Program :

WEBMASTERS

1.cjelina.....Razmatranje ASP tehnologije .....5 sati
2.cjelina.....VBScript na serverskoj strani.....20 sati
3.cjelina.....ASP objekti i komponente.....40 sati
4.cjelina.....ACCESS , SQL .....15 sati
5.cjelina.....ASP i baze podataka.....30 sati
6.cjelina.....Sigurnost podataka i ostalo.....10 sati

1.cjelina Razmatranje ASP tehnologije

ASP tehnologija radi na Microsoftovim serverima. ASP je ustvari skupina tehnologija tj, u njega se mogu dodavati razne komponente. Nalazi se na serverskoj strani. Da bi koristili Active server pages na našem serveru moramo imati Microsoftov web server. Trenutno postoje samo dva - Internet Information Server (IIS) 3.0 ili noviji, i Personal Web Server, koji je u stvari samo manja verzija IIS-a, a oba rade samo na Windowsima. Ako naš server koristi UNIX, za korištenje Active server pages moramo koristiti posebne programe za obradu ASP stranica, koje proizvode neke druge firme, neovisno od Microsofta, npr. Chilisoft.
Ako imamo IIS 3 ili noviji, da bi kreirali ASP stranicu, sve što trebamo napraviti je da stranici koja sadrži ASP skripte dodamo ekstenziju .asp. Server će tada sam znati da na takvoj stranici treba prvo izvršiti skripte prije nego se pošalje klijentu. Ako ne pronađe nikakve skripte, stranicu će poslati nepromijenjenu. Iako bi bilo jednostavno svim stranicama na web site-u promijeniti ekstenziju u .asp, stranice koje ne sadrže ASP skripte moraju ostati sa ekstenzijama .htm ili .html jer server svaku .asp stranicu temeljito pretražuje u potrazi za skriptama, što oduzima procesorsko vrijeme.

Jedna od najvažnijih odlika ASP stranica je njihovo jednostavno povezivanje sa bazama podataka (MS SQL, Access, Oracle, Informix, ili bilo koja baza podataka koja podržava ODBC standard) i dinamičko ubacivanje podataka iz baze u HTML stranice. To otvara razne napredne mogućnosti za upotrebu ASP-a, kao što su npr. Internet trgovine, site-ovi koji se prilagođavaju pojedinom korisniku, sustavi za unos i izmjenu podataka preko Interneta itd. ASP je u neku ruku jedinstven skriptni jezik, jer nam dopušta da slobodno biramo sintaksu u kojoj želimo programirati. Dva najpopularnija jezika u kojima se pišu ASP skripte su *VBScript* i *Jscript*. Parseri za ta dva jezika su ugrađeni u IIS. Korištenjem dodatnih scripting engine-a drugih firmi, možemo ASP skripte pisati i u PerlScript-u, REXX-u ili Python-u. ASP stranice se sastoje iz teksta, HTML tagova i ASP naredbi. HTML tagovi se, kao što znamo, nalaze u zagradama oblika < … >, običan tekst je izvan njih, a ASP naredbe stavljamo u zagrade oblika <% … %>. Te zagrade govore serveru kako njihov sadržaj ne ide korisniku već ga treba prvo izvršiti, a na njihovo mjesto uvrstiti izlazni rezultat skripte, što god on bio (izlazni rezultat može biti HTML, tekst, style sheet, javascript ili bilo što što standardni browser razumije).

ASP skripte sadrže naredbe VBScript-a, njegove varijable, funkcije i procedure, kao i neke stvari svojstvene samo ASP-u, kao što su funkcije za rad s datumima, pretprocesorske direktive, `include` naredbe ili ActiveX i ugrađeni objekti.

Kada klijent zatraži neku URL adresu, on šalje ***request*** . Request se sastoji od :

- Informacija o URL tj. IP adresi zadanog servera
- Informaciji o browseru klijenta tj. Tipu browsera
- Informaciji klijenta tj. o lokaciji i IP adresi

Request se provodi preko glavnog TCP/IP protokola. Da bi se klijent mogao sporazumjeti sa serverom, on mora poslati ispravno napisan request. Nakon toga server tako dugo vrti petlju, dok ne ispuni request, i čeka za novi request.

Građa URL adrese :

Npr: http://www.yahoo.com/shoping/show.asp?ID=234&ID2=235

http:// - protokol
www - host name
yahoo - enterprise domain name
com - topic level domain name (domena)
shoping - virtualni direktorij
show.asp - datoteka
? - razdvaja parametre od datoteke
ID=234&ID2=235 - tokini (parametri)

Što server radi nakon requesta ?

1. svaku adresu će pretvoriti u fizičku adresu svog servera npr: c:\ine\co
2. ako ne pronađe datoteku izbacuje 404Error
3. pregleda da li je dozvoljen pristup tim podacima

Kako server procesira traženu datoteku ?

- datoteke pronalazi po extenzijama u registriju (Microsoft)
- UNIX i ostali se ne oslanjaju na registri, nego na listu datoteka MIME type formata(html, htm, asp, php....)
- Za gif, jpg, wav, mid....je određena neka akcija
- Za datoteke tima exe, com, zip...nudi DOWNLOAD

Response koji server šalje klijentu sastoji se od glave i tijela (head i body)

- Body – sadržaj stranice
- Head – cookie, dužina vremena prepoznavanja, redirekcija, MIME type lista

Server i klijent moraju imati iste MIME type liste i plug-inova
Klijentov browser pročita response, utvrđuje datoteku, te počinje generirati html dokument.
HTML tagovi opisuju kako prikazati sadržaj, dok XML tagovi prikazuju koji je zadržaj.

SERVERI:

I.S.A.P.I. – Internet Server Aplication Programming Interface

1 Filteri- zaštita datoteka(Firewall) ---prije requesta
2.Aplikacije – ASP engine – poslije requesta

IIS – web server

- u sebi sadrži i filtere i aplikaci

2. cjelina VBScript

- scripta se piše unutar <head> tagova

```
<head>

<script languge="VBScript">
<!- -
        kod
        //- ->
</script>
  </head>
```

VBScript se sastoji od:

Varijable, Nizovi i Objekti

Varijable:

A=10 -> a je 10

Nizovi :

DIM B(4)

B(4) je ime niza i rezervirano je 5 mjesta u nizu: B(0),B(1),B(2),B(3),B(4).

Objekti:

Set Rs=CreateObject("ADODB.Recordset")

Set – metoda za stvaranje objekata
Rs – ime objekta

Primjeri varijabli i nizova:

```
<html>
<head>      <script language="vbscript">
   <!--

        sub button1_OnClick()
          dim c(3)
          c(1)=44
          c(2)=33
          c(3)=333
                      text1.value=c(1)
                      MsgBox "Izvršeno je " & c(3)+c(1) &" puta sa probom od
"  & c(3) &" pokušaja !"

                      end sub

       //-->
       </script>

       </head>
       <body>
        <center><input type=text name=text1> <br>

<input type=button name=button1 value="Klikni me" >   </center>

 </body></html>
```

PROCEDURE:

Procedure počinju sa **sub** , a završavaju sa **end sub**

Npr:

```
sub window_onload()
Msgbox "Dobro došli ! "
    End sub
```

FUNKCIJE:

Funkcije počinju sa **function**, a završavaju sa **end function**

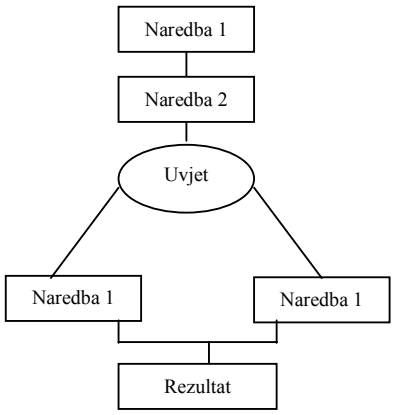
Npr:

```
<html>
<head>

<script language="vbscript">
   <!--
         function capitalize(Rijec)    'ime funkcije sa ulaznim
parametrima(rijec)

capitalize=UCase(Left(Rijec,1))&LCase(Right(Rijec,Len(Rijec)-1))
'vrijednost funkcije
         end function

         sub button1_OnClick()
             text2.value=capitalize(text1.value)
          end sub
```

```
//-->
</script>

</head>
<body bgcolor=black>
 <center>
  <br><br><br><br><br><br><br><br><br><br><br>
<table border=30 bordercolor=mangenta>
<tr>
<td>
 <p><input type=text name=text1> <p>
 <center><input type=button name=button1 value="GO !" ></center>
  <p>
 <input type=text name=text2>
 </td></tr></center>
</table>
</body>
</html>
```

Control flow

Elementi grananja



If...Then...Else

Blok primjer :

If <uvjet> **Then**
seqence1
else
sequence2
end if

ili **if** <uvjet> **Then** naredba --→ nema end if

ili **if** <uvjet1> **Then**

(sequenca1)
elseif <uvjet2> **Then**
(sequenca2)
.....
else
(sequencaN)
end if

Primjer 1:

```
<html>
<head>
<script language="vbscript">
   <!--
          sub t4_onFocus()
                op=t2.value
                if op="+" Then call zbroji
                if op="-" Then call oduzmi
                if op="*" Then call pomnozi
                if op="/" Then call podijeli
                end sub
            sub zbroji
              t4.value=CInt(t1.value)+CInt(t3.value)
              end sub

                 sub oduzmi
              t4.value=CInt(t1.value)-CInt(t3.value)
              end sub

                 sub pomnozi
              t4.value=CInt(t1.value)*CInt(t3.value)
              end sub

                    sub podijeli
              t4.value=CInt(t1.value)/CInt(t3.value)
              end sub

      //-->
      </script>

 </head>
      <body bgcolor=black>
 <center> <font color=red> <br><br><br><br><br><br><br><br><br><br> <b>
 A  
<input type=text name="t1" size=10>   
 <input type=text name="t2" size=10 >&nbspB 
<input type=text name="t3" size=10>
  = 
<input type=text name="t4"></font></center>
      </body></html>
```

Primjer 2:

```
<html>
<head>

<script language="vbscript">
   <!--
```

```
          sub racun_onclick()


            if t1.value=""  then
                    msgbox "Niste unijeli sve ocjene !",vbExclamation
                    exit sub
            end if

            if t2.value=""  then
                msgbox "Niste unijeli sve ocjene !",vbExclamation
                exit sub
            end if

           if t3.value=""  then
               msgbox "Niste unijeli sve ocjene !",vbExclamation
               exit sub
           end if

           if t4.value=""  then
               msgbox "Niste unijeli sve ocjene !",vbExclamation
               exit sub
           end if

           if t5.value="" then
               msgbox "Niste unijeli sve ocjene !",vbExclamation
               exit sub
           end if


          if t1.value="1" or t2.value="1" or t2.value="1" or t3.value="1" or
t4.value="1" or t5.value="1" Then
                 t8.value="Pad"
          else
                  pro
                  x = t7.value
                  if  (x>=1.5) and (x<2.5) then
                      t8.value="Dovoljan"
                  elseif (x>=2.5) and (x<3.5) then
                     t8.value="Dobar"

                  elseif (x>=3.5) and (x<4.5) then
                      t8.value="Vrlo dobar"

                  elseif (x>4.5) then
                        t8.value="Odličan"
                  end if
        end if
 end sub

                sub pro()

                        a=CInt(t1.value)
                        b=CInt(t2.value   )
                        c=CInt(t3.value  )
                        d=CInt(t4.value )
                        e=CInt(t5.value)
                        suma=(a+b+c+d+e)/5
                        t7.value=suma
```

```
                end sub
      //-->
      </script>

 </head>
      <body bgcolor="#658945" onload="t1.Focus"> <center>
                     <b>
 <table border=1>
<tr><td> <b> Dreamweaver</td>
<td><input type=text name=t1  size=7><br></td></tr>
<tr><td> <b>Flash</td><td><input type=text name=t2  size=7><br> </td></tr>
 <tr><td> <b>JavaScript</td><td><input type=text name=t3  size=7> </td></tr>
<br>
 <tr><td> <b>Photoshop</td><td><input type=text name=t4  size=7></td></tr>
<br><tr><td> <b>HTML</td><td><input type=text name=t5  size=7></td></tr>  <br>
    </table><p>
<table>
<tr><td ><input type=button name=racun Value=Izračunaj></td></tr>
</table>     <p>
 <table border=1>
 <tr><td> <b>Prosjek</td><td> <input type=text name=t7  size=7> </td></tr>
 <tr><td> <b>Uspjeh</td><td><input type=text name=t8  size=13>  </td></tr>
      </center>
      </body></html>
```

Select Case

Služi za ispitivanje neke varijable pomoću ključnih riječi
-> blok za ispitivanje uvjeta

Select Case <varijabla>
Case <value1> -ako je varijabla 1 pokreće naredbu1
(sequenca1)
Case <value1,value2...>
(sequenca2)

.....
Case else
(sequencaN)
End Select

Načini ispitivanja text boxova :

- **if ime_textboxa.value="" Then**
- **Len (ime_textboxa.value)=0 Then**
- **ime_textboxa.value=vbNullString Then**

Ulazni upit :

Varijabla=**InputBox**("Unesite ime","Unos imena"**)**

Primjer:

```
Sub window_onload
    a=Inputbox("Unesite Ime", Unos imena")
```

```
            Msgbox  "Dobro došli gospodine/gospođo : "  & a
        End sub
```

Kako upisati neku skriptu unutar html-a bez text boxa ?

```
html>
<head>

<script language="vbscript">
   <!--
         sub window_onload

         prvi.innerhtml="<b>Dobro došli"
          drugi.outerhtml="<input type=text>"


          end sub

       //-->
       </script>
      </head>
      <body bgcolor="#658945"> <center>

            <p id="prvi"> </p>
             <p id="drugi"> </p>



      </center>
      </body></html>
----------------------------------
```

PETLJE

For...Next

For varijabla=poč. vrijednost **To** završna vrijednost

(sequenca)
Next -kraj petlje

Primjer 1 :

```
 <html>
<head>

<script language="vbscript">
   <!--
            sub window_onload
```

```
            for i=1 to 10
              document.write i &"<br>"
            next
              end sub
      //→
      </script>

     </head>
     <body bgcolor="#658945"> <center>

<input type="button" value="Proba" name="gumb">

     </center>
     </body></html>
```

Primjer 2:

```
<html>
<head>

<script language="vbscript">
   <!--
         sub window_onload()

         for a=1 to 12
              k=inputbox("Unesite dohodak za " & a & " mjesec:
","Unos",3500)

                   document.write "<b>" & "Placa u " & a & ".mjesecu je bila
: " & k & "Kn" & "<br>"
                    prosjek = prosjek+CInt(k)
                    next
                   zbroj =prosjek/12

                   document.write "<hr>"
                   document.write "Prosjecna godisnja vrijednost place: " &
FormatCurrency(CInt(zbroj),2)

            end sub
      //-->
      </script>

 </head>
     <body bgcolor="#658945"> <center>
     </center>
     </body></html>
```

While

While <uvjet>
(sequenca)
Wend

Primjer:

```
<html>
<head>

<script language="vbscript">
   <!--

                  sub window_onload()
                  gr=cdbl(inputbox(" Unesi :"))
            while suma<gr
                a=inputbox(" Unesi :")
                   suma=suma+cint(a)
            wend
              document.write "Prešli ste granicu !"
                           end sub
                  //-->
      </script>

  </head>
      <body bgcolor="#658945"> <center>
      </center>
      </body></html>
```

Do..Loop

```
Do
  (sequenca)
Loop Until <uvjet>
```

For each...Next

```
For each a in b
   (sequenca)
Next
```

a = neodređena varijabla koja prolazi kroz niz b i postaje određena
b = ime niza

Primjer :

```
     for each a in b
        document.write a & "<br>"
     next
```

Podtipovi varijabli

Osnovni fleksibilan tip varijabli se zove *VARIANT*.

Podtipovi :

Integer	**Cint**	
Long	**CLng**	
Currency	**Ccur**	
Single	**CSng**	
Duble	**CDbl**	
Date	**CDate**	Date, time
Boolean	**Cbool**	True, false
String	**CStr**	
Object	**/**	
Error	**/**	
Byte	**Cbyte**	

Kako ispitati varijable ?

Naredba za takve slučajeve : **varType(**varijabla koju ispitujemo**)**

Npr:

```
If varType(var.koj.isp.)=8 Then
```

Svaka varijabla koju ispitujemo ima svoju znakovnu vrijednost, tako, ako je 8, onda je string, a ako je 7, onda je datum itd.

The **VarType** function returns the following values:

Constant	Value	Description
vbEmpty	0	**Empty** (uninitialized)
vbNull	1	**Null** (no valid data)
vbInteger	2	Integer
vbLong	3	Long integer
vbSingle	4	Single-precision floating-point number
vbDouble	5	Double-precision floating-point number
vbCurrency	6	Currency
vbDate	7	Date
vbString	8	String
vbObject	9	Automation object
vbError	10	Error

vbBoolean	11	Boolean
vbVariant	12	Variant (used only with arrays of Variants)
vbDataObject	13	A data-access object
vbByte	17	Byte
vbArray	8192	Array

STRINGOVI

Stringovi su textualni tip podataka (niz znakova)

Vrste :

Lenght	**Len(var)**	Broj znakova (dužina riječi)
Lower Case	**Lcase(var)**	Ispisuje riječ malim slovima
Uper Case	**Ucase(var)**	Ispisuje riječ malim slovima
Left	**Left(var)**	Pretražuje riječ od lijeve strane
Right	**Right(var)**	-||- od desne strane
Middle> 1.parametar-string s kojim radimo, 2. parametar-pozicija odakle uzimamo, 3.parametar – broj znakova koliko uzimamo	**Mid(var)**	-||- od sredine
Replace (varijabla,"što","u što", pozicija od kojeg mjesta, koliko promjena(ako je –1, onda mijenja sve riječi), vrste traženja 0- binarno 1- tekstualno 2- baza podataka	**Replace()**	Jedan dio stringa zamjeni s drugim
StrReverse	**StrReverse(var)**	Okreće riječi naopako
String	**String(koliko ponavljanja, ascii kod)**	Više puta ispisuje slovo koje smo zadali ascii kodom
Asc	**ASC()**	Traženje ascii kodova: A->65
CHR	**CHR()**	Obrnuto od ascii 65->A
SPACE	**Space()**	Broj razmaka u msgboxu
Trim	**TRIM()**	Izbacuje razmake sa svih strana
Ltrim	**Ltrim()**	Izbacuje razmake sa lijeve strane
Rtrim	**Rtrim()**	Izbacuje razmake sa desne strane
Lower Bound	**Lbound(niz)**	Vraća najniži index u nizu
Uper Bound	**Ubound(niz)**	Vraća najviši index u nizu

Za ostalo pogledati VBScript dokumentaciju.

