Dim objFSO, objTS1, objTS2, strText
    Set objFSO = Server.CreateObject("Scripting.FileSystemObject")
    Set objTS1 = objFSO.CreateTextFile("c:\student\testfile.txt", True)
    objTS1.WriteLine("This is a test.")
    objTS1.Close
  %>
  <P>Filen öppnas igen för att läsa, strängen läses in och skrivs till denna
  sida.</P>
  <%
    Set objTS2 = objFSO.OpenTextFile("c:\student\testfile.txt", 1)
    strText = objTS2.ReadLine
    objTS2.Close
    Response.Write "<P>" & strText & "</P>"
  %>
```

(För exempel på att lista filer i en mapp och skapa en hemsida med länkar till filerna – se *Metoden MapPath* för objektet Server ovan.)

2.6.2. TextStream

Objekt av typen TextStream kan inte de heller fungera utan att en instans av FileSystemObject har skapats. Följande metod kan användas för att öppna en fil med hjälp av ett File-objekt.

- `.OpenAsTextStream(iomode)` – öppnar filen som finns i File-objektet. Parametern iomode används för att avgöra på vilket sätt filen ska öppnas: 1 – öppna för läsning, 2 – öppna för att skriva eller 8 – öppna för att lägga till (*append*). (Enligt hjälpen ska det finnas konstanter för dessa värden – `ForReading`, `ForWriting` och `ForAppending`.)

Egenskaper och metoder i objektet TextStream förklaras kort nedan och exemplet nedan visar hur några av dessa egenskaper och metoder kan användas.

- `AtEndOfLine` – returnerar `True` om aktuell position i filen är i slutet på raden.
- `AtEndOfStream` – returnerar `True` om aktuell position i filen är i slutet på filen.
- `Close()` – stäng filen.
- `ReadAll()` – läs alla rader i filen.
- `ReadLine()` – läs en rad i filen.
- `Write(string)` – skriv en sträng till filen.
- `WriteLine(string)` – skriv en sträng till filen och flytta markören till nästa rad.
- `WriteBlankLines(antal)` – lägg till tomma rader i filen och flytta markören.

```
    Dim objFSO, objFile1, objFile2, objTS, strText
    Set objFSO = Server.CreateObject("Scripting.FileSystemObject")
      'Om fil existerar...
    If objFSO.FileExists("C:\Student\textstream.txt") Then
        '... hämta en referens till filen
      Set objFile1 = objFSO.GetFile("c:\student\textstream.txt")
    Else
        '... eller skapa filen och hämta en referens till filen
      objFSO.CreateTextFile("c:\student\textstream.txt")
      Set objFile1 = objFSO.GetFile("c:\student\textstream.txt")
```

```
End If
  'Öppna filen som ett TextStream-objekt
Set objTS = objFile1.OpenAsTextStream(8) '8 = ForAppending
objTS.WriteLine "This was added " & Now & "."
objTS.Close
Set objTS = Nothing
  'Öppna filen igen för att läsa
Set objFile2 = objFSO.GetFile("c:\student\textstream.txt")
Set objTS = objFile2.OpenAsTextStream(1)  '1 = ForReading
Response.Write "The following was read from file: <BR><B>"

While Not objTS.AtEndOfStream  'Loopa tills slutet på filen
  strText = objTS.ReadLine
  Response.Write strText & "<BR>"
Wend
objTS.Close

Response.Write "</B></P>"
```

(Denna sida har avsiktligt lämnats tom.)

3. Formulärhantering

Formulär är det främsta sättet för en utvecklaren av en webbapplikation att kommunicera med besökaren. Det finns bl.a. två sätt att hantera formulär och den mottagande ASP-filen:

- formuläret skapas i en ”vanlig” HTML-fil och anropar en ASP-fil för att hantera resultatet från formuläret.
- formuläret skapas i en ASP-fil som anropar sig själv för att hantera resultatet från formuläret.

3.1. Formulär i HTML-fil som anropar en ASP-fil

Koden för denna typ av formulärhantering blir relativ enkel. Men om besökaren inte fyller i alla fält så måste besökaren backa ett steg om han missat något. Och i vissa webbläsare så försvinner informationen man fyllt i formulär. Exemplet nedan består av två filer – en ren HTML-fil och en ASP-fil.

Formuläret som ren HTML-kod

```
<HTML>
<HEAD>
  <TITLE>Sign up for our news letter</TITLE>
</HEAD>
<BODY>
<H1>Sign up for our news letter</H1>
<P>Enter your name and e-mail address</P>
<FORM METHOD="GET" ACTION="formsimple_save.asp">
  <P>Name: <INPUT TYPE="text" NAME="txtName" SIZE="20"></P>
  <P>E-mail: <INPUT TYPE="text" NAME="txtEmail" SIZE="20"></P>
  <P><INPUT TYPE="submit" VALUE="Submit" NAME="btnSubmit">
    <INPUT TYPE="reset" VALUE="Reset" NAME="btnReset"></P>
</FORM>
</BODY>
</HTML>
```

ASP-filen som hanterar resultatet från HTML-filen ovan.