Primjer 1 :

```
<html>
<head>

<script language="vbscript">
   <!--
  sub window_onload()

     a="Bok ljudovi ! "

        document.write  "Len-broj slova = " & len(a) & "<br>"
        document.write "LCase-mala slova = " & lcase(a) & "<br>"
        document.write  "UCase-velika slova = " & ucase(a) & "<br>"

        c = left(a,4)
        d = right(a,9)
        document.write "Lijevo = " & c & "<br>"
        document.write "Desno = " & d & "<br>"
        document.write "Sredina rijeci = " & mid(a,3,5) & "<br>"
      document.write " Promjena rijeci = " & replace(a
,"ljudovi","slonovi",1,-1,1)  & "<br>"
        document.write " Promjena taga italic u bold = " &
replace("<i>Italic</i>","i>","b>") & "<br>"
        document.write " Okretanje rijeci naopako = " & StrReverse(a) &
"<br>"

        document.write " Ponavljanje vise puta = " & String(3,13) & "<br>"
        document.write " ASCII kod od 'A' = " & asc("A") & "<br>"
        document.write " 65 u ASCII kodu je = " & chr(65) & "<br>"
        document.write " Brisanje razmaka          
bok ljudovi             !  u  = " & trim(a) & "<br>"


         msgbox " Razmaci = " & space(20) & "Kraj"

         end sub

    //-->
      </script>

  </head>
     <body bgcolor="#658945"> <center>
     </center>
     </body></html>
```

Primjer 2:

```
<html>
<head>

<script language="vbscript">
   <!--
            sub window_onload()

            a = "Nenad;Mario;Bojan;Mladen;Tomislav"
            imena = split(a,";")

            for i = LBound(imena) to UBound(imena)
```

```
            document.write  imena(i) & "<br>"

          next
      document.write "<hr width=10% align=left>"

      for each ime in imena
       document.write ime & "<br>"
       next
            end sub

      //-->
      </script>

     </head>
     <body>
       </body></html>
```

Pretraživanje stringova

InStr(pozicija,gdje,što, koja komparacija(0-binarno, 1-textualno, 2- baza podataka**)**

InStrRev(gdje,što,pozicija(-1),koja komparacija**)** **-** isto kao InStr, samo pretražuje Odozada

Primjer:

```
<html>
<head>

<script language="vbscript">
   <!--

            sub window_onload()


   a= " Dobro dosli ljudovi !"

document.write " Kljucne rijeci = " & a  & "<br>"

 document.write " Promjena taga italic u bold = " &
replace("<i>Italic</i>","i>","b>") & "<br>"

  document.write " Promjena rijeci = " & replace(a ,"ljudovi","slonovi",1,-
1,1)  & "<br>"

   document.write " Trazimo na kojem mjestu pocinje rijec dosli = " &
InStr(1,a,"dosli",1)  & "<br>"

 document.write " Trazimo na kojem mjestu pocinje rijec dosli odozada = " &
InStrRev(a,"dosli",-1,1)  & "<br>"

  end sub
      //-->
      </script>
     </head>
```

```
<body bgcolor="#658945"> <center>
</center>
</body></html>
```

Nizovi stringa

Split- podijeli

Split(a,";")

a – varijabla,tj.ime niza koji razdvajamo

-razdvaja varijablu koja sadrži neko nabrajanje razdvojeno sa **;** u niz

Primjer:

```
<html>
<head>

<script language="vbscript">
   <!--
            sub window_onload()

            a = "Nenad;Mario;Bojan;Mladen;Tomislav"
            imena = split(a,";")

            for i = LBound(imena) to UBound(imena)

             document.write  imena(i) & "<br>"

            next
       document.write "<hr width=10% align=left>"

       for each ime in imena
        document.write ime & "<br>"
        next
            end sub

       //-->
       </script>

      </head>
      <body>
         </body></html>
```

Primjer 2:

```
<HTML>
<TITLE>Current Date/Time 2</TITLE>
<HEAD>

<script language=vbscript>
<!--

       sub b1_onclick()


      baza="Drazen;Davor;Nenad;Darko;Tomislav;Mario;Pero;Jura;Stevo;Branko;
Branimir;Kreso;Bojan;Mladen"

             unos=t1.value
```

```
            imena = split(baza,";")


              for each ime in imena

                if ucase(ime)=ucase(unos) then
                     found = true
                       exit for
                  else
                     found = false

          end if
            next

              if found then
               t2.value = "Osoba postoji !"
                 else
                 t2.value = "Osoba ne postoji !"
                 end if

                 end sub
//-->
      </script>

</head>
<body bgcolor=black text=white onload="t1.focus">
<br><br><br><br><br><br>
<center>
<table width=50% height=20% bgcolor=blue border=2 bordercolorlight="#c0c0c0"
bordercolordark="#505050">
<tr><td align=center><b>Unesite ime :</td>
<td align=center>
<input      name="t1" type="text" value=""></td><td> 
<input type="button" name=b1 value="Provjeri !">
</td></tr><tr><td></td><td align=center><b>Potvrda !<br>
<input name="t2" type="text" value="" >
</td></tr></table>

</center>
</body>
</html>
```

Funkcije **Array,Join,Filter**

Join

Spajanje slova(riječi), tj redove u rečenicu, tj niz u rečenicu

Var=Join(var1,",")
Var1- određena varijabla,tj niz

Array

- funkcija za stvaranje niza

Primjer array i join :

```
<html>
<head>

<script language="vbscript">
   <!--
   sub window_onload()

            a=array("Nenad","Pero","Jura","Mloaden")

          for each x in a
            document.write x & "<br>"
              next

              b=join(a,", ")

 document.write "<hr width=20% align=left>" & b

              end sub
                  //-->
      </script>

</head>
      <body bgcolor="#658945">
      </body></html>
```

Filter

- iz niza izdvaja samo neke elemente, i od toga odvojenoga niza stvara novi niz
- var=Filter(niz,"što",true,komparacija)

Primjer 1:

```
<html>
<head>

<script language="vbscript">
   <!--
   sub window_onload()

            a=array("Proanima d.o.o","Gramip d.o.o","Pik Vrbovec
d.o.o","Agritec d.o.o","Pliva d.d")

              x=inputbox("unesi koju vrstu firme treba")

b=filter(a,x,true,1)

              for each c in b

              isp = isp & c & "<br>"

              next

              prv.innerhtml = isp

              end sub
                 //-->
      </script>

 </head>
      <body bgcolor="#658945">

        <p id=prv></p>

      </body></html>
```

VARIANTI

Typename – javit će ime stringa
IsArray – da li je varijabla niz
IsNumeric – da li je varijabla broj
IsDate – da li je varijabla datum
IsEmpty – da li je polje ili neka varijabla prazna
IsNull – posebno stanje kada je polje neupotrebljivo
-različito od empty- tj. Polje nije upotrebljivo

Primjer :

```
<html>
<head>

<script language="vbscript">
   <!--
   sub window_onload()


   a=12.654546546546546546546546546546465465
   document.write vartype(a) & "<br>"
   document.write typename(a) & "<br>" & "<hr width=20% align=left>"
```

```
   b=#19/9/2001#

document.write vartype(b) & "<br>"
   document.write typename(b) & "<br>" & "<hr width=20% align=left>"

   c=array("1","2","3")

   if isarray(C) then

      document.write "IsArray--Je, to je niz" & "<br>" & "<hr width=20%
align=left>"
else

    document.write "Ne, to nije niz" & "<br>" & "<hr width=20% align=left
>"
   end if

    if isdate(b) then

      document.write "<body bgcolor=black text=green>" &"IsDate--Je, to je
datum" & "<br>" & "<hr width=20% align=left >"
else

    document.write "Ne, to nije datum" & "<br>" & "<hr width=20% align=left
> "
   end if

      if isnumeric(a) then

      document.write "IsNumeric--Je, to je broj" & "<br>" & "<hr width=20%
align=left>"
else

    document.write "Ne, to nije broj" & "<br>" & "<hr width=20%
align=left>"
   end if

   if isempty(k) then

   document.write "IsEmpty--Datum nije upisan" & "<br>" & "<hr width=20%
align=left>"

    end if


      l=null

      if isnull(l) then
      document.write "IsNull--Je, null je" & "<br>" & "<hr width=20%
align=left>"
               end if

               end sub
            //-->
      </script>

</head>
      <body bgcolor="#658945">
        <center>
        <p id=prv></p>
      </body></html>
```

Dinamički niz

Dinamički niz je niz u kojem u tijeku izvođenja programa mijenjamo broj elemenata.

DIM(6) – fiksni niz od 6 elemenata
DIM() - fleksibilan niz sa mogućom promjenom

ReDim Preserve var(broj elemenata u varijabli**)**
ReDim Preserve a(Ubaund(a)+1) ->trenutni broj elemenata kojem dodajeno još jedan

Primjer:

```
<html>
<head>

<script language="vbscript">
   <!--
   sub window_onload()


   dim a()
   redim preserve a(5)
   ReDim Preserve a(UBound(a)+1)

   document.write Ubound(a) & "<br>"

              end sub
            //-->
      </script>

      </head>
      <body bgcolor="#658945">
         <center>
         <p id=prv></p>
      </body></html>
```

3. cjelina ASP

-active server pages
za rad sa ASP-om služi asp.dll

COM – common object model

ASP sadrži pet ugrađenih objekata. To su :
Response, *Request*, *Application*, *Session* i *Server*.

Svaki od njih ima svoje odlike koje svrstavamo u četiri grupe: *Methods*, *Properties*, *Events* i *Collections*. Methods su ugrađene funkcije koje pozivamo kao `ImeObjekta.Method(parametri)`. Properties su varijable posebne za svaki objekt. Njima pristupamo na sličan način: `ImeObjekta.Property = neka_vrijednost`. Events su događaji koji pokreću određene procedure. Njih sadrže samo objekti Application i Session, a te procedure se pišu u posebnoj datoteci `global.asa` o kojoj će biti riječi u posebnom odjeljku. Collections su, kao što im i samo ime kaže, kolekcije varijabli kojima određeni objekt može pristupati. Tim varijablama se pristupa ovako: `ImeObjekta.Collection("Varijabla")`.

Postoje 7 vrsta objekata:

1. Response - object s kojim server nešto vraća klijentu
2. Request - object s kojim klijent šalje upit serveru
3. Server - uz njega upravljamo serverom – pregled IP adresa, browser**..**
4. Aplication - služi za upravljanje aplikacijama od connest do disconnect
5. Session - služi za spremanje podatka koji su došli sa klijenta
6. ASPError - govori o nastalim greškama
7. ObjectContext -služi za upravljanje transakcijama I kreiranje objekata kroz Microsoft Transaction server

ASP datoteka je textualna datoteka sa svim svojstvima html dokumenta, ali različite ekstenzije i ima mogučnosti dometanja serverskih skripti.

Da bismo na stranici definirali da će glavni skriptni jezik biti vbscript moramo prije <html> upisati ***<%@language=''VBScript'' %>***
Serverski kod ASP-a :

<%
kod
%>

Primjeri:

```
<html>
<head>
	<title>Untitled</title>

	<%
	a =  " Dobar dan ! " & date & " " & time

%>

</head>

<body>

<%=a%>      može se upisati  <% response.write a %>

 ili  <% response.write " Dobar dan ! " & date & time %>


</body>
</html>
```

Kako prikazati nešto sa servera?

```
<%
 response.write "Srever time =" &time &"<br>"

response.write "Server date =" &date  &"<br>" &"<hr>"

 response.write "Server port = " &
request.serverVariables("SERVER_PORT")

response.write "Server ime = " &
request.serverVariables("SERVER_NAME")

 response.write "IP adresa servera=" &
request.serverVariables("LOCAL_ADDR")

 response.write "Vrsta browsera prikljucenog serveru = " &
request.serverVariables("HTTP_USER_AGENT")

response.write "IP adresa klijenta = " &
request.serverVariables("REMOTE_ADDR")      %>
```

Refreširanje stranice :

```
'piše se izvan html-a
'redirect na klijentu

<% @Language="vbscript"%>
<%

response.addheader "Refresh","5;url=http://localhost/mario/zadatak1.asp"

%>
```

Kako izraditi redirect na dr. stranicu ?

```
'metoda redirect na serveru
'prvo se provrti cijela skripta, onda ide redirekcija

<html>
<head><%

response.buffer=true

response.redirect "http://localhost/mario/zadatak2.htm"

 %>
```

Definiranje nakon koliko vremena se stranica mora ponovno downloadirati !

```
<% @Language="vbscript"%>

<% response.expires=1

response.cachecontrol="public"

'definira nakon kolio minuta se stranica mora ponovno downloadati da bi se
refresirali podaci na stranici
'proxy serveri cachiraju stranice, tako da ne mogu cachirati koristimo

response.cachecontrol="private"

'ako stavimo "public" onda se stranice cachiraju kod klijenta

%>
```

Cookies

Cookeis služe za pohranjivanje podataka u tekstualnom obliku.

Specifikacije cookia:

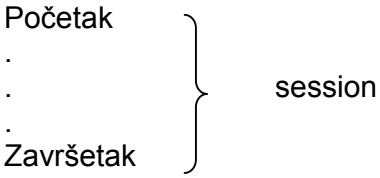
1. **cookie je specifičan za pojedine stranice—**što znači, da bi vratili cookie serveru, moramo upisati točnu URL adresu.

2. **sigurnost—**uvjet je da server I domena pašu. https:// protocol je stalno pod enkripcijom (sigurnost podataka, slovo ili riječ se zamjenjuje nekim drugim znakom).
3. **expiration—**može se zadati rok trajanja, a ako nije zadan rok trajanja, onda vrijedi samo tijekom session.
4. **text file –** nije za passworde, upise kreditnih kartica,……

Da bismo mogli raditi sa cookiima moramo imati **session**.
Session je vezan uz browser. Koliko imamo otvorenih browsera, toliko imamo sessiona, to znači da svaki browser ima svoj session.

Početak sessiona počinje kada browser zatraži stranicu I traje :

a) dok se browser ne zatvori
b) session je zaključen u kodu



Imamo dvije vrste potreba da se sačuvaju podaci :

1. **privremene prirode** – varijable, nizovi idr. u VBScriptu. Kada ASP prelazi u html, podaci se gube.
2. **permanentno** :
   a) **unutar session** — Cookies; Pohrana u bazu podataka; Tekstualne datoteke
   b) **između session** — Cookies; Session

Cookie je dio response objekta.

Da bi klijent I server mogli komunicirati, server za vrijeme sessiona sebi radi jedan ID preko kojeg pregledava, da li su još u vezi.

Ako želimo da se skripta koju je klijent zatražio cijela izvrti, pa pošalje na html, moramo upsati na početku asp skripte **response.buffer=true ,** a ako stavimo da je **false**, onda će skidati malo po malo.

Postoje dvije vrste cookia, I kako ih zadati :

1. **Jednostruki (Top-level cookies)** --- **response.cookies("Mario")="123"**, gdje su Mario I 123 proizvoljne vrijednosti

Primjer:

```
<%
response.buffer=true

response.Cookies("mario")="ozvatic"

a=request.cookies("mario")
response.write "Cookie = " & a
%>
```

2. **Višerazinski (multilevel cookies)** ---

response.cookies(“voditelj”)(“ime”)=”Tomislav”
response.cookies(“voditelj”)(“prezime”)=”Ocvirek” itd.

Primjer:

```
<%
response.buffer=true

response.Cookies("ime")("Ime")="Mario"
response.Cookies("prezime")("Prezime")="Ozvatic"
response.Cookies("telefon")("Telefon")="012726015"

a=request.cookies("Ime")
b=request.cookies("Prezime")
c=request.cookies("Telefon")

response.write a &"<br>"
response.write  b &"<br>"
response.write  c &"<br>"
%>
```

Brisanje cookia :

Response.cookies(“Mario”)=””

Čitanje cookia:

a = request.cookies(“Mario”)

response.write “Cookie = “ & a

Pregled I ispis svih cookia :

```
<% for each x in request.cookies
                response.write x & “ = ” & request.cookies
            next
                                                    %>
```

primjer:

```
<%
response.buffer=true

response.Cookies("ime")("Ime")="Mario"
response.Cookies("prezime")("Prezime")="Ozvatic"
```

```
response.Cookies("telefon")("Telefon")="012726015"

for each x in request.cookies
     response.write x & " = " & request.cookies(x) & "<br>"

          next
%>
```

Naredba za isčitavanje (ID) parametara:

token ne moramo nazvati ID, možemo ga nazvati proizvoljno.

Request.QueryString ("ime tokena")

Primjer kako iz html-a pozvati asp skriptu pomoću tokena :

Html stranica:

```
   <html>
<head>
   <title>Untitled</title>
</head>

<body >

<a href="http://localhost/mario/zad4.asp?ID=1">Link 1</a><br>
<a href="http://localhost/mario/zad4.asp?ID=2">Link 2</a><br>
<a href="http://localhost/mario/zad4.asp?ID=3">Link 3</a><br>

</body>
</html>
```

zad4.asp stranica :

```
   <%
  response.buffer=true

  Select Case Request.QueryString("ID")

                                  Case "1"
                Response.write "<b> Odabrali ste 1 !</b>"
          response.write "<center><table border='3' bgcolor='white'
bordercolor='green' width='40%'><tr><td>"
              response.write "Dobro dosli ! </td></tr></table>"

                                  Case "2"

            response.write "<input type='text' value='Bok,zurim'>"

                                  Case "3"

                 response.redirect "http://localhost/mario"

                                  end select
 %>
```

Zadavanje cookiu vrijeme trajanja :

Response.cookies("Mario").expires=#28/09/2001#

Zadavanje cookia samo za index stranicu:

Response.cookies("Mario").domain=http://www.yahho.com/

Unutar kojeg foldera će vrijediti cookie:

Response.cookies("Mario").path="/baza" **----** unutar foldera

Response.cookies("Mario").path="/" **--** unutar cijelog sitea

Forme

U ovome primjeru ćemo prikazati kako html stranicu preko forme povezati sa asp skriptom, tako da skripta izbaci podatke koji su upisani u formularu ,a aako ima koja greška da je ispiše.

Html stranica :

```
<html>
<head></head>
<body text=yellow>
<center>
<form name="form1" method="post" action="obradi.asp">
<table width=50% bgcolor=red ><tr><td width=60%>
Unesite ime:</td>
<td><input type=text name=ime></td></tr>
<tr><td>Unesite prezime</td><td><input type=text name=prezime></td></tr>
<tr><td>Muski</td><td><input type=radio name=spol value="m"> </td></tr>
<tr><td>Zenski</td><td><input type=radio name=spol value="z"></td></tr>
<tr><td colspan=2><center><input type=submit name=submit
value=submit></center></td></tr>

</form>
</body></html>
```

ASP stranica- obradi.asp

```
<%
ime=request.form("ime")
prz=request.form("prezime")
spl=request.form("spol")

if ime="" then response.write "Niste unijeli ime ! <br>"

if prezime="" then response.write "Niste unijeli prezime ! <br>"
```

```
if spl="" then response.write "Niste oznacili spol !<br>"

response.write "<b><i>" & ime &"<br>" & prz &"<br>"

if spl="m" then
 response.write "<br>" &" Rođeni ste u znaku kao slon !"
 else
  response.write "<br>" &" Rođeni ste u znaku kao koza !"
end if

%>
```

Validacija(provjera) formulara

Html stranica - formular

*Polja označena s * su obavezna!*

Ime*:

Prezime*:

e-mail*:

Pasword*:

Ponovi pasword*:

Spol: M Ž

Država: Država....

Grad:

Zip code:

Područje interesa:

IT Music
Software Hoby
Hardware Knjige
Internet Sport

Kako ste doznali o nama:

Od.....

Kako ste doznali o nama:

Pošalji Obriši

asp stranica

- **kako ispitati sva polja da li su pravilno unesena :**

```
<% @Language="VBScript"%>
<%

  'ovim if pitanjem pitamo da li je klijent poslao formular metodom get ili
   post. Ako je get, onda mu skripta ne obrađuje formular, a ako je post,
   onda zove proceduru validate, koja dalje nastavlja pregledavati da li su
   ostala polja ispravno ispunjena.

  If request.ServerVariables("REQUEST_METHOD")="POST" Then

 Call Validate()

  Else

    Response.Write "Access denied!"
  End If


   ' procedura koja obrađuje daljnje podatke iz formulara
   ' success je niz koji postavljamo da je true(polja su pravilno
     unesena)
   ' ako nešto nije uneseno, stavimo succes=false i server klijentu
     vraća poruku da nije nešto upisao

  Sub Validate()

    Dim Success
    Success=True
      FN=Request.Form("Ime")
      LN=Request.Form("Prezime")
      email=Request.Form("email")
      psw1=Request.Form("password1")
      psw2=Request.Form("password2")
      Request.Form("check")


      If Len(FN)=0 Then
        Response.Write "Niste unjeli ime!"
        Success=False
          End If

      If Len(LN)=0 Then
        Response.Write "Niste unjeli prezime!"
        Success=False
      End If

       ' emailcheck je funkcija koju pozivamo da provjeri da li je
         mail pravilno unesen

      If EmailCheck(email)=False Then
        Response.Write "Pogrešna e-mail adresa!"
      End If

       ' passcheck je funkcija koju pozivamo da provjeri da li je
         password pravilno unseen


        If PassCheck(psw1,psw2)=False Then
       Response.Write "Pasword mora sadržavati najmanje 5 znakova, a od toga
```

```
                    najmanje jedno veliko slovo i jedan broj!"
  End If
```

```
  ' ispisuje sa forme za što se kljient interesira, tj. koje je
    checkboxove ukljucio

  If Request.Form("check").Count=0 Then
    Succeses=False
  Else
    For i = 1 To Request.Form("check").Count
      Response.Write Request.Form("check")(i) & "<br>"
    Next
  End If
```

```
  End Sub
```

```
  'adr je zamjena za request.form("mail")
' Validacija e-mail adrese

 ' Zamislimo da je e-mail ispravno unsen(true), kasnije mu postavimo
false za svaki slučaj, ako je krivo unesen
```

```
  Function EmailCheck(adr)
  EmailCheck=True
```

```
'Varijabla valid sadrži sve znakove koji se smiju nalaziti u e-mail
adresi

       valid="abcdefghijklmnopqrstuvwxyz1234567890@._-"
```

```
 'adr je zamjena za request.form("mail")

      For i = 1 To Len(adr)
        If InStr(1, valid, LCase(Mid(adr, i, 1)))=0 Then
          EmailCheck=False
          Exit Function
        End If
      Next
```

```
  'Vraća poziciju na kojoj se nalazi @

                          m=InStr(1, adr, "@")
                               If m<4 Then
      EmailCheck=False
      Exit Function
    End If
    If InStr(m+2, adr, ".")=0 Then EmailCheck=False
```

```
  End Function
```

```
' funkcija za provjeravanje passworda
‘ Zamislimo da je password ispravno unsen(true), kasnije mu
postavimo false za svaki slučaj, ako je krivo unesen

    Function PassCheck(psw1,psw2)
      PassCheck=True

 'koja slova mogu biti u passwordu

        lett="abcdefghijklmnopqrstuvwxyz"

 ' koji brojevi mogu biti u passwordu

        numb="1234567890"

      For i = 1 To Len(psw1)
        If InStr(1, lett, LCase(Mid(psw1, i, 1)))>0 Then
            lett=lett+1
            Exit For
        End If
      Next
      If lett=0 Then Exit Function

      For i = 1 To Len(psw1)
        If InStr(1, num, LCase(Mid(psw1, i, 1)))>0 Then
            num=num+1
            Exit For
        End If
      Next
      If num=0 Then Exit Function

      If psw1<>psw2 Then Exit Function

     End Function
%>
```

Session

- object koji se kreira u trenutku kada klijent zatraži sa browsera neki ID ili informaciju.
- Služi za pohranu podataka (informacija za vrijeme sessiona
- On je tipa variant

Session(“naziv”)=vrijednost

Vrijednost- varijabla, string, niz, object

Session se gubi :

- kada se browser zatvori
- kada browser ne prihvaća session cookie
- kada je istekla

zaustavljanje sessiona : **Session.Abandon**

Pokazivanje koliko neka forma sadrži bytova:

Request.TotalBytes

Validacija prethodnog formulara pomoću sessiona :

```
<% @Language="VBScript"%>
<%
  Response.Buffer=True

  If request.ServerVariables("REQUEST_METHOD")="POST" Then
    Call Validate()
  Else
    Response.Write "Access denied!"
  End If

  Sub Validate()
    Dim Success
    Success=True
      FN=Request.Form("Ime")
      LN=Request.Form("Prezime")
      email=Request.Form("email")
      psw1=Request.Form("password1")
      psw2=Request.Form("password2")
      Request.Form("check")

      If Len(FN)=0 Then
        Response.Write "Niste unjeli ime!"
        Success=False
      Else
        Session("Ime")=FN
      End If

      If Len(LN)=0 Then
        Response.Write "Niste unjeli prezime!"
        Success=False
      Else
        Session("Prezime")=LN
      End If

      If EmailCheck(email)=False Then
        Response.Write "Pogrešna e-mail adresa!"
      Else
        Session("E-mail")=email
      End If

      If PassCheck(psw1,psw2)=False Then
        Response.Write "Pasword mora sadržavati najmanje 5 znakova, a od toga
najmanje jedno veliko slovo i jedan broj!"
      Else
        Session("Password")=psw1
      End If

      If Request.Form("check").Count=0 Then
        Succeses=False
        Response.Write "Odaberite najmanje jedno područje interesa!"
      Else
        Dim intrs()
        br = Request.Form("check").Count
        ReDim intrs(br)
        For i = 1 To br
          intrs(i)=Request.Form("check")(i)
        Next
        Session("Interests")=intrs
      End If
```

```
    Session("Spol")=Request.Form("gender")
    Session("Država")=Request.Form("lista1")
    Session("Grad")=Request.Form("grad")
    Session("Zip")=Request.Form("zip")
    Session("od")=Request.Form("lista2")
    Session("Komentar")=Request.Form("komentar")

    Response.Redirect "form_ispisi.zadatak.asp"

    End Sub


    Function EmailCheck(adr) 'Validacija e-mail adrese
      EmailCheck=True

      valid="abcdefghijklmnopqrstuvwxyz1234567890@._-" 'Varijabla valid sadrži sve
znakove koji se smiju nalaziti u e-mail adresi

        For i = 1 To Len(adr)
          If InStr(1, valid, LCase(Mid(adr, i, 1)))=0 Then
            EmailCheck=False
            Exit Function
          End If
        Next

        m=InStr(1, adr, "@") 'Vraća poziciju na kojoj se nalazi @
        If m<4 Then
          EmailCheck=False
          Exit Function
        End If
        If InStr(m+2, adr, ".")=0 Then EmailCheck=False

    End Function


    Function PassCheck(psw1,psw2)
      PassCheck=True

      lett="abcdefghijklmnopqrstuvwxyz"
      numb="1234567890"

      For i = 1 To Len(psw1)
        If InStr(1, lett, LCase(Mid(psw1, i, 1)))>0 Then
            lett=lett+1
            Exit For
        End If
      Next
      If lett=0 Then Exit Function

      For i = 1 To Len(psw1)
        If InStr(1, num, LCase(Mid(psw1, i, 1)))>0 Then
            num=num+1
            Exit For
        End If
      Next
      If num=0 Then Exit Function

      If psw1<>psw2 Then Exit Function

    End Function

%>
```

Ispisivanje sadržaja ispunjenog formulara

```
<% @Language="VBScript"%>
<%
  Response.Write "<p align='center'><br><br><font face='Arial'
size='5'><u><b>Čestitamo"
  Response.Write "</b></u><br><font size='3'>Registracija je bila
uspješna</font></p><hr>"
  Response.Write "<font face='Arial' size='2'><u>Jesu li ovo vaši
podaci</u></font></font><br><br>"
  Response.Write Session("Ime") & "<br>"
  Response.Write Session("Prezime") & "<br>"
  Response.Write Session("E-mail") & "<br>"
  Response.Write Session("Password") & "<br>"
  Response.Write Session("Spol") & "<br>"
  Response.Write Session("Država") & "<br>"
  Response.Write Session("Grad") & "<br>"
  Response.Write Session("Zip") & "<br>"

    intrs=Session("Interests")
    For i = LBound(intrs) To UBound(intrs)
      Response.Write intrs(i) & "<br>"
    Next

  Response.Write  Session("od") & "<br>"
  Response.Write Session("Komentar") & "<hr>"


  For Each C In Session.Contents
    itm=Session(C)
    If VarType(itm)>=vbArray Then
      Response.Write "<b>Interesi: </b>"
      For i = LBound(itm) To UBound(itm)
        Response.Write itm(i) & "<br>"
      Next
    Else
      Response.Write "<b>" & C & ": </b>"
      Response.Write itm & "<br>"
    End If
  Next
%>
```

Objekt : Application

- traje dok je server upaljen, a server je upaljen 24h na dan
- može pohranjivati različite vrijednosti

Pohranjivanje vrijednosti:

Application.Value(“naziv varijable”)=vrijednost

Čitanje vrijednosti:

Var=Application.Value(“naziv”)

Brisanje vrijednosti :

`Application.Value("naziv")=empty`

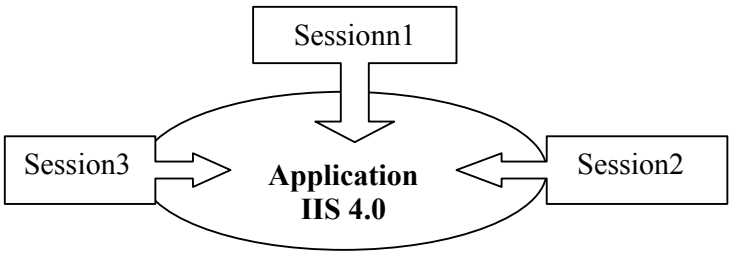
Da bismo za svakog klijenta napravili poseban session, tj. pristup stranicama moramo upisati unutar koda :

Application.Lock

promjena koja se događa na stranici (counter…)

Application.UnLock

Kod samog čitanja ne radimo zaključavanje, nego samo ako se kod pristupa klijenta nešto mora promjeniti ili dodati.



Prije svakog pristupa asp datoteci, server prvo pristupa datoteci po imenu **Global.asa,** da bi vidio, da li ima što za izvršiti prije pristupa stranici, jer u samoj asp stranici može biti nešto što je potrebno za izvršavanje prije pokretanja.

Global.asa – je pisana VBScriptom.Mora uvijek biti u glavnom root direktoriju

```
<script language="VBScript" runat="server">
                                                                    kod
</script>
```

Kod: procedura koja počinje sa:

```
Sub Session_OnStart          Sub Session_OnEnd
    naredbe                      naredbe
End Sub                      End Sub

Sub Application_OnStart      Sub Application_OnEnd
                          naredbe                  naredbe
End Sub                      End Sub
```

primjer: ***Counter- global.asa***

```
'global.asa mora biti u root direktoriju
'to je prva datoteka kojoj server pristupa, prije nego pristupi stranici
'ov je samo primjer, inace sluzi za pregledavanje cookia
'na klijentovoj strani, pa ako klijent nema cookie, on pokrece proceduru, koja
'odmah salje cookie klijentu ......itd

<script language="vbscript" runat="server">

sub session_onstart

 if isempty(application.value("caunter")) then
    application.lock
    application.value("caunter")=0
                  application.unlock
end if
   application.lock
     application.value("caunter")=application.value("caunter")+1
                   application.unlock
 end sub
 </script>
```

asp datoteka : default.asp

```
<% @Language="VBScript"%>
<%
Response.Write "Vi ste posjetitelj broj : " & application.value("caunter")
%>
```

Object: SERVER

Var = Server.MapPath(relacija)

-ova funkcija služi za pronalaženje folder aplikacije

relacija (parametar) :

- **root folder aplikacije (“..”)**
- **current folder aplikacije (“.”)**
- **izabir direktorija (“..”) & (“\direktorij\”)**

Sve vrijednosti koje se ponavljaju stavljaju se u global.asa - keširaju se u application_onStart

***.transfer* – metoda** - PWS 4 ne podržava

→ prije nego klijent dobije bilo kakav odgovor na traženu informaciju, na serverskoj strani dolazi do redirekcije I prenašanja podataka na drugu stranicu.

Server.Transfer “datoteka u koju prenaša”

include i SSI

→ služi za dodavanje vanjskih skripti na postojeću. Kod SSI-a može se ubaciti i unutar stranice, dok standardan include dolazi samo na početku stranice.

<!--#include file="datoteka"-->
<%….
→ kod includa server kod traženog upita pregleda sve skripte I sažme ih u jednu I izvršava upit.

.execute -PWS 4 ne podržava

→ poziva se kao podskripta unutar koda
→ radi bolje od include directive, jer se može ubaciti bilo gdje
→ razlika : include sažima skripte u jednu , pa ih zatim izvaršava, dok execute odmah izvršava skriptu tamo u kojem je kodu pozvana

Server.Execute ("ime.asp")

.ScriptTimeOut

→ određeno vrijeme koje zadajemo skripti da se izvrši
→ ako se u to vrijeme ne izvrši, skripta staje sa izvršavanjem

Server.ScriptTimeOut = 10

10—vrijeme zadano u sekundama

.HTMLEncode

→ enkodiranje znakova

Server.HTMLEncode ("koji znak")

```
response.write server.htmlencode("ž")
```

.URLEncode

→ služi za enkodiranje tokena I URL-a (ako u tokenima imamo ime I prezime, on enkodira ime I prezime tako da ih spaja , jer između ne smije biti razmaka)

var = Server.URLEncode("?ime=pero peric")

rezultat: ?ime=pero+peric

```
response.write server.urlencode ("jura juric")
```

.GetLastError PWS 4 ne podržava

→ vraća naziv greške koja se pojavila na serveru. Greške izvlači iz ASPError objekta koji u sebi sadrži nastale greške na stranici

Server.GetLastError

```
<%
dim pero

 set pero = Server.GetLastError   ' na pws 4 ne radi

 response.write pero.Description

 %>
```

Object : ASPError

→ sistem za prepoznavanje pogreški

Propertie – pomoću kojeg tražimo greške

- **.ASPCode → IIS error code – greška servera**
- **.Number → broj grešaka**
- **.Source → daje tekst linije koda s greškom**
- **.File → ime skripte(datoteke) gdje su nastale greške**
- **.Line → u kojoj liniji koda je nastala greška**
- **.Description → opis greške**

```
<%

dim pero

 set pero = Server.GetLastError  ' na pws 4 ne radi

 response.write pero.Description

 %>
```

Uzroci grešaka :

- **nemogučnost čitanja ili pisanja datoteke**
- **ako je datoteka prazna**
- **nema podataka (varijable, ili nešto nije definirano)**
- **pri obradi forme**

Tipovi grešaka :

- **Predprocesing → greška prije procesiranja ASP skripte (npr. Nema include datoteke)**

- **Script design time → tijekom pisanja skripte – greška u kodu**

a) **Syntax error → nije zatvorena petlja itd**
b) **Semantie error → logička greška (može se ukloniti jedino ručnom inspekcijom podataka) – krivo pisanje koraka, algoritama itd.**

- **Script runtime errors → vanjski utjecaji na skriptu**

Error handling

+ → linija 1
→ linija 2
+ → linija 3

→ ako je moguće da dođe do greške u nekoj liniji, preskoči liniju I kreni dalje

On Error Resume Next

→ isključivanje preskakanja linija

On Error Goto 0

FileSystemObject

→ služi za rad s datotekama, folderima

→ object se kreira sa : *Server.CreateObject("Scripting.FileSystemObject")*

FileSystemObject
|
|---- Drives – Drive
|---- Folders – Folder
|---- Files – File
'---- TextStream

Kod rada sa .txt datotekama moramo kreirati podobjekt *.OpenTextFile*

.OpenTextFile("ime",mode,true,0)

mode → 1- read
2 – write
8 – append (promjena)
da bismo snimili upisani tekst, moramo staviti na podobject
podobj.writeline (“tekst” ili varijabla)