```
<HTML>
<HEAD>
  <TITLE>You have entered the following information</TITLE>
</HEAD>
<BODY>
<H1>You have entered the following information</H1>
<P>
<%
  Dim strName, strEmail
  strName = Request.QueryString("txtName")
  strEmail = Request.QueryString("txtEmail")
  Response.Write "Your name is <B>" & strName & "</B><BR>"
  Response.Write "Your e-mail address is <B>" & strEmail & "</B>"
%>
</P>
</BODY>
</HTML>
```

3.2. Formulär i ASP-fil som anropar sig själv

När ASP-filen begärs första gången så bifogar inte webbläsaren någon information från formuläret. Genom att kontrollera att begäran inte innehåller någon information från formulär så kan avgöra om formuläret ska visas eller om man ska hantera resultatet från formuläret.

Genom att använda denna form av formulärhantering så kan man lite lättare kontrollera att alla obligatoriska fält har fyllts i innan man behandlar informationen. Och om besökaren har missat ett obligatoriskt fält så kan man uppmana besökaren att fylla i fältet innan besökaren går vidare.

Den generella strukturen för denna typ av ASP-fil är

```
<%
  If {info från formulär} Then
     'Behandla informationen
  Else
     'Visa formulär
%>
<!-- T.ex. HTML-kod för formuläret -->
<%
  End If
%>
```

Nedan visas ett exempel med två textrutor där man i formuläret anropar ASP-filen som formuläret finns i. Observera att om man använder värden från ett formulär fler a gången så är det mer effektivt att lagra värdet i en variabel och sen använda variabeln. Det är kostsamt att indexera i ett objekt av typen Collection (som både QueryString och Form är).

```
<HTML>
<HEAD>
  <TITLE>Sign up for our news letter</TITLE>
</HEAD>
<BODY>
<%
    'Use variables to store form fields - indexing in Collections are expensive!
    ' These variables are used 3 times.
  Dim strName, strEmail, strWarning
  strName = Request.QueryString("txtName")
  strEmail = Request.QueryString("txtEmail")
    'If form not filled in properly
  If ( strName = "") Or ( strEmail = "") Then
    If Request.QueryString.Count <> 0 Then 'Only entered in one field
      strWarning = "<B>You must fill in both fields!</B>"
    End If
%>
<H1>Sign up for our news letter</H1>
<P>Enter your name and e-mail address. <%=strWarning%></P>
<FORM METHOD="GET" ACTION="formsimple.asp">
  <P>Name: <INPUT TYPE="text" NAME="txtName" VALUE= "<%=strName%>" SIZE="20"></P>
  <P>E-mail: <INPUT TYPE="text" NAME="txtEmail" VALUE= "<%=strEmail%>"
    SIZE="20"></P>
  <P><INPUT TYPE="submit" VALUE="Submit" NAME="btnSubmit">
    <input TYPE="reset" VALUE="Reset" NAME="btnReset"></P>
</FORM>
<%
  Else
    Response.Write "<H1>You have entered the following information</H1>"
    Response.Write "Your name is <B>" & strName & "</B><BR>"
    Response.Write "Your e-mail address is <B>" & strEmail & "</B>"
  End If
%>
</BODY>
</HTML>
```

4. Använda komponenter i ASP

Att använda sig av komponenter i ASP skiljer sig inte så mycket från att använda dem i Visual Basic. De största skillnaderna är att

- variabler behöver inte deklareras av en viss klass (gränssnitt).
- man bör skapa objekten med `Server.CreateObject(ProgID)`.
- objekten existerar endast så länge som ASP-sidan exekverar.

Generellt ser kod för att skapa och använda komponenter i ASP ut enligt följande

```
<%
  Dim objKomp
  Set objKomp = Server.CreateObject(”Server.Komp”)
  ‘Anropa metoder
  Set objKomp = Nothing
%>
```

För att skapa och anropa komponenten `CompTestCpp.Random1` så skulle koden kunna se ut enligt följande.

```
Dim objRand, i
Set objRand = Server.CreateObject("CompTestCpp.Random1")
objRand.MaxValue = 10
objRand.Randomize
For i = 1 To 10
  Response.Write objRand.Value & "<br>"
Next
Set objRand = Nothing
```

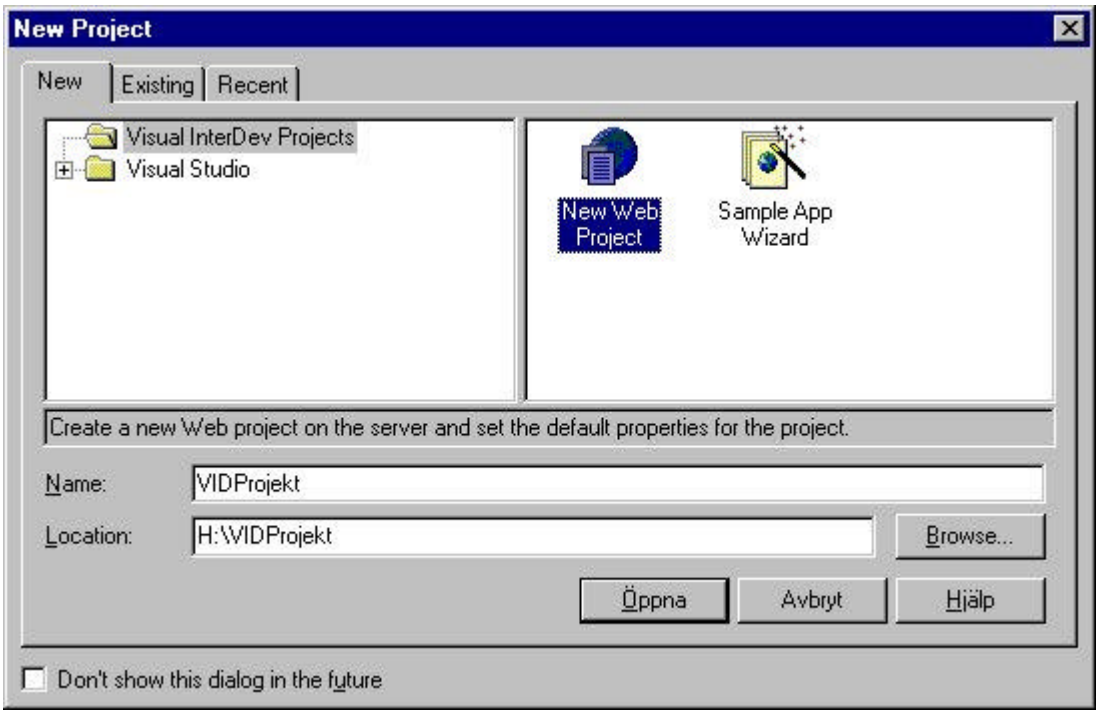
5. Använda Visual InterDev

Microsofts Visual InterDev (VID) är ett verktyg för att hantera kompletta ”webblösningar”, d.v.s. hela eller delar av webbplatser. Med hjälp av verktyget kan man hantera flera samtidiga utvecklare som redigerar hemsidor (ASP-sidor) utan att utvecklarna ska kunna redigera samma hemsida.

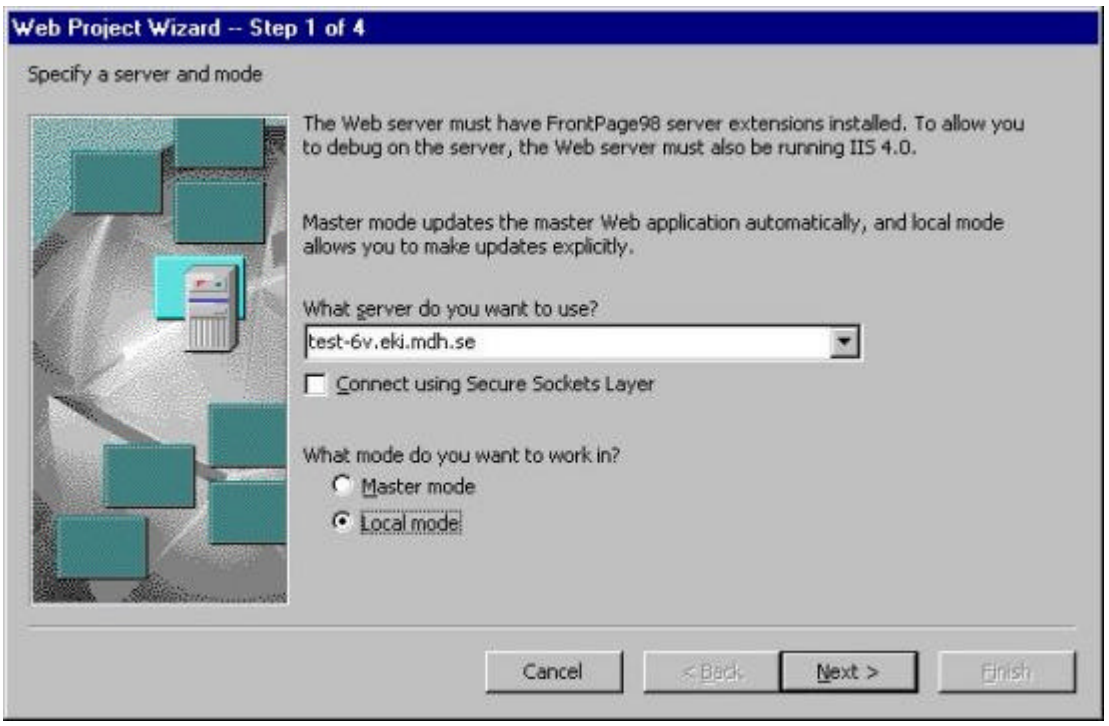
För att kunna utnyttja VID bäst bör man skapa ett (lokalt) projekt.

5.1. Skapa ett nytt projekt

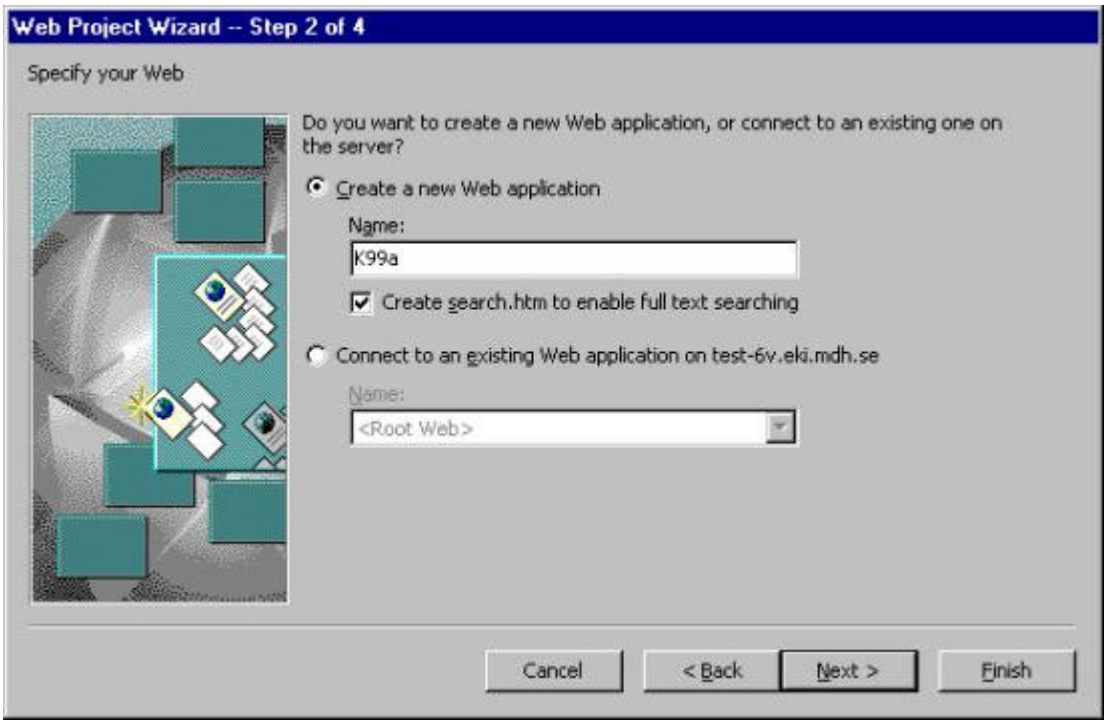
1. Starta Visual InterDev. Vanligen visas dialogrutan för att skapa ett nytt projekt eller öppna ett existerande. Om inte kan du skapa ett nytt projekt genom att välja **New Project…** från File-menyn.
2. Fyll i ett namn på projektet, välj mapp att spara projektet i och klicka på **Öppna** (se bild nedan).



3. Fyll i adressen till webbservern du vill att hemsidorna (ASP-sidorna) ska finnas. Markera även alternativet **Local mode** så att du redigerar hemsidorna lokalt **innan** du kopierar filerna till webbservern (d.v.s. du kommer inte redigera hemsidorna direkt på webbservern vilket inte är att rekommendera). Klicka på **Next>**.