Primjer : kako ispisati datoteke u folderu

```
<%
    Set fso = Server.CreateObject("Scripting.FileSystemObject")
    Set MyFiles = fso.GetFolder(Server.MapPath("."))

        For Each MyFile In MyFiles.Files
            Response.write MyFile.Name & "<BR>"
        Next
 %>
```

Primjer : kako ispisati datoteke u folderu kao linkove

```
<%
                    Set fso = Server.CreateObject("Scripting.FileSystemObject")
    Set MyFiles = fso.GetFolder(Server.MapPath("."))
        For Each MyFile In MyFiles.Files
            Response.write "<a href='"
            ' fizicka putanja
            FilePath = Server.MapPath(".") & "/" & MyFile.Name
            Response.write FilePath & "'>" & MyFile.Name & "<a><BR>"
                                Next
        ' umjesto Server.MapPath(".") mozemo staviti MyFiles.Name, a to je ime
          foldera
        ' tako dobijemo relativnu putanju
 %>
```

Primjer: kako izraditi .txt datoteku

Default.asp

```
<form name="form3" action="snimi.asp" method="post">

textarea name=”memo” rows=”20” cols=”50”>"
  </textarea>


   <BR> <BR>
   <input type="Submit" value="Posalji">        
   <input type="reset" value="Obrisi  "> <BR><BR>
   File Name:  

<input type=”text” name=”ime” ></input><BR><BR>   </form>
```

Snimi.asp

```
<%

a = request.form("memo")

if len(a)="" then
response.write " Nemoguce kreiranje datoteke !"
else
```

```
call datoteka1()
end if

sub datoteka1()
dim fs,ts
set fs=server.CreateObject("Scripting.FileSystemObject")
b=request.form("ime")

  Set ts=fs.OpenTextFile(Server.Mappath(b ,2,true)
  ts.writeline a
  ts.close
  response.write "Datoteka --><u>" & c &"</u> <--uspješno snimljena !"
  end sub

  %>
```

Primjer : kako otvoriti neku .txt datoteku, upisati nešto u nju I snimiti je ili samo napisati neku novu datoteku :

Default.asp

```
<form name="form1" action="default.asp" method="post" >
<select name="sel" ><option selected>

<%
dim fil
set fil=server.createobject("scripting.filesystemobject")
set file=fil.getfolder(server.mappath(".."))

for each files in file.files
 if lcase(right(files.name,4))=".txt" then

 response.write  "<option value='" & files.name
 response.write "'>" & files.name
 end if
  next
 response.write "</select>"
 %>

 <br><br>
 <input type="submit" name="open" value="Otvori">
 </form>
 <form name="form2" method="post" action="snimi.asp">
  <textarea name="pp" rows="20" cols="80">

<%
 tekst=request.form("sel")
 response.write citaj(tekst)

 function citaj(tekst)
 if tekst="" then exit function

 dim citajte , citati
 set citajte=server.CreateObject("Scripting.FileSystemObject")
 set citati=citajte.OpenTextFile(Server.Mappath("..") & "\zvata\datoteke\" & →
→ tekst,1,true,0)
    citaj = citati.readall
  citati.close
 end function
 %>

 </textarea><br><p><center>
 <input type="submit" name="go" value="Snimi kao:">
```

```
<%
 response.write "<input type='text' name='tt' value='" & tekst &"'>"
  %>
</form></p>
```

Snimi.asp

```
<%
a = request.form("pp")
if len(a)="" then
response.write " Nemoguce kreiranje datoteke !"
else
call datoteka1()
end if

sub datoteka1()
dim fs,ts
set fs=server.CreateObject("Scripting.FileSystemObject")
c=request.form("tt")

  Set ts=fs.OpenTextFile(Server.Mappath("..") & c ,2,true)
  ts.writeline a
  ts.close
  response.write "Datoteka --><u>" & c &"</u> <--uspješno snimljena !"
  end sub
    %>
```

Randomize

Randomize **- inicijaliziramo da pokrećemo random mašinu**
RND **- izbacujemo nešto iz mašine**

Primjer :

```
<%
 randomize
a=(rnd*10+1)
response.write a
%>
```

svaki puta kada refreširamo stranicu ispisuje nam neki drugi broj u razmaku od jedan do 10

Isto tako možemo randomizirati I slike, tako da se svaki puta pojavi druga slika:

```
<%
randomize

a=int(rnd*10+1)

b=cstr(a)

response.write "<img src='slike/" & b & ".gif'>"


%>
```

Kreiranje **Dictionary** objekta

- **služi za pohranu niza riječi**
- **radi kao riječnik (u txt datoteku ili bazu podataka upišemo pojmove I njihova značenja I pomoću njega pretraživamo pojmove)**
- **npr. Ja upišem 'asp' , on pretražuje I izbaci active server pages**

Set var=Server.CreateObject("Scripting.Dictionary")

Dodavanje podataka :

Set dic=server…..

ključ vrijednost

dic.Add "IIS" "Internet Information Server"
dic.Add "ASP" "Active Server Pages"
……

Iščitavanje podataka :

a = dic("IIS")

ako je pohranjen object, onda set a = dic("IIS")

pohrana u session :

set session("DIC")=dic

određivanje ulaznih parametara (da li će se razlikovati a od A ili neće) :

dic.CompareMode= 0 ili 1

0 – binarni a ≠ A
1 – tekstualni a = A

brisanje određenog ključa :

dic.Remove ("IIS")

brisanje svega :

dic.RemoveAll

provjera postojećeg podatka :

if *dic.Exists(ASP)* then…
…

izlistavanje popisa svih ključeva :

kljucevi = dic.keys →niz

For i=Lbound(kljucevi) to Ubound(kljucevi)

Response.write kljucevi(i)

Izlistavanje popisa svih vrijednosti :

vrijednosti = dic.items →niz

For i=Lbound(vrijednosti) to Ubound(vrijednosti)

Response.write vrijednosti (i)

Kada petlja pretraži do kraja datoteka I da bi ispisala nema vrijednosti, moramo koristiti :

If *dic.AtEndOfStream* then…
…

Primjer kako pretraživati I upisivati nove pojmove :

Dictionary.asp

```
<%
b=request.form("klj")
c=request.form("vri")

' upisivanjeu postojeću txt datoteku

if len(b)>0 and len(c)>0 then
                    set fs=server.createobject("scripting.filesystemobject")
                    set ts=fs.opentextfile(server.mappath(".") & "\kratice.txt",8)
                                    ts.writeline b
                                    ts.writeline c
                  ts.close
                  set fs=nothing
end if


'pretraživanje txt datoteke

set dic  = server.createobject("scripting.dictionary")
set dat  = server.createobject("scripting.FileSystemObject")
set dict = dat.opentextfile(server.mappath(".") & "\kratice.txt",1)

do
```

```
  kljuci=dict.readline
  vrijednosti=dict.readline

  If Len(kljuci)>0 then dic.add kljuci,vrijednosti
loop until dict.atendofstream

a=request.form("ab")

if dic.exists(a) then
   response.write "<center><font color=blue size=4>" & a & " :  </font>"
   response.write "<font color=red size=4>>> " &dic(a) &" <<</font>"
elseif len(a)>0 then
  response.write "Pojam ne postoji !"
end if

dict.close
set dat=nothing
set dic=nothing

%>

 ' forma za pretraživanje i upisivanje novih pojmova

 <center>
 <form name="" action="dictionary.asp" method="post">
<input type=text name=ab>
<input type="submit" value="Trazi">
</form></center>
<p align=left>
<form name="pep" method="post" action="dictionary.asp">
Unesite kljucnu rijec :<input type="text" size="10" name="klj">
<br>
Unesite vrijednost kljucne rijeci :  <input type="text" name="vri">
<input type="submit" value="Snimi">
</form></p>
```

Objekt za slanje e-maila

```
Set mail=Server.CreateObject("Persist.MailSender")

mail.Host = "mail.iskon.hr"       'smtp mail server
mail.From = "mozvatic@inet.hr"       'od koga
mail.Subject = "Proba"
mail.Body = "Kako želimo da izgleda dobiveni mail"
mail.AddAdress = "vxbcv@SDfg"  'može ih biti koliko želimo,
                                ali uvijek novi addadress
mail.AddAtachment "c:\virus.exe"

On Error Resume Next
mail.send

if Err<>0 then response.write "Nije prošlo"!

Set mail=nothing
```

4. cjelina ***Access – relacijske baze podataka***

Autor : Ocvirek Tomislav (c)1997 - 2001.

Uvod

Baza podataka (grč. ***basis*** **–** osnova, temelj, skladište, spremište) je skup podataka koji se odnosi na određene objekte (entitete) i podatkovno ih opisuje. **Entitetom** smatramo **sve što možemo opisati**.

Npr. u kontekstu baze podataka '**Knjižnica'** to mogu biti :

- članovi (subjekti),
- knjige (objekti) i
- posudbe (događaji, procesi).

Svaki entitet opisuje se nizom podataka (informacija) koji su smješteni u jednu (najčešće) ili više tablica. Tablice su idealni oblik za smještaj podataka u bazu. Možemo reći da se podaci 'spremaju' u tablice i na taj način ostaju sačuvani. Baza podataka je pohranjena na disk u obliku jedne (npr. Access) ili više datototeka. Tablice svojim oblikom moraju omogućiti pohranu svih potrebnih podataka za opis jednog entiteta.

Npr. baza podatka '**Adresar'** može sadržavati jednu tablicu **tblAdresar** koja opisuje entitet 'osobu' te sadrži slijedeće podatke: *Ime, Prezime, Telefon, Fax, E-mail i Adresu*. Ti podaci zovu se polja (fields) i zajedno čine jedan zapis (record). Na slici vidi se tablica tblAdresar. Primjetite: Stupci (Columns) su polja, a redci (Rows) su zapisi.

tblAdresar:

Zapis br.	Ime	Prezime	Telefon	Fax	E-Mail	Adresa
1	Tomislav	Ocvirek	098/619-721	-	tocvirek@inet.hr	-
2	Ana	Perković	6623-458	-	ana.perkovic@zg.hinet.hr	-
.... N						

Relacijska baza podataka (grč. ***relatio*** – izvještaj, zapis, veza, doticaj) je baza u kojoj su podaci povezani relacijama čija je uloga da se izbjegne zalihost (redundancija, ponavljanje) podataka. Osim toga relacijama se osigurava integritet podataka. Baza podataka je izgrađena na teorijskom modelu, a **Access** utjelovljuje efikasnu i vjernu primjenu tog modela (iako treba reći da se složeni matematički model ne može 100% izvesti u praksi zbog njegove apstraktne naravi).

Microsoft Access nije baza podataka već ***Relationship Data Base Management System*** (RDBMS), dakle *mašinerija* (engine), program za

upravljanje bazom podataka. Access **na zadani način** podatke smještene u tablicama **obrađuje i** pri tom **poštuje integritet** relacijskog modela.

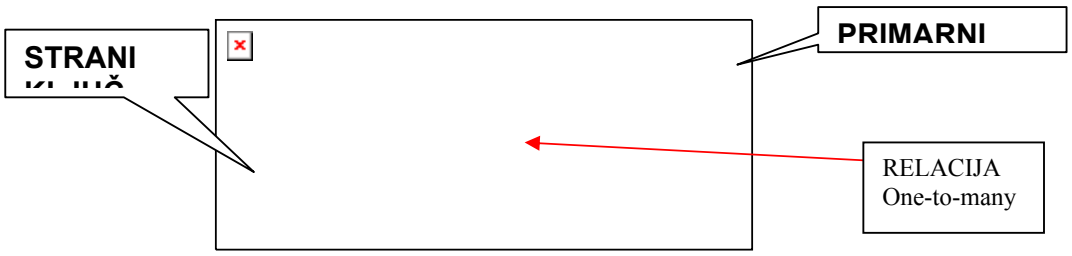
Do konačnog relacijskog modela baze dolazimo tako da potrebne podatke smještene u većim tablicama podvrgnemo normalizaciji (postoji više normalizacijskih shema) i poštujući pravila normalizacije razvijemo idealan oblik baze. To rezultira pretvorbom velikih tablica u veći broj manjih tablica koje su međusobno relacijski povezane. Time se izbjegava ponavljanje podataka i olakšava unos, održavanje i pretraživanje podataka.

Zašto je **relacijski model** bolji od klasičnog '**tabelarnog modela'**? Pogledajmo na primjeru: imamo jednu veliku tablicu koja sadrži podatke o **proizvodima :**šifra proizvoda, naziv, vrsta, cijena, naziv tvrtke dobavljača, adresa dobavljača, telefon dobavljača i odgovorna osoba. Dakle svi ti podaci vezani uz jedan proizvod (uključujući i podatke o dobavljaču).

Šifra proiz.	**Naziv**	**Vrsta**	**Cijena**	**Naz.tvrtke**	**Adresa**	**Telefon**	**Odg.osoba**
1023345	LCD 4GP	Monitor	2345 Kn	I.P.M.	Matoša 12	2345-678	M.Marulić

Zamislite da imamo 2500 zapisa i od toga 325 zapisa imaju istog dobavljača. Dobavljač je promjenio broj telefona. To znači, premo sadašnjem modelu moramo taj broj ispraviti 325 puta (na 325 različitih zapisa u bazi). Time se izlažemo mogućnosti da pogriješimo pri ispravljanju podataka (naporno, zar ne ?). Osim toga zašto se svi ti podaci moraju 325 puta ponavljati ? To je nepotrebno ako je moguće da taj podataka navedemo samo jednom. Time bi uštedili na diskovnom prostoru, brzini rada baze (performance) jer bi proporcionalno bi porasla i brzina sortiranja, pretraživanja – ukratko - obrade podataka !

Ako bismo te podatke rastavili u dvije manje tablice (***normalizacijom***) od kojih bi jedna sadržavala podatke o proizvodima, a druga podatke o dobavljačima, tada smo stvorili relacijski model. Tablice su međusobno povezane relacijom. To je učinjeno tako da je u obje tablice dodano polje '**Šifra dobavljača' (ID)** a zatim je preko tog polja ostvarena relacija između tablice Proizvodi i tablice Dobavljači.



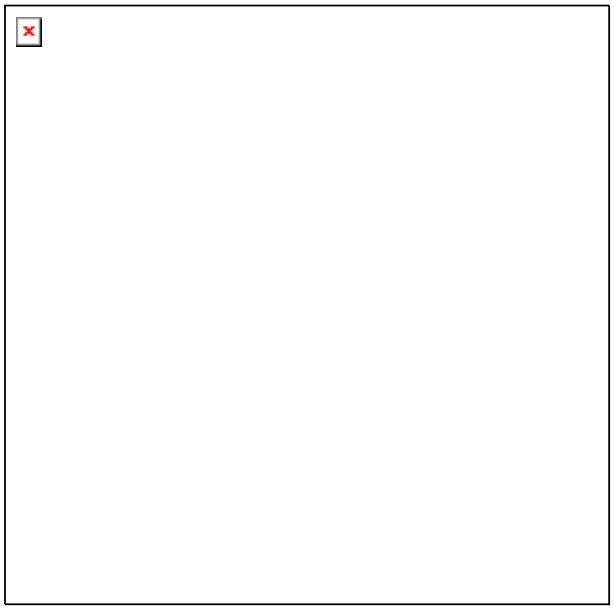
Polje '**Šifra dobavljača'** u tablici Dobavljači mora osigurati jedinstvenost svake šifre (mora biti zaključano tj. biti ***primarni ključ – primary key***), a polje *šifra dobavljača* u tablici Proizvodi (***strani kjuč*** – ***foreign key***) ne smije biti zaključano jer se šifra smije ponavljati tj. moguće je da više proizvoda dolazi od istog dobavljača. To je najšešći oblik relacije ONE-TO-MANY. Dakle, jednom zapisu iz tablice Dobavljači odgovara više zapisa iz tablice Proizvodi ili ponovo da to kažemo na drugi način: jedan dobavljač može dobavljati i više od jednog proizvoda.

Koje su prednosti ovakvog pristupa ? Kao prvo, sve podatke bilo bi dovoljno unjeti samo jednom. Dakle, i u slučaju promjene nekog podatka (npr. telefona dobavljača) dovoljno je tu promjenu učiniti samo jednom i na jednom mjestu – u tablici dobavljača! Jednostavnije je administriranje i održavanje baze, očito !

Kada bi stvorili relaciju između dva primarna ključa, to bi bila relacija ONE-TO-ONE, tj. jednom zapisu iz prve tablice odgovara samo jedan zapis iz druge tablice. Takvim relacijama povezujemo one tablice koje opisuju isti entitet.

Početak rada s Accessom

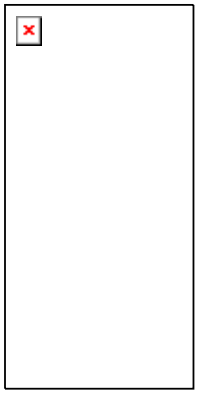
Pri pokretanju Access-a, u Access-ovom se prozoru otvara dijalog *Microsoft Access* koji omogućava kreiranje nove prazne baze podataka (Blank database), kreiranje nove baze podataka pomoću *Database Wizard* ili otvaranje postojeće baze. Ako kreiramo novu bazu podataka, Access će prvo kreirati datoteku baze podataka na disku (naravno, pitat će nas za ime pod kojim želimo snimiti bazu), a tek potom je moguće raditi s novom bazom.Nakon što smo otvorili postojeću ili kreirali novu bazu podataka, u Accessovom prozoru se otvara *Database* prozor, koji sadrži šest kartica – svaka od njih služi za upravljanje jednim od objekata Access baze podataka.



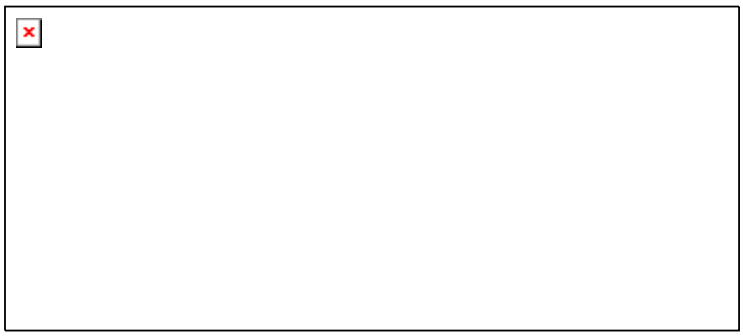
Objekti Access

Svaka baza sastoji se od nekih osnovnih ***ciglica – tj. objekta.*** Tako se Access sastoji od slijedećih skupina objekata:

Access 2000:



Access 97:



1. Tablice (Tables) - u njih unosimo podatke, a sastoje se od **redaka** – **rows**, slogova, zapisa, recorda i **stupaca** – polja, **fiedls**, podatka). Tablice su jedini fizički nosioci podataka u bazi podataka. Svi ostali objekti služe za ažuiriranje i pregled tih podataka.

2. Upiti (Queries) - pomoću njih na temlju postavljenih kriterija, pretražujemo jednu ili više tablica i vadimo podatke od interesa i time dajemo svrhu bazi podataka, a to je da u svako vrijeme uz minimalno truda imamo potreban podatak na raspolaganju. Zapisi koji zadovoljavaju zadane kriterije (npr. da Prezime počinje slovom 'P') stvaramo nove privremene tablice (setove zapisa, tj. Recordset-ove) koje su promjenjive u vremenu a mijenjaju se sukladno promjenama u izvornim permanentnim tablicama. Upiti su pogodni za pregledavanje i stvaranje izvještaja, ali i za mnoge druge operacije koje trajno mijenjaju strukturu, mjesto i oblik podataka. Query su ključni dio RDBMS mašinerije. Queryi se mogu graditi na dva načina: pisanjem u SQL (Structured Query Language) koji je prvenstveno i izmišljen za rad s relacijskim bazama podataka (pregled, filtiranje, izmjenu, dopunu podataka). No, bez straha, QBE (Query by example) je vizualni pristup postavljanje upita i ne iziskuje poznavanje SQL sintakse: vi doslovno nacrtate upit (dobro, morate nešto i napisati – kriterije), a QBE će na temelju vaših zahtjeva generirati SQL-kod (kojeg možete i pogledati ; zgodno za učenje, zar ne ?). Upiti mogu biti pasivni (SelectQuery, CrossTabQuery itd.) i aktivni (MakeTableQuery, DeleteQuery, AppendQuery, UpdateQuery itd). Ovi drugi su znatno opasniji jer oni fizički mijenjaju tj. dopunjuju i brišu podatke iz vaših tablica !

3. Forme ili obrasci (Forms) - grafička sučelja koja olakšavaju rad s bazom kao i unos podataka. Ona su zahvalna za osjećaj udobnosti pri radu s bazom, kako vas tako i budućih korisnika vaših baza koji, pretpostavljamo ne razlikuju operativni sustav od hardware-a.

4. Izvještaji (Reports) - *ušminkani* dokumenti koji prikazuju tablice i upita grafički i pregledno, namjenjeno ispisu na papir (da možete nešto šefu staviti na stol).

5. Web stranica (Pages) - s verzijom Access 2000 dolaze i objekti **Pages** koji omogućuju da se za unos i pregled stranica kreiraju web stranice koje se mogu pokretati i koristiti kroz sučelje Internet explorer-a kao klasična web stranica bez potrebe za Access-ovim sučeljem. To je izuzetno pogodno za Intranet (velike tvrtke s razvijenom lokalnom mrežom).

6. Makroa (Macros) - automatsko ponavljanje čestih radnji (tipično za Microsoftove proizvode). Može poslužiti, ali nije neophodno ako dobro dizajniramo i isprogramiramo sučelje baze. Ipak accessovci su ih zavoljeli, pa ćemo i njih upoznati.

7. Modula s kodom (Modules) - samo za teške programere, dakle, za vas. Manevriranje bazom, postavljanje raznih uvjeta itd. Ovdje nam pomaže znanje iz Visual Basica i VBA-koda. Tu smještamo opće procedure i funkcije koje koristimo pri obradi podataka. Iskreno, nećete imati potrebu često pisati module jer kod najčešće vezan uz forme (odnosno nalazi se u modulu forme i odgovara događajima na formi).

Tables - kreiranje tablica

Novu tablicu u bazi kreiramo u dijalogu koji dobijemo klikom na gumb *New* u *Table* kartici i odabirom *Design View* pogleda. U stupcu *Field Name* upisujemo ime polja tablice, a u stupcu *Data Type* odabiremo tip podataka za to polje tablice. U kartici *General* u donjem dijelu dijaloga podešavamo ostala svojstva za svako od definiranih polja:

Field Name
ime polja, vrlo važno je kako imenujemo polje (izbjegavati dijakritičke znakove), pa je uputno zapisati ili isprintati nazive svih polja u vašim tablicama

Data Type
tip podataka. Najčešće je to **Text** (90 %) maks. dužine 255 znakova, za vremenske podatke valja odabrati **Date/Time,** za valute i novčane iznose **Currency**, za količinske i općenito brojevne veličine **Number,** za dugačke tekstualne opise **Memo**, za logične podakte tipa Da/Ne tip **Yes/No**, za internet adrese **Hyperlink**, za OLE objekte **OLE.** Napomena: JMBG, telefonski brojevi i sl. uvijek će biti **Text** polja ! **AutoNumber** je tip podatka koji predstavlja stalno rastući cijeli broj i pogodan je za upotrebu kao ID polje kada ne želimo unositi podatke u to polje i kada se kasnije **nećemo nikada** pozivati na to polje tijekom pretraživanja baze.

Description – opis polja, vaš podsjetnik

- **Format** – definiranje oblika podataka u polju; ovisi o tipu podataka polja. Npr. Number može biti Integer, Long (cijeli broj), Single (Decimalni) itd. Datum može biti kraćeg ili dužeg formata prikaza itd.

- **InputMask** – kreiranje maske za unos podataka; masku za unos kreiramo tako da kliknemo u područje za upis InputMask svojstva, pri čemu se uz desni rub područja pojavi gumb *Builder-a*. *Builder* (graditelj) će se pojavljivati na više mjesta u Access-u, a on nam olakšava kreiranje potrebnog izraza pokretanjem čarobnjaka ili otvaranjem dijaloga *Expression Builder*. U ovom slučaju *Builder* pokreće čarobnjaka za kreiranje InputMask svojstva. (npr. maska za unos datuma u koji prikazuje datum u obliku 15-svi-98, mogla bi izgledati ovako: __-___-__, a maska za unos tel. broja formata 098/619-721 će izgledati ovako:
999\/999\-9999.

- **Caption** – naslov polja u prikazu sadržaja tablice (podrazumijevanbo je jednak imenu polja)

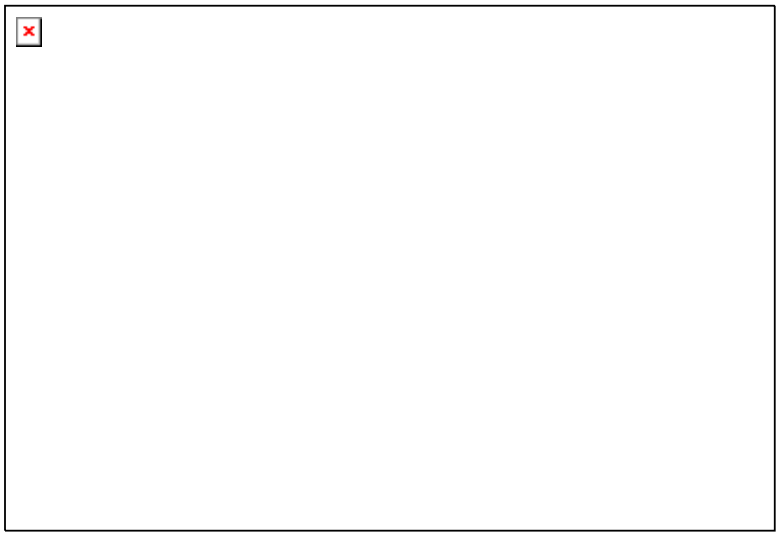
- **DefaultValue** – podrazumijevana vrijednost polja (automatski se unosi u polje ako pri unosu sloga u tablicu izostavimo podatak za to polje). Npr. ako je polje datum i želimo da svaki put kad se stvori novi zapis da se automatski upiše današnji (tekući) datum, tada ćemo za Default value staviti vrijednost **Date() –** to je ugrađena funckija Accessa koja vraća sistemski datum

- **Validation Rule** – definiranje pravila (uvjeta) koja moraju zadovoljiti podaci da bi bili unjeti u polje. Npr. cijena proizvoda ne smije biti nula: **cijena > 0**

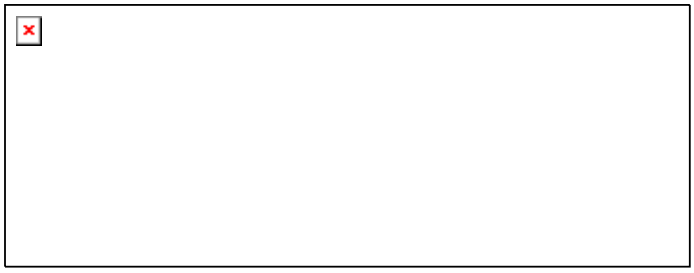
- **Validation Text** – upozorenje koje se ispisuje ako pri unosu podataka narušimo Validation Rule.
Npr. **"Cijena ne može biti nula !"**

- **Required** – definiramo je li pri unosu podataka obavezno unijeti podatak za to polje, ili ga je dozvoljeno izostaviti

- **Indexed** – omogućava definiranje indexa na polje (*No duplicates* index osigurava jedinstvenost podataka u polju, a *Duplicates OK* dozvoljava duplikate vrijednosti u polju). Mada nije nužno, uputno je u svakoj kreiranoj tablici definirati primarni ključ radi osiguravanja integriteta relacije. Primarni ključ definiramo tako da označimo sva polja od kojih će se ključ sastojati i pokrenemo naredbu **Edit → primary Key** ili jednostavno kliknemo gumb sa slikom ključa na traci s alatima.



Moguće je definirati i svojstva na razini čitave tablice, a ne samo na razini jednog polja. Time je npr. omogućeno definiranje pravila za unos (Validation Rule) koja će povezivati više polja tablice. Ova svojstva definiramo u dijalogu ***Table Properties*** koji dobijemo naredbom **View → Properties**. Ako bismo npr. imali tablicu sa stupcima *maloprodajna cijena* i *veleprodajna cijena* moguće je defiirati uvjet: **maloprodajna cijena > veleprodajna cijena**.

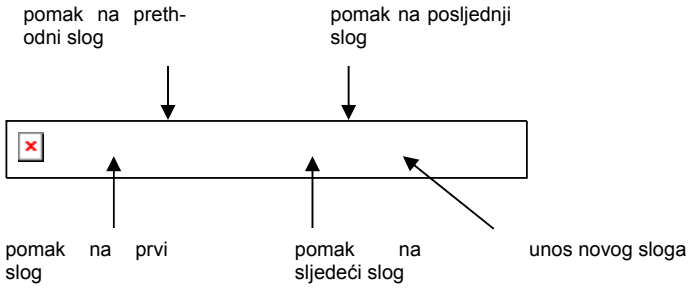


Kada smo definirali polja tablice i njihova svojstva, snimamo tablicu naredbom **File → Save As** ili klikom na gumb sa slikom diskete na traci s alatima, pri čemu dajemo ime tablici.
Preporuka je svakoj tablici dodati prefix **tbl**. Tako npr. tablica Adresar zvati će se **tblAdresar.**

Unos podataka u tablicu

Podatke u tablicu baze podataka možemo unositi na više načina. Najjednostavniji način, koji ćemo ovdje opisati je unošenje direktno u *datasheet* prikaz tablice. Moguće je podatke unositi i preko formi kreiranih u Access-u (o tome kasnije) ili iz npr. iz forme Visual Basic aplikacije (o tome također kasnije). Za unošenje podataka u *datasheet* prikaz tablice potrebno je u *Tables* kartici *Database* prozora označiti željenu tablicu, te kliknuti gumb *Open*. Podaci se jednostavno upisuju u odgovarajuće ćelije tablice. Među ćelijama tablice krećemo se pomoću strelica ili tipkom TAB za slijedeći ćeliju , a Shift+TAB za prethodnu ćeliju. Među

unesenim slogovima tablice moguće je kretati se i pomoću gumba sa strelicama koji se nalaze pri dnu prozora. To su navigacijski gumbi:



Podatke u tablici je moguće kopirati, isijecati i pomicati standardnim *Cut, Copy, Paste* naredbama, kao i *drag&drop* operacijom miša:

Relationships – uređivanje relacija između tablica

Ključevi

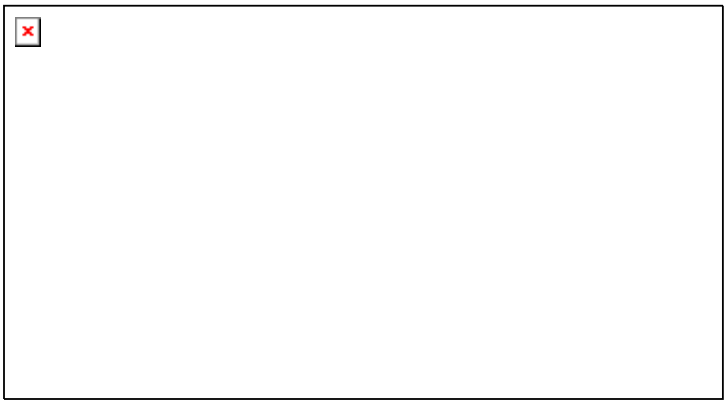
Primarni ključ (PRIMARY KEY) postavlja se na jedno ili više polja u tablici. Njihova uloga je da onemogućuju ponavljanje podataka u tim poljima. Njima osiguravamo jedinstvenost svih podataka u poljima na koja smo ih postavili (**jedan ključ = nema duplikata u polju zaključanom polju**).

Dakle polja poput **JMBG, ID, Šifra artikla, Članski broj** itd. su polja na koja svakako treba postaviti ključ jer ti se podaci ne smiju ponavljati. Zamislite dva čovjeka u knjižnici s istim članskim brojem !?! Zbrka ?

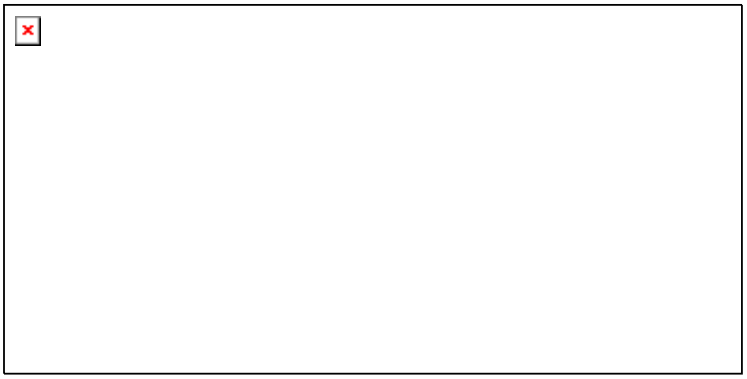
Ako u jednoj tablici na više polja postavimo ključeve tada kreiramo **složeni ključ.** Sada svako polje unutar složenog ključa može i ne mora imati duplikate, ali ne mogu postojati dva zapisa u jednoj tablici s istom kombinacijom podatraka tj. da imaju sve podatke identične u sva tri zaključana polja.

Indexi

Index-i su onaj su onaj dio mašinerije RDBMS-a koji ubrzavaju sortiranje i pretraživanje podataka u velikim tablicama kreirajući nama korisnicima nevidljive index tabele. Da li je neko polje indeksirano ili ne određujemo u Field Properties pojedinog polja u Design View-u tablice (vidi sliku). Indexi su ti koji dopuštaju ili zabranjuju duplikate u pojedinom polju (vidi sliku):



Indexe je moguće urediti i na razini čitave tablice **View/Indexes** gdje još određujemo smjer sortiranja indeksiranih polja (rastući ili padajući sort). Svako polje s ključem (primarni ključ ili dio složenog ključa) je automatski indeksirano.



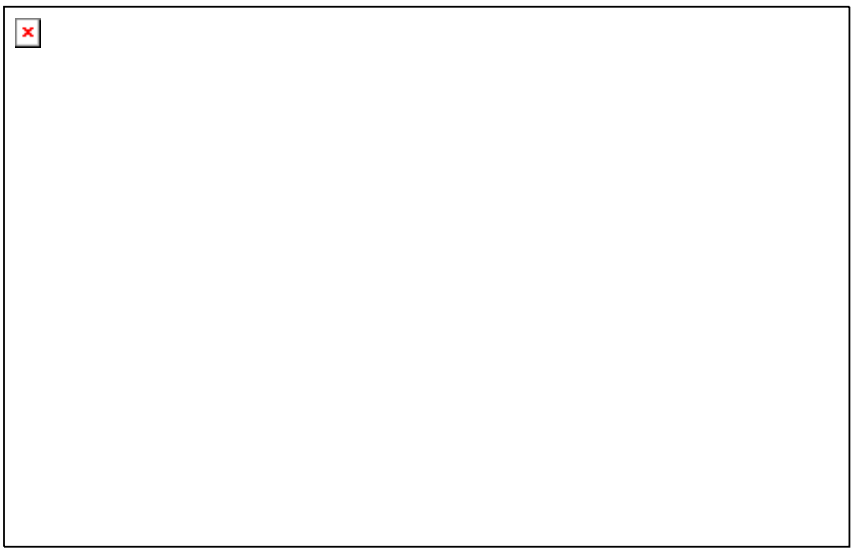
Relacije

Relacije (veze) između tablica mogu biti permanentne na razini baze podataka i njih definiramo u prozoru **Tools/Relationships** , dok privremene relacije za potrebe izvršavanja query-a (upita) definiramo pri stvaranju samih querya i one se nakon prekida izvršavanja query-a brišu. Privremene relacije su podređene permanentnima i na temelju njih se definiraju.
U prozoru Relationship opcijom **Show Table** biramo samo one tablice između kojih zaista želimo kreirati relacije (najčešće je to većina tablica). Relacije se

kreiraju jednostavnoim **Drag'n'Dropom** polja iz prve tablice na ciljano polje u drugoj tablici (svejedno je iz koje tablice krećemo jer je relacija obostrana).