4. VID kontaktar nu webbservern och ber dig eventuellt att logga på webbservern. Om det inte finns en existerande webbapplikation på webbservern bör du skapa en ny. Markera **Create a new Web application** och fyll i namnet på applikationen (vilket kommer motsvara namnet på mappen som hemsidorna kommer att placeras i, d.v.s. använd inte svenska tecken). Klicka på **Next>**.



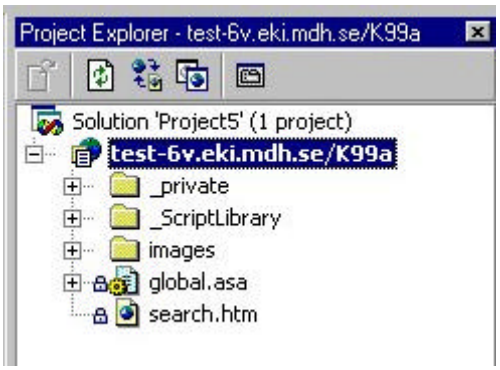
5. I nästa dialogruta kan man skapa en layout som ska gälla för alla hemsidor i webbapplikationen och i nästa dialogruta igen kan man välja ett ”tema” (*theme*) för

hemsidornas design. Vi är inte intresserade av dessa såvi klickar på **Next>** i första dialogrutan och **Finish** i andra dialogrutan.

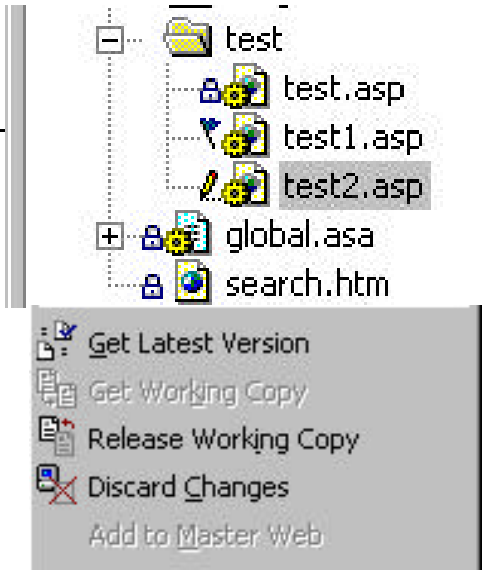
6. Visual InterDev genererar nu en massa filer, både i den lokala mappen med projektet och påservern.

5.2. Lägga till nya hemsidor

För att lägga till en hemsida klickar man påprojektet i **Project Explorer** (d.v.s. adressen till webbservern följt av ett snedstreck, ”/”, och namnet på webbapplikationen – se bild till höger). Från menyn som visas väljer man **Add…** och sen typen av fil man vill ha (t.ex. **HTML Page…** eller **Active Server Page…**). I dialogrutan behöver man endast fylla i namnet påfilen. Om man inte vill att hemsidan ska placeras i roten påwebbapplikationen kan man skapa en mapp (genom att välja **New folder…** från menyn som visades) samt sen högerklicka påmappen och välja **Add…** där istället.