Relacije na razini baze Tools / Relationships:

Relacije definirane u Queryima (tzv. 'Joinovi'):

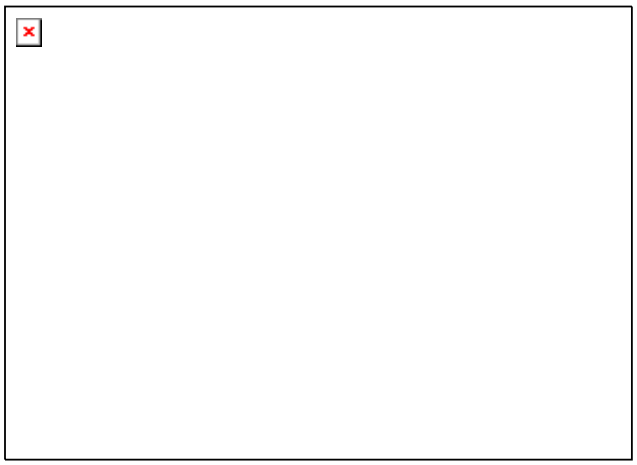


U kontekstu relacije definirane između dviju tablica, primarni ključ (**Primary Key**) nalazi se u glavnoj, **parent** tablici, dok se strani ključ (**foreign key**) nalazi u podređenoj, **child** tablici. Ovdje je naglasak na terminologiji zbog eventualne dodatne literature koju ćete koristiti u radu.

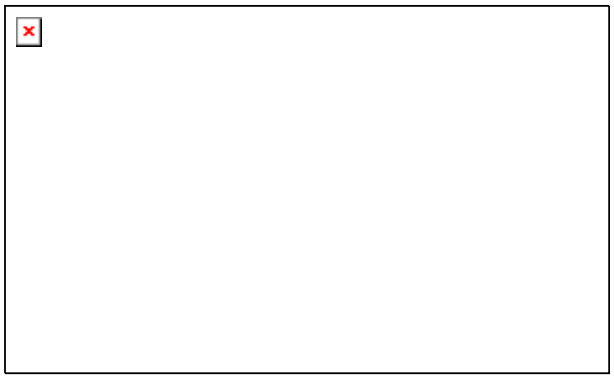
Postupak uređivanja relacija između tablica

Dakle, u Accessu je moguće definirati povezanosti među podacima na način da uspostavljamo veze među tablicama preko polja koja sadrže jednaku vrstu podataka. Tako na primjer u bazi 'Trgovina' i tablica *Proizvod* i tablica *Prodaja* sadrže polje *Sifra,* u koje se spremaju podaci o šifri proizvoda. Očito su ove dvije tablice povezane preko tih polja.
Veze među tablicama uređujemo u dijalogu *Relationships* (**Tools → Relationships)**. Postupak je sljedeći:

1. **Tools → Relationships**
2. odaberemo sve tablice baze među kojima želimo editirati veze (najčešće sve tablice baze)
3. u prikazu jedne od tablica među kojima uspostavljamo vezu uhvatimo mišem polje preko kojega se ostvaruje veza među tablicama, odvučemo ga do tablice s kojom uspostavljamo vezu i otpustimo tipku miša iznad odgovarajućeg polja u toj tablici.
4. pojavljuje se dijalog Relationships (možemo ga pozvati iz kontekst menija preko opcije **Edit Relationship**) – u njemu podesimo opcije na željeni način:

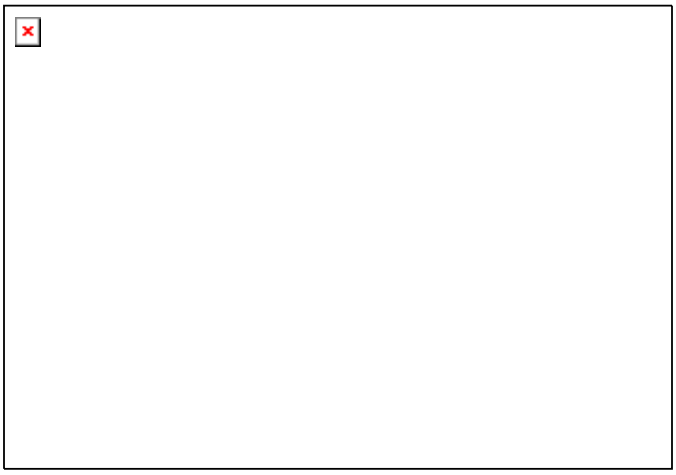


Nakon što definirate vezu među tablicama Access prikazuje tu vezu slikovito na način vidljiv na slici 2. Ovdje je još bitno zapamtiti da ono polje u vezi koje treba činiti stranu **1** veze **1–1** ili **1–N** mora biti primarni ključ ili mora imati definiran jedinstveni indeks (indeks koji ne dozvoljava dupliciranje vrijednosti u polju).



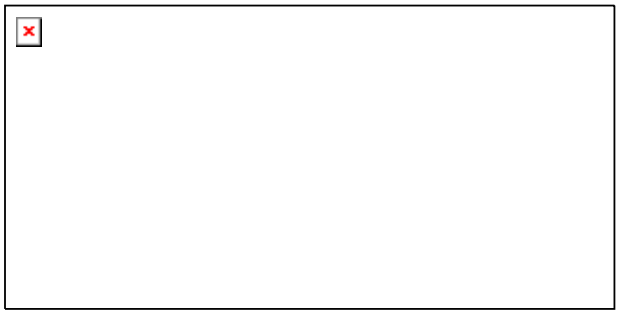
Referencijalni integritet

Referencijalni integritet osigurava postojanost svih podataka u bazi. Npr. u bazi podataka Videoteke u tablici **Posudbe** ne možemo imati naveden zapis sa brojem iskaznice člana ako taj član i taj broj nisu istovremeno nevedni i u tablici **Članovi** pod istim brojem iskaznice. Dakle, svaki podataka u ovakvoj vrsti relacije mora imati pokriće. Ako želimo obrisati jednog člana iz tablice članovi jer se ispisao, i pri tom želimo izbrisati sve zapsie o njegovim dosadašnjim posudbama iz **tablice posudba**, uključiti ćemo opciju ***Cascade delete related records***. ***Cascade update related fields*** će reagirati na promjenu u izvornoj tablici (s primarnim ključem) i osvježiti promjenjeni podatak u svim ostalim tablicama kaskadno u relacijskim vezama u dotičnom polju s ciljem da i one budu dosljedne izvornoj. Preporučljivo je ovu opciju uključiti na samom kraju kreiranja baze podataka, dakle prije neposrednog 'puštanja baze u upotrebu' jer u protivnom ta nam opcija može postati kamenom spoticanja tijekom raznih testiranja, debuggiranja i razvoja baze.

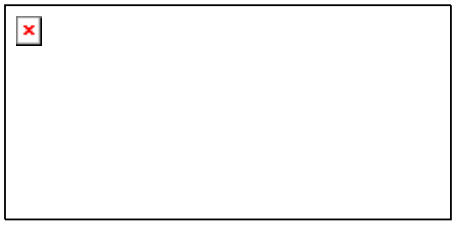


Join Type – podvrste relacije

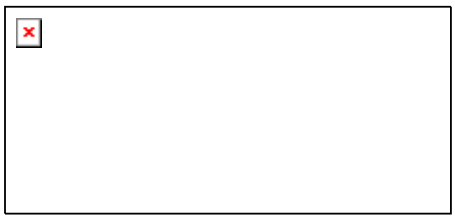
Postoje tri podvrste relacije (tri tipa združivanja). Očito, svrha relacije je da se iz dviju ili više tablica izdvoje zapisi koji imaju nešto zadjedničko, u nekoj su vezi. Npr. tablica tblPosudbe sadrži popis **signatura** svih knjiga koje su posuđene određenog datuma, a tablica tblKnjige sadrži popis svih knjiga koje postoje u knjižnici zajedno s pripadajućim **signaturama**. Naravno, mi vezu ostvarujemo upravo preko polja signatura, tako da mi možemo saznati **naslov, autora** i ostale pojedinosti o knjizi iz tblKnjige na temelju **singnature** iz tblPosudbe. Za to je zaslužna relacija, a najčešći tip relacije je tip 1.



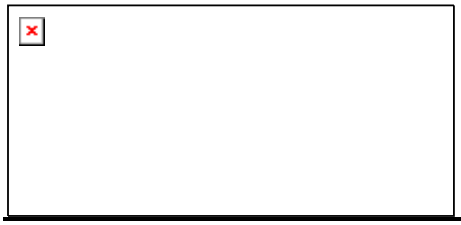
Tip 1: Uzmi u obzir samo one zapise za koje je relacijsko polje navedeno u obje tablice. To znači da će iz relacije biti razmatrani samo oni zapisi kod kojih je npr. signatura knjige navedena u obje tablice (tblKnjige i tblPosudbe). Ako je u tblPosudba navedena signatura knjige koja ne postoji u tblKnjige, taj će zapis biti ignoriran. Referencijalni integritet, sjetimo se, spriječava nastanak ovakvih situacija. Ova vrsta veze (Join-a) se u SQL treminologiji zove **INNER JOIN.**



Tip 2: Uzmi u obzir sve zapise iz prve tablice te one za koje je relacijsko polje navedeno i u drugoj tablici. To znači da će iz relacije biti razmatrani oni zapisi kod kojih je npr. signatura knjige navedena u obje tablice (tblKnjige i tblPosudbe) kao i svi ostali zapisi iz tblKnjige bez obzira na relaciju i na to da li ta signatura postoji u tblPosudbe. Ova vrsta veze (Join-a) se u SQL treminologiji zove **LEFT JOIN**



Tip 3: **Obrnuta situacija što se tiče tablica od tipa 2. Ova vrsta veze (Join-a) se u SQL treminologiji zove** RIGHT JOIN



Queries – kreiranje upita

Upiti su objekt baze podataka koji nam omogućava prikaz željenih podataka iz baze. Tradicionalno se u relacijskim bazama upiti postavljaju **SQL jezikom** (**Structured Query Language**) , što je moguće i u Access-u, ali Access ima i zgodnu mogućnost kreiranja upita uz pomoć grafičkog sučelja – **QBE mreže** (**Query By Example Grid**), što je praktičnije i jednostavnije za korištenje pa ćemo se mi ovdje baviti samo ovim načinom kreiranja upita. Upite ćemo pomoću QBE mreže kreirati tako da kliknemo gumb *New* u kartici *Query* u *Database* prozoru. Prvo je potrebno u *Show Table* dijalogu odabrati za prikaz sve tablice čiji će nam podaci trebati za dohvat željenih podataka. Nakon toga jednostavno spuštamo mišem (drag'n'drop) željena polja iz prikazanih tablica na QBE rešetku i postavljamo željene kriterije (Criteria) za odabir podataka iz tablica.

Npr. kreirat ćemo upit koji će nam prikazivati pregled prodaje namještaja na način da za svaku ostvarenu prodaju prikaže šifru, vrstu i model prodanog namještaja, te datum prodaje i broj prodanih komada namještaja. Rješenje možete vidjeti na slici 1.

Slika: *Select Query upit koji prikazuje pregled prodaje*

Nakon što kreiramo upit, da bismo ga pokrenuli i prikazali željene podatke, izvršavamo naredbu **Query → Run**. Upit snimamo naredbom **File → Save As** pod željenim imenom. Preporuka je svakom query-u dodati prefix qry.
Tako npr. query Prodaja zvati će se **qryProdaja.**

Značenja područja na QBE mreži su sljedeća:

- **Field** – ime odabranog polja
- **Table** – ime tablice u kojoj se nalazi odabrano polje
- **Sort** – definiramo hoćemo li i kako (uzlazno, silazno) sortirati rezultate upita prema vrijednostima određenog polja
- **Show** – da li će dotično polje biti prikazano u rezultantnoj tablici

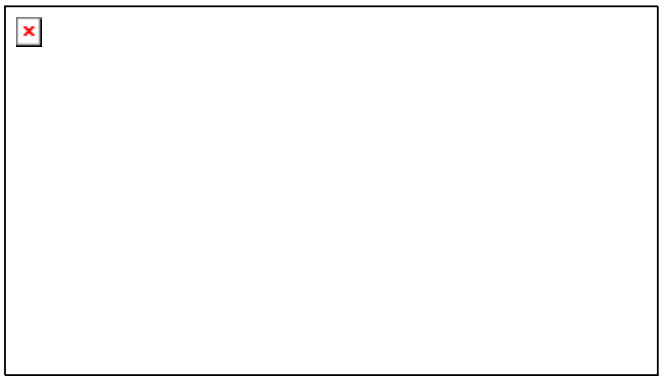
- **Criteria** – definiramo uvjete na podatke određenog polja da uđu u rezultatni skup podataka
- **Or** – drugi dio kriterija koji omogućuje da ako je je prvi **ili** drugi dio kriterija zadovoljen, da zapis bude uzet u obzir.

Field Properties

Svakom je polju query-a moguće odrediti format podatka. U kontekstualnom meniju polja obaberemo opciju **Properties** i zatim odaberemo željeni format u Field Properties prozoru:

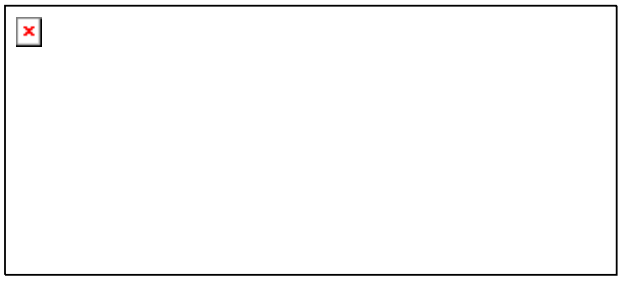
Lookup fields – polja za obabir predefiniranih vrijednosti

Lookup svojstvo polja (kako u queryu, tako i u tablicama) nam olakšavaju unos podatka u tablicu i to na taj način da nam se na mjestu unosa pojavi Drop-down lista (**Combo Box**) iz kojeg mi odaberemo željenu vrijednost za to polje. Podaci s kojima je popunjen taj ComboBox mogu se uzeti iz postojeće tablice ili query-a (**Row Source Type**) tako da odaberemo ime tablice ili query-a (**Row Source**). Zatim odaberemo redni broj polja (**Bound Column**) iz tog izvora podataka (tablice ili querya) iz kojeg uzimamo podatak koji će se nakon odabira upisati u ciljano polje (preslikati). Najčešće je to prvo polje odn. ID ili Šifra proizvoda itd. Row Source-a. **Column Count** određuje koliko će se polja u combu prikazati (vidi sliku 1. - odabrano je 2 polja, s tim da se podatak uzima samo iz prvog !) , a **Column Heads** da li će se ispisati nazivi polja (vidi sliku 2.). **List Rows** određuje koliko će se redaka u combo boxu pojaviti odjednom (za ostale moramo scrollati).



Slika 1.

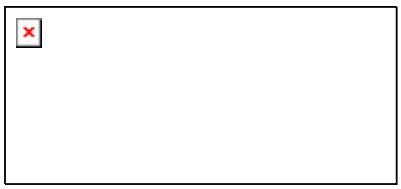
Prilikom unosa podataka u tablicu ili query (a kasnije i na formi), to će izgledati ovako:



Slika 2.

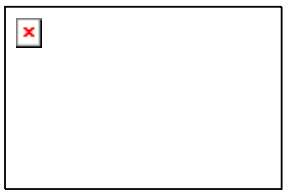
Criteria – AND i OR operatori

a) Želimo li više kriterija povezati AND operatorom to ćemo napravit tako da navedemo kriteriji u istom Criteria retku:



b) Želimo li više kriterija povezati OR opertatorom to ćemo napraviti tako da kriterije navedemo kaskadno:

c) Želimo li za jedno polje postaviti više mogućih kriterija s tim da nam je dovoljno da jedan od njih bude ispunjen tada ćemo kriterije navesti u istom stupcu:



d) Kombinacijom obojeg možemo dobiti AND i OR veze između kriterija.

Join Properties

Podvrstu relacije u queryu (Join Type) moguće je definirati baš kao i u Relationship prozoru. Dovoljno je u kontekstualnom meniju odabrati **Edit Relationship ...**

Calculated Fields - Izračunata polja

Stupci prikazani u rezultatima upita ne moraju uvijek biti postojeća polja iz tablica ili query-a baze podataka. Moguće je u upitu definirati i **izračunata polja (Calculated Fields).** To su nova polja, generirana polja na temelju određene formule koju zadajemo. To su funkcijska polja.

Tako npr. ako imamo polje **Jedinič_cijena** iz tablice **Skladište** i polje **Količina** iz tablice **Stavke** možemo u upitu kreirati polje novo izračunato polje **Ukupno** formulom:

Ukupno: Skladište!Jedinič_cijena*Stavke!Količina

Nakon toga možemo polju odrediti format **Currency** ili bez toga, pomoću konverzijske funkcije **CCur -** convert to currency - cijeli izraz prekonvertirati u valutu:

Ukupno: CCur(Skladište!Jedinič_cijena * Stavke!Količina)

gdje je **Ukupno** ime novog izračunatog polja, a nakon dvotočke '**:**' pišemo ***formulu***. Ovdje se očito radi o jednostavnom umnošku cijene i broja prodanih komada što daje ukupan prihod.

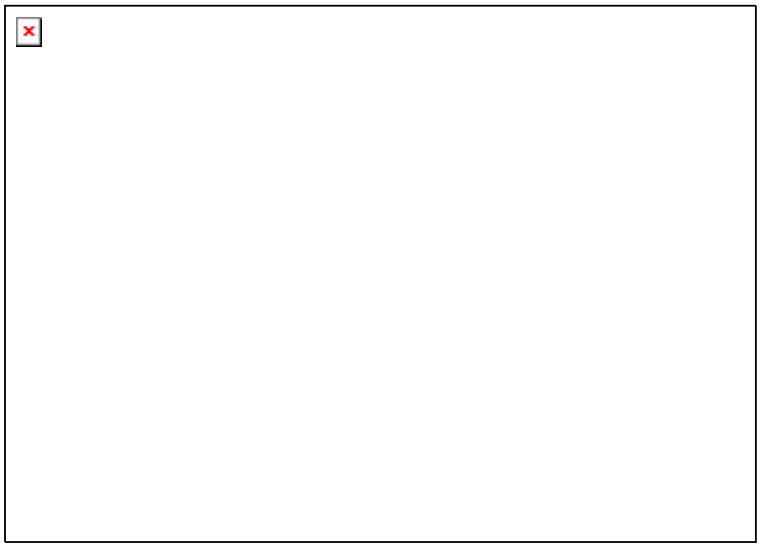
To će omogućiti da se u svakom retku query-a prikaže umnožak jednične cijene i broj prodanih komada (iz odgovarajućeg retka).