Nya filer som inte kopierats till servern markeras med en blåflag framför filens namn (se bild till höger – test1.asp är en ny fil).



5.3. Kopiera den nya hemsidan till webbserver

Innan man kopierar en fil till webbservern bör man spara och stänga filen. Detta för att undvika att man redigerar en hemsida som ”är påservern”, d.v.s. att man kopierat upp ”originalfilen” till servern. För att kopiera filen till webbservern såhögerklickar man påfilen och väljer Add to Master Web (se bild till höger – i bilden har alternativet inaktiverats dåden redan kopierats till servern).

När ”originalfilen” finns påwebbservern såvisas ett gråt hänglås framför filen (se bild snett ovan – test.asp är påservern).

5.4. Redigera hemsidor

Innan man börjar redigera en fil, där ”original” filen finns påservern, bör man kontrollera att man inte har filen öppen lokalt. För att hämta ner en lokal kopia att redigera från servern så dubbelklickar man påfilen och klickar påGet Working Copy (kryssa gärna för kryssru tan nere till vänster i dialogrutan som visas). Man kan även högerklicka påfilen och välja Get Working Copy (se bild ovan – alternativet är inaktiverat dåfilen finns påservern).

När man redigerar en fil som hämtats från webbservern såvisas en gul penna framför filen (se bild ovan – test2.asp har hämtats en lokal kopia för att redigera).