Primjetite da je, ako se u izračunatom polju nalaze podaci iz više različitih tablica, potrebno osim imena polja navesti i ime tablice u kojoj se polje nalazi i to u slijedećem sintaksnom obliku:

ime tablice!ime polja

To se zove referenciranje. Referenciranje nije potrebno provoditi ako je query baziran na jednoj tablici, no ako imamo više tablica referenciranje je neophodno !

Za kreiranje formula (izraza) pri definiranju izračunatih polja može koristitimo ***Expression Builder*.** Pokrećemo ga sa toolbara ili tako da u kontekst. meniju obaberemo **Build ...**



U Builderu možemo birati postojeće tablice ili query-e i pripadajuća polja, formu, report, ugrađene funkcije (ovo je potrebno proučiti, ima ih mnogo), te operatore (And, Or, Not, Like), simbole (zagrade, plus, minus, puta ...) – i sve to s ciljem da uz što manje tipkanja kreiramo formulu za izračunato polje.

Totali – agregatne funckije

Podatke je moguće i grupirati te prikazati sumarne vrijednosti za svaku od grupa podataka. Za grupiranje podataka u upitu potrebno je kliknuti gumb sa sličicom slova Σ (Totals) ili **View/Totals**. Pri tome se na QBE rešetci pojavljuje još jedno polje **Total** u kojem za svako polje određujemo da li će se podaci u tom polju samo grupirati (Group By) ili će se na njih primjeniti neka **agregatna funkcija** (Sum, Max, Min, Count, Average …).

Npr. Kreirati ćemo upit koji prikazuje pregled prihoda ostvarenog prodajom pojedinih modela namještaja po mjesecima. Rješenje je dano na slici 1. Uočite da su polja Mjesec i Prihod izračunata polja. Podaci se također grupiraju prema mjesecu, šifri, modelu i vrsti namještaja, te po datumu prodaje. Podaci su sortirani po datumu prodaje.

Query za pregled ostvarenog prihoda za svaki model namještaja po mjesecima:

Rezultati dobiveni pokretanjem ovog upita izgledaju kao na slici:

SQL
(Structured Query Language)

Select - *prikaz polja iz tablice*
From - *iz koje tablice*
Order by - *sortiraj po (prezime ili ime ili….)*
Order by Desc - *sortiraj od z-a*
Select * - *selectiraj sva polja (kada koristimo asp, * mijenjamo sa %*)

Ubacivanje uvjeta :

Where polje **Like 'M*'**
Order by Prezime

Primjer :
Zamislimo si da imamo tablicu koja se zove tblAdresar I da ona sadrži polja Ime, Prezime, Telefon, Adresa I želimo ispisati sve osobe koje se zovu Mario I da ih sortira od z-a I da izlista sva polja traženih osoba:

SELECT * FROM tblAdresar
WHERE Ime Like 'Mario'
ORDER BY DESC Ime

Višestruko sortiranje : **ORDER BY** polje1 **ASC,** polje2 **DESC**

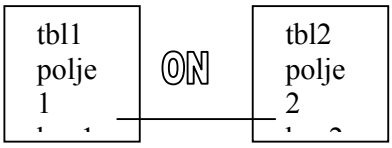
JOIN
-naredbe koje vade podatke iz više tablica koje su u relaciji

Vrste JOIN-ova:

- **INNER JOIN**
- **OUTER JOIN - LEFT JOIN**
  - RIGHT JOIN

INNER JOIN

-*koristi se samo ako tabele imaju iste primarne ključeve*
SELECT tbl1.polje1, tbl2.polje2
FROM tbl1 **INNER JOIN** tbl2
ON tbl1.key1=tbl2.key2
WHERE……



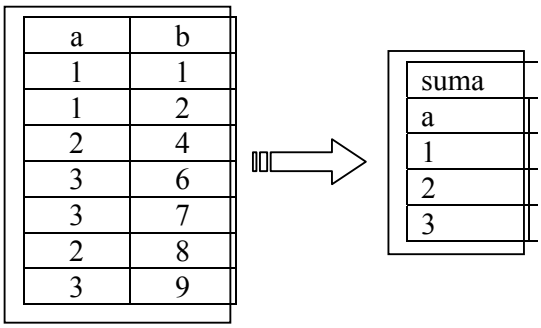
INNER JOIN – združena tabela, tj. Relacija između tabela
ON - po kojem ključu, tj. Određivanje ključa

Agregatne funkcije

-specifične (T-SQL)- Microsoft Transaction SQL

GROUP BY - grupira polja , sve iste brojeve pretvara u istu grupu
AS - pridruživanje imena polja rezultatu
SUM(polje) - suma pojedine grupe
MIN(polje**)** - minimalni broj u grupi
MAX(polje**)** - maximalni broj u grupi
AVG(polje**)** - prosjek pojedine grupe
COUNT(polje**)** - prebrojava koliko polja ima jedna grupa

Primjer:



a	b
1	1
1	2
2	4
3	6
3	7
2	8
3	9

suma
a
1
2
3

SELECT a,**SUM**(b) **AS** Suma **FROM** tbl
GROUP BY a

INSERT

-ubacivanje podataka u tablicu

INSERT INTO *ime_tablice*

INSERT INTO tblKupnja(broj_artikla,cijena)
SELECT broj_artikla,cijena **FROM** tblKošarica
WHERE kupac=kupacID

-iz tablice košarica prebacuje u tablicu kupnja

DELETE

-brisanje iz tablice

DELETE FROM *tablica*

Primjer . **DELETE FROM** tblKošarica **WHERE** kupac=kupacID

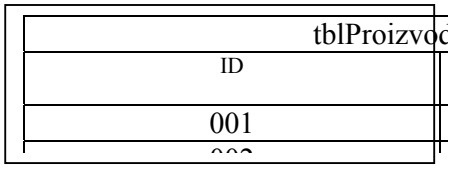
UPDATE

-omogućuje da jednom intervancijom promjenimo više vrijednosti

UPDATE tablica **SET** polje1=vrijednost/formula, polje2=vrijednost/formula

npr.
-u kategoriji 1 cijena mora pojeftiniti za 5%

UPDATE tblProizvodi **SET** cijena=cijena*0,05 **WHERE** kategorija=1



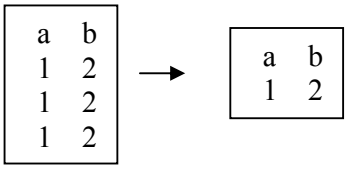
Primjer 2 . – Kako smanijiti količinu

UPDATE tblProizvodi **SET** količina=količina-1 **WHERE** ID=001

DISTINCT

-služi za ispisivanj samo jednog (npr. U danu je jedan proizvod kupljen 10 puta. DISTINCT upisujemo, da nam taj proizvod ne bi ispisalo 10 puta, nego samo jednom.)

SELECT DISTINCT polja **FROM** tablica



Primjeri :

Naziv	Adresa	Telefon	Dobavlač ID
Pliva d.o.o.	Markuševac bb	01/258-698	1
Zeljezara	Sisačka 38	01/256-987	2

Naziv	Cijena	Kategorija	Proizvod ID	Dobavljač ID
Cijev	48,55	Vodovod	3	1
Grickalice	22,33	Kučanstvo	4	2
vbxcbvxcvb	88	vbv	5	2

- združena tabela sa zajedničkim ključem

SELECT tblDobavljaci.Naziv, tblProizvodi.Naziv, tblProizvodi.Cijena
FROM tblDobavljaci INNER JOIN tblProizvodi ON tblDobavljaci.Dobavlač ID = tblProizvodi.Dobavljač ID;

Rezultat:

tblDobavljaci.Naziv	tblProizvodi.Naziv	Cijena
Pliva d.o.o.	Cijev	48,55
Zeljezara	Grickalice	22,33
Zeljezara	vbxcbvxcvb	88

Primjer2 :

ArtID	Naziv	Kategorija	Cijena	Stanje
1	Kljuc 18	Kljucevi	28,00 kn	45
2	Kruh	Hrana	4,50 kn	450
3	Mala bilježnica	Bilježnice	3,50 kn	550
4	Velika bilježnica	Bilježnice	5,50 kn	600

ID	Ime	Prezime	IPAdresa
1	Pero	Kvrgić	128
2	Jura	Perić	129
3	Stevo	Jagić	130
4	Zvone	Trbić	155

ID	ArtID	QTY	Expire_Date	IDtable
1	1	5	9.11.2001	1
1	2	3	9.11.2001	2
1	3	4	9.11.2001	3
1	4	3	9.11.2001	4

- ukupna potrošnja svakog kupca sa ispisom njegove IP adrese:
- -primjer sa tri združene tabele :
-

sve_ukupno	IPAdresa
193,00 kn	128
97,50 kn	129
66,00 kn	130
37,50 kn	155

```
SELECT Sum((tblArtikli.Cijena*tblKosarica.QTY)) AS sve_ukupno, tblKupci.IPAdresa
FROM tblArtikli INNER JOIN (tblKupci INNER JOIN tblKosarica ON tblKupci.ID=tblKosarica.ID) ON
tblArtikli.ArtID=tblKosarica.ArtID
GROUP BY tblKupci.IPAdresa;
```

- ukupno po datumu sa upitom za IP adresu (128), dobije se rezultat

Ukupno_po_datumu
193,00 kn

```
SELECT Sum(queKošara.Ukupno) AS Ukupno_po_datumu
FROM queKošara
GROUP BY queKošara.IPAdresa;
```

-web košarica – koliko je kupac sa IP adresom128 proizvoda kupio

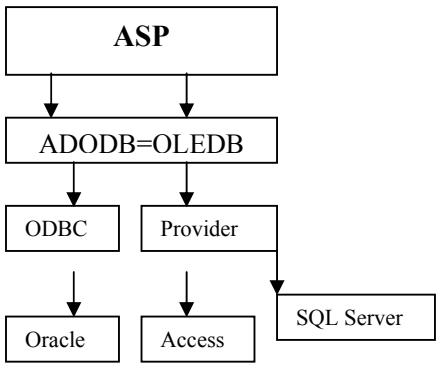
Ime	Prezime	IPAdresa	Naziv	Cijena	QTY	Ukupno
Pero	Kvrgić	128	Kruh	4,50 kn	2	9
Pero	Kvrgić	128	Velika bilježnica	5,50 kn	3	16,5
Pero	Kvrgić	128	Mala bilježnica	3,50 kn	4	14
Pero	Kvrgić	128	Kruh	4,50 kn	3	13,5
Pero	Kvrgić	128	Kljuc 18	28,00 kn	5	140

```
SELECT tblKupci.Ime, tblKupci.Prezime, tblKupci.IPAdresa, tblArtikli.Naziv, tblArtikli.Cijena,
tblKosarica.QTY, (tblArtikli.Cijena*tblKosarica.QTY) AS Ukupno
FROM tblArtikli INNER JOIN (tblKupci INNER JOIN tblKosarica ON tblKupci.ID=tblKosarica.ID) ON
tblArtikli.ArtID=tblKosarica.ArtID
WHERE (((tblKupci.IPAdresa)=[Unesite IP adresu]))
ORDER BY tblKupci.Ime;
```

5. Cjelina Asp i baze podataka

ADO

Za spajanje na bazu podataka, ASP koristi ***ActiveX Data Objects***, ili, kraće, ***ADO***. ADO je biblioteka gotovih objekata koja dolazi zajedno s ASP-om i omogućava spajanje na bilo koju bazu podataka koja podržava ODBC standard. Većina modernih baza podataka podržava taj standard, kao što su npr. MS SQL Server, MS Access, Informix, Oracle ili FoxPro. ADO objekti čiju uporabu ćemo podrobno opisati su ***Connection*** i ***Recordset***.



Kreiranje Objekta:

Connection -- priključivanje na bazu podataka

Za ostvarivanje veze s bazom podataka potrebna su nam tri koraka:

- Kreiramo ADO objekt koji će se spajati na bazu podataka
- Dostavimo tom objektu informacije o lokaciji i tipu naše baze podataka
- Naredimo objektu da otvori vezu s bazom

npr:

```
SET  konekcija = Server.CreateObject("ADODB.Connection")
konekcija.Provider = "Microsoft.Jet.OLEDB.4.0"        - koji provajder
na koju bazu
konekcija.ConnectionString = "Data Source=" & Server.MapPath("..") & "/baza/mojabaza.mdb"
konekcija.Mode = 3      -- 1 – Read , 2 – Write, 3 – Read and write
konekcija.Open          - otvaranje konekcije
```

Recordset -- otvaranje sadržaja baze podataka

Za otvaranje sadržaja baze podataka potrebna su nam dva koraka :

- Kreiramo ADO object koji će ispisivati sadržaj
- Otvorimo konekciju -- object.Open "SQL", object otvaranja baze,kursor,ključ

Kursori

Za naš recordset objekt možemo odabrati jednu od četiri vrste kursora kojim ćemo prolaziti kroz redove s podacima. To su *Forward-Only*, *Static*, *Dynamic* i *Keyset*. Po defaultu, kada kreiramo recordset objekt, on ima Forward-Only kursor. Njime se možemo kretati samo prema naprijed, tj. omogućene su nam funkcije `MoveNext`, `MoveLast` i `MoveFirst`, dok `MovePrevious` i `Move x`, ako je x negativan broj, daju grešku. Iako je Forward-Only default kursor, možemo ga zadati i eksplicitno:

```
rs.CursorType = adOpenForwardOnly
```

`adOpenForwardOnly` je konstanta čija vrijednost je definirana u datoteci *adovbs.inc* koju je korisno uključiti pomoću include direktive u svaku našu stranicu na kojoj koristimo ADO objekte. Ta datoteka sadrži sve potrebne konstante za ADO objekte, dolazi sa ASP-om i naći ćemo je nakon instalacije u Windows\System direktoriju.
Kursor tipa Static dozvoljava nam putovanje u svim pravcima, kao i još neke funkcije, npr. RecordCount, koja nam daje broj zapisa u recordsetu. Zadajemo ga ovako:

```
rs.CursorType = adOpenStatic
```

Kursori tipa Keyset i Dynamic nam omogućuju da dodajemo, mijenjamo ili brišemo podatke pomoću AddNew, Update i Delete method-a i da odmah vidimo promjene koje su u međuvremenu nastale u bazi ali, iako su vrlo korisni u Visual Basic-u, nisu zgodni za korištenje na web-u jer je nemoguće imati ažurne podatke u browseru bez stalnog osvježavanja stranice. Oni isto tako koriste puno više resursa nego Static i Forward-Only kursori. Zato ih, u principu, ne treba koristiti, a upisivanje, brisanje i izmjenu podataka treba vršiti pomoću SQL naredbi, što ćemo prikazati u slijedećem odjeljku.

Ako ne koristimo **adovbs.inc** onda :

1 – forward only (0)
2 – keyset (1)
3 – dynamic (2)
4 – static (3)

Ključevi (lokoti)

- kako promjeniti podatke u bazi u odnosu na ostale klijente koji se priključe na bazu

1 – read only - samo za čitanje baze
2 – pessimistic - kada se priključimo, zaključamo bazu za ostale
3 – optimistic – kada pozovemo UPDATE , record se zaključava za ostale klijente –najbolja primjena
4 – batch optimistic – konektamo se na bazu, diskonektamo, promjenimo podatke, konektamo se I izvršimo masovnu projenu

Npr :

```
SET citaj = Server.CreateObject( "ADODB.Recordset")
Citaj.Open  "SELECT * FROM tblMoja" , konekcija,1,1
```

Pregled sadržaja podataka možemo napraviti I bez kreiranja Recordset objekta. To možemo pomoću naredbe **.Execute .**

Npr.

```
SET  konekcija = Server.CreateObject(“ADODB.Connection”)
konekcija.Provider = “Microsoft.Jet.OLEDB.4.0”
konekcija.ConnectionString = “Data Source=“ & Server.MapPath(“..”) & “/baza/mojabaza.mdb”
konekcija.Mode = 3
konekcija.Open
SET citaj = konekcija.Execute “SELECT …….
```

Recordset funkcije :

citaj.ReadFile ---- ispis iz tablice

primjer – ispis traženog pojma po ključnim riječima

```
set baza=server.createobject("ADODB.Connection")
set ispis= server.createobject("ADODB.Recordset")

baza.Provider="Microsoft.Jet.OLEDB.4.0"
baza.ConnectionString="Data source=" & server.mappath(".") &"/baza/" & "Posloprimac.mdb"
baza.mode=3
baza.open

  a=request.form("kategorija")
  b=request.form("rijeci")

  ispis.open "select Zanimanje  from " & a & " where Kljucne_rijeci like '%" &b&"%' or Kljucne_rijeci like '" &b&"%' or
Kljucne_rijeci like '%" & b & "'",baza,0,3

    while not ispis.eof
  for each polje in ispis.fields
  response.write "<tr>"
   sad=ispis(polje.name)
if isnull(sad) then sad= " "
response.write "<td ><font size=2 color=black> " & sad  &"<hr></td>"
next
response.write "</tr>"
ispis.movenext
wend

   set ispis=nothing
   set baza=nothing
```

upis u tablicu :

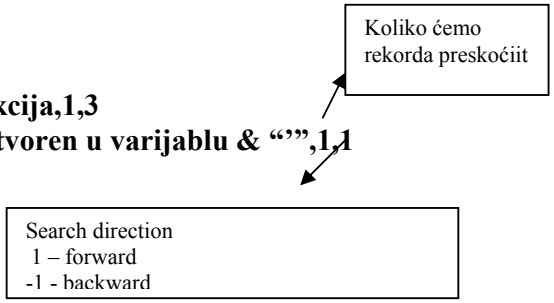
```
citaj.AddNew
citaj(ime_polja1) = podatak1
citaj(ime_polja2) = podatak2
citaj(ime_poljaN) = podatakN
citaj.Update
```

citaj.Delete --- brisanje iz tablice

if not citaj.EOF then
citaj.Delete
end if

---

pronalaženje rekorda unutar otvorenog recordseta



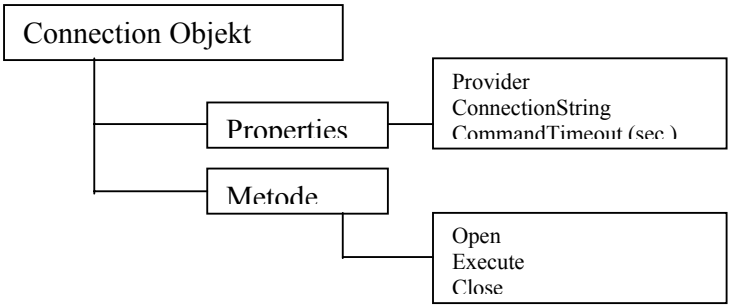
citaj.Open "SELECT * FROM tblMoja",konekcija,1,3
citaj.Find "[Naslov] = ' " & traženi pojam pretvoren u varijablu & """,1,1

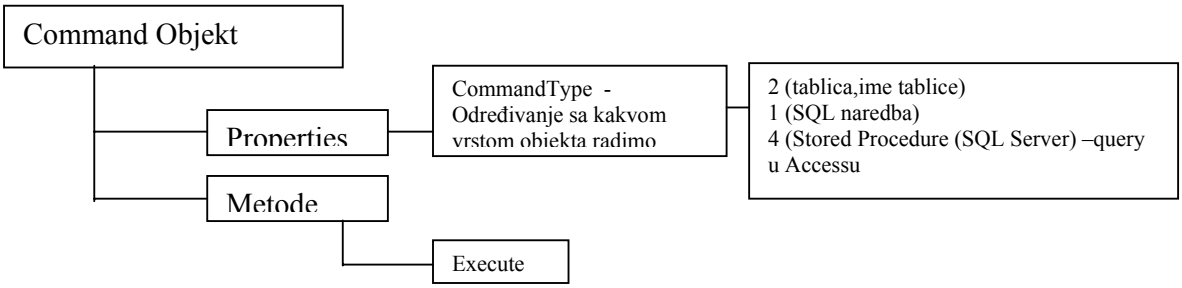
-ova funkcija je ograničena na samo jedno polje
- da bismo našli više polja složimo ga ovako :

SELECT *,(Ime &' ' &Prezime) as Osoba from tblMoja
Citaj.Find "[Osoba] =""& varijabla od imena& " "& varijabla od prezimena & """

If not citaj.eof then
…..
end if

Specifikacije ADO 2.6 Objekta





SET cmd = Server.CreateObject(“ADODB.Command”)

Drugi način povezivanja sa bazom podataka :

Za ostvarivanje veze s bazom podataka potrebna su nam tri koraka:

- Kreiramo ADO objekt koji će se spajati na bazu podataka
- Dostavimo tom objektu informacije o lokaciji i tipu naše baze podataka
- Naredimo objektu da otvori vezu s bazom

Za prvi korak, iskoristit ćemo ADO objekt koji je zadužen za povezivanje s bazama: *Connection objekt*. Za početak, moramo ga kreirati. To činimo pomoću `Server.CreateObject` methoda:

```
Dim c
Set c = Server.CreateObject("ADODB.Connection")
```

Objekt, kojeg smo nazvali c, je sada kreiran. Sada mu moramo reći potrebne informacije za povezivanje s bazom. Imamo dva načina za to. Prvi je kroz korištenje sistemskog DSN-a (*Data Source Name*). Sistemski DSN je datoteka koja sadrži informacije o pojedinoj bazi podataka. U tim informacijama se nalaze fizička lokacija baze na našem kompjuteru, tip baze (MS SQL, Access …) i još neke sistemske informacije.
Sistemski DSN u Windowsima možemo kreirati tako da u Control Panelu otvorimo applet ODBC. Na njemu se nalazi System DSN tab. Ovdje možemo kreirati sistemski DSN koji želimo (Oracle, Informix, MS SQL, Access …) i dati mu ime po našoj želji.
Nakon što smo kreirali sistemski DSN, moramo našem connection objektu c reći ime baze podataka na koju se mora spojiti. To ćemo učiniti pomoću slijedeće linije:

```
c.ConnectionString = "DSN = Ime sistemskog DSN-a"
```

Pri tom za ime stavljamo isto ono ime koje smo dali našem sistemskom DSN-u u Control Panelu.
Drugi način za ostvarivanje veze s bazom podataka je tzv. *DSN-less* veza. Kod korištenja ovog načina, moramo sve podatke koje inače sadrži sistemski DSN posložiti u naš connection string. Ovakav pristup se obično koristi kod MS Access baza podataka. Evo primjera DSN-less connection stringa:

```
c.ConnectionString = "DBQ = C:\Webroot\Jackbase.mdb;
DRIVER = {Microsoft Access driver (*.mdb)}"
```

`DBQ` govori connection objektu fizičku lokaciju baze, a `DRIVER` tip drivera koji će znati pričati s našom bazom.
Sada, sve što trebamo učiniti je otvoriti vezu s bazom. To ćemo učiniti pomoću Open method-a connection objekta:

```
c.Open
```

Radi kraćeg pisanja, drugi i treći korak se, ako koristimo sistemski DSN, mogu spojiti u jedan:

```
c.Open "Ime sistemskog DSN-a"
```

Sada je naša baza spremna za prihvaćanje SQL naredbi.

Recordset objekt

Ovaj objekt će sadržavati podatke koje pročitamo iz baze podataka, uredno složene u redove. Uporabu recordset objekta najlakše ćemo prikazati kroz jednostavan primjer.
Recimo da imamo jednostavnu bazu podataka i u njoj tablicu *Cjenik*, koja sadrži kolone ID, Ime i Cijena, i da želimo napraviti ASP stranicu koja će otvoriti bazu, pročitati sve proizvode jeftinije od 100 kuna i uredno ih ispisati u HTML stranicu.
Najjednostavniji i najsigurniji način pristupa bazi podataka je pomoću slijedeća dva koraka:

1. Spojimo se na bazu pomoću `ADODB.Connection` objekta
2. Pričamo s bazom pomoću `ADODB.Recordset` objekta

Postoji i način spajanja pri kojem preskačemo kreiranje jednog od ova dva objekta, koji ćemo opisati kasnije. Za sada, ići ćemo korak po korak:
Kreirat ćemo `ADODB.Connection` objekt c i spojiti ga na našu bazu *Jackbase*, za koju smo već ranije kreirali sistemski DSN:

```
Dim c
Set c = Server.CreateObject("ADODB.Connection")
c.Open "Jackbase"
```

Sada ćemo kreirati `ADODB.Recordset` objekt r pomoću kojega ćemo pribaviti podatke iz baze koju smo upravo otvorili:

```
Dim r
Set r = Server.CreateObject("ADODB.Recordset")
```

Nadalje, moramo kreirati SQL naredbu koja će iz baze pročitati točno one podatke koji nam trebaju. Pametno je SQL naredbe spremati u stringove, kako bismo ih mogli koristiti kad god nam zatrebaju:

```
Dim s
s = "SELECT Ime, Cijena FROM Cjenik WHERE Cijena < 100"
```

Primijetimo da, za razliku od c i r, ispred s ne stavljamo naredbu `Set` pri dodjeljivanju vrijednosti. To je zbog toga što su c i r objekti, a s običan string. Sada ćemo konačno iz baze pročitati naše podatke:

```
r.Open s, c
```

To je sve. Objekt r sada sadrži sve podatke koje smo tražili. Osim pomoću `Open` method-a, ovu istu naredbu možemo izvršiti i drugačije:

```
Set r = c.Execute(s)
```

U ovom slučaju nismo trebali prije deklarirati r kao Recordset, jer ga ova naredba automatski kreira. Isto tako, mogli smo preskočiti deklariranje objekta c i direktno iz Recordset objekta otvoriti bazu:

```
r.Open s, "Jackbase"
```

Potpuno je svejedno na koji način smo pročitali podatke, no poželjno je pridržavati se nekih osnovnih opće prihvaćenih konvencija programiranja ASP stranica, koje preferiraju naš prvi način – korak po korak.
Naš recordset objekt r sada izgleda kao dvodimenzionalna matrica. Ako nam je tablica *Cjenik* sadržavala slijedeće podatke:

ID	Ime	Cijena
1	Novčanik Samsonite	88
2	Olovka Parker	30
3	Sat Casio	50
4	Microsoft Miš	150
5	Grim Fandango 2CD	300
6	Stolac IKEA	750
7	Torba Benetton	90
8	Telefon Philips	100

tada r izgleda ovako:

Ime	Cijena
Novčanik Samsonite	88
Olovka Parker	30
Sat Casio	50
Torba Benetton	90

Kada izvršimo Open method kursor našeg recordseta pokazuje na prvi red rezultirajuće tablice. Dakle, sad naš r pokazuje na red koji sadrži "Novčanik Samsonite" i 88. Da bismo pristupili tim podacima sve što trebamo je navesti ime kolone koju želimo:

```
r("Ime")
```

Ovo će nam vratiti vrijednost kolone Ime u redu na kojeg trenutno pokazuje kursor. Sada to možemo smatrati za običnu varijablu i tako koristiti:

```
Response.Write r("Ime")
```

Ovo će ispisati "Novčanik Samsonite". Sada bismo željeli pomicati kursor po našem recordsetu. To činimo pomoću ovih pet methoda: *MoveNext*, *MovePrevious*, *MoveFirst*, *MoveLast* i *Move x*. Neki od njih nam sada nisu dostupni, iz razloga koje ćemo opisati kasnije. Ako želimo prijeći u novi red, to činimo pomoću `MoveNext` method-a:

```
r.MoveNext
```

Objekt r sada pokazuje na red koji sadrži "Olovka Parker" i 30. Kada prođemo zadnji red, property `EOF` postaje true, a r više ne pokazuje nigdje, tj. `r("Ime")` će javiti grešku. Isto tako, ako prođemo pomoću `MovePrevious` ispred prvog reda, onda property `BOF` postaje true. Sada možemo ispisati cijeli recordset u HTML tablicu:

```
<TABLE>
<% Do Until r.EOF = True %>
<TR>
<%
   Response.Write "<TD>" & r("Ime") & "</TD>"
   Response.Write "<TD>" & r("Cijena") & "</TD>" %>
</TR>
'Idemo u slijedeći red (važno!!)
<% r.MoveNext
   Loop %>
</TABLE>
```

Za kraj, važno je uvijek eksplicitno zatvoriti i obrisati sve connection i recordset objekte koje smo koristili, kako ne bi zauzimali memoriju na serveru:

```
'Zatvaramo Recordset objekt
r.Close

'Brišemo Recordset Objekt
Set r = Nothing

'Zatvaramo Connection objekt
c.Close

'Brišemo Connection Objekt
Set c = Nothing
```

Recordset objekt ima još mnogo mogućnosti od kojih ćemo nabrojati samo neke. Na primjer, imena i broj kolona u recordset objektu možemo dobiti pomoću kolekcije Fields. Broj kolona će nam dati slijedeća naredba:

```
Response.Write r.Fields.Count
```

Dok ćemo imena svih kolona u recordsetu doznati pomoću slijedeće skripte:

```
<%
For i = 0 to r.Fields.Count - 1
        Response.Write r.Fields(i).Name
Next
%>
```

Kao što vidimo, imena kolona su, kao što je to i inače slučaj s kolekcijama varijabli, pohranjena u niz koji počinje s indeksom 0. Isto tako možemo pristupiti i sadržaju reda na koji nam kursor trenutno pokazuje:

```
<%
For i = 0 to r.Fields.Count - 1
        Response.Write r(i)
Next
%>
```

Zapisivanje podataka u bazu

Kao što smo rekli, najbolji način za zapisivanje podataka u bazu je direktno pomoću SQL naredbi. Recimo da smo pomoću FORM-a na prethodnoj stranici unijeli podatke: ime, prezime, telefon, e-mail i komentar, i da ih želimo upisati u tablicu *Komentari* u našoj bazi podataka. Kao prvo, ta tablica mora biti kreirana u bazi i to tako da sadrži još jednu kolonu u kojoj se nalazi glavni ključ kojeg baza automatski generira. Kada ne bi imali tu kolonu, morali bismo u našu stranicu uključiti i kod koji generira jedinstveni ključ. Jednostavnije je prepustiti taj posao bazi. Slijedeće, moramo pročitati varijable koje nam je poslala prethodna stranica:

```
ime = Request.Form("ime")
prezime = Request.Form("prezime")
tel = Request.Form("tel")
mail = Request.Form("mail")
komentar = Request.Form("komentar")
```

Sada iz tih podataka kreiramo SQL string koji će te podatke upisati u bazu:

```
s = "INSERT INTO Komentari([Ime],[Prezime],[Telefon]," & _
    "[E mail],[Komentar]) VALUES " &  "('" & ime & _
    "','" & prezime & "','" & tel & "','" & mail & _
    "','" & komentar & "')"
```

Pri izradi SQL upita moramo paziti na više stvari. Imena kolona moramo stavljati u uglate zagrade, a stringove koje upisujemo moramo stavljati unutar jednostrukih navodnika. Stringove je također poželjno prije kreiranja SQL stringa provjeriti na korisnikovom računalu da slučajno ne sadrže neke nepoželjne karaktere. To činimo pomoću javascripta u `OnSubmit` event-u našeg formulara ili, ako želimo samo jednostruke navodnike zamijeniti sa duplim jednostrukim navodnicima, kako ne bi izazvali grešku u SQL-u, pomoću naredbe `replace`. Na primjer, da bi pripremili string `Komentar` za SQL, pišemo:

```
Komentar = Replace(Request.Form("Komentar"),"'","''")
```

SQL naredbu izvršavamo pomoću `Execute` methoda connection stringa:

```
c.Execute(s)
```

Connection objekt na kraju skripte koja upisuje u bazu moramo eksplicitno uništiti. Update baze vršimo na potpuno isti način. Na primjer, ako želimo umjesto starog komentara nekog korisnika upisati novi, pišemo:

```
s = "UPDATE Komentari SET Telefon = '" & tel & "'," & _
    "[E mail] = '" & mail & "', Komentar = '" & _
    komentar & "' WHERE Ime = '" & ime & "' AND " & _
    "Prezime = '" & Prezime & "';"

c.Execute(s)
```

Korisni savjeti

Pri korištenju baza podataka iz ASP-a moramo stalno imati na umu kako je naša aplikacija predviđena za web, tj. za više stotina istovremenih korisnika. Zato nam kod mora biti optimiziran kako bi što manje vremena provodili spojeni na bazu podataka koja mora odgovoriti na više stotina zahtjeva, i kako bi što prije oslobađali memoriju. U prethodnim odjeljcima objašnjena je štetnost prekomjerne uporabe session varijabli i važnost eksplicitnog zatvaranja i uništavanja svih objekata na kraju skripte. Još jedna greška koju ASP programeri često čine je krivo prebrajanje zapisa u nekoj tablici.

Prebrajanje zapisa u tablici

Da bismo prebrojali koliko ima zapisa u nekom recordsetu nikada ne valja prvo pročitati recordset pomoću Forward-Only kursora, pa se onda snalaziti pomoću petlje koja prelazi preko cijelog recordseta brojeći zapise ili sličnim nepraktičnim metodama. Takve metode troše vrijeme na serveru vršeći gomilu nepotrebnih operacija i pogotovo su nepoželjne ako naš recordset ima na tisuće zapisa. Prava načina su dva: ili deklarirati recordset sa Static kursorom koji, za razliku od Forward-Only kursora, podržava `RecordCount` method, ili iskoristiti `COUNT()` naredbu u SQL-u. Ispis svih tablica u Access bazi

Ako želimo iz ASP-a ispisati sve tablice koje imamo u Access bazi podataka koristimo `OpenSchema` method connection objekta. Naše tablice će imati `TABLE_TYPE = TABLE`, dok će ostale biti `SYSTEM TABLE`, `VIEW` i slično. Ovaj primjer ispisuje samo prave tablice iz baze:

```
<%
Set c = Server.CreateObject("ADODB.Connection")
c.Open "Jackbase"

Const adSchemaTables = 20

Set r = c.OpenSchema(adSchemaTables)
Do Until r.EOF
   if r("TABLE_TYPE")="TABLE" then
   Response.Write r("TABLE_NAME") & "<BR>"
   End If
r.MoveNext
Loop
%>
```

ODBC

Open Database Connection **ili** ***ODBC*** je Microsoftova implementacija specifikacije X/Open i SQL Access Group call level interface-a. Pomoću ODBC-a, naš program može čitati i pisati u bilo koju bazu podataka pomoću standardnih SQL naredbi. ODBC driveri prevode SQL u pravilne naredbe specifične za svaku pojedinu bazu. U slučaju SQL baza podataka kao što su to Microsoft SQL Server ili Oracle, nije potrebno puno prevođenja. U slučaju pristupa podacima unutar tekst datoteka, Microsoft Excel datoteka, ili drugih tipova pohranjenih podataka koji nisu predviđeni da rade sa SQL-om, driveri moraju biti prilično komplicirani. Ovakav pristup omogućava da bez obzira kako pohranjivali podatke, sve procedure koje im pristupaju pišemo na isti način, mijenjajući samo drivere.

6.Cjelina Zaštita podataka (enkripcija)

primjer:

```
set baza=server.createobject(“ADODB.Connection”)
set pogled=server.createobject(“ADODB.Recordset”)

        baza.mode=3
        baza.Provider=”Microsoft.Jet.OLEDB.4.0”
        baza.ConnectionString=”Data Source=” &server.mappath(“.”) &”/baza.mdb”
        baza.open

pogled.open “select * from tabela “,baza,1,3

    pogled.addnew
    pogled(“ime”)=protect(ime)
    pogled(“prezime”)=request.form(“prezime”)
    pogled.update

pogled.close
set baza=nothing

function protect(x)

 y=request.form(“ime”)
for i = 1 to len(y)
w=asc(mid(y,i,1))
w=w+i-6

v=w/2
q=v+w
r=rnd*255
z=z&chr(w)&chr(v)&chr(q)&chr(r)
next
protect=z
end function
```

Reversna funkcija

```
function deprotect(ime)

        y= request.form("ime")
    for i = 1 to len(y) STEP 4
     j=j+1
    w = asc(mid(y,i,1))
      w=w+i-6

      z=z&chr(w)
      NEXT
      deprotect=z

      end function
```