

Sadalītu datu apstrāde datoru tīklos

Dr. sc. ing., as. prof. *Pāvels Rusakovs*

Mg. sc. ing. *Andrejs Jeršovs*

Literatūras saraksts:

1. Russel Jones. Mastering TM Active Server Pages 3.
Sybex, Inc., 2000.

Рассел Джонс. Active Server Pages 3. Полное руководство.
Киев, "Век+", 2001.

2. Buyens Jim. Web Database Development. Step by Step.
Microsoft Press, 2000.

Байенс Джим. Разработка баз данных для Web. Шаг за шагом.
Москва, Microsoft Press, 2001.

3. Williams Al, Barber Kim, Newkirk Paul. Active Server Pages.
The Coriolis Group, 2000.

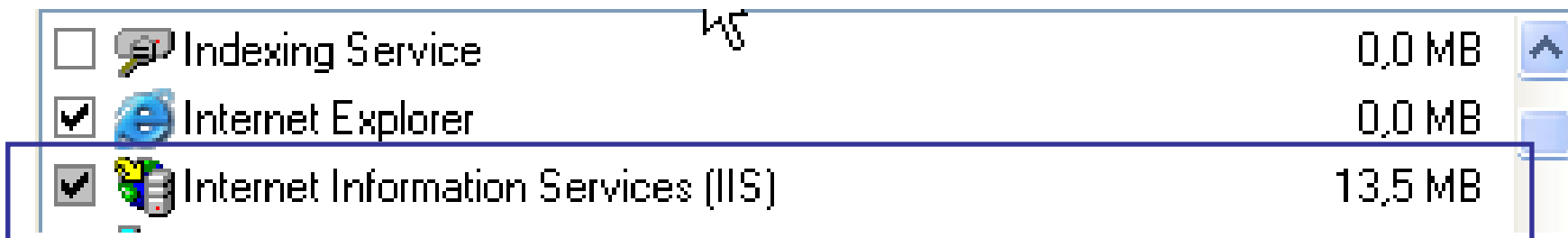
Уильямс Эл, Барбер Ким, Ньюкирк Пол. Active Server Pages.
Санкт-Петербург, "БНВ", 2001.

ASP programma ir *skripts*, kas izpildās uz servera.

Darbam ar ASP izmanto *Internet Information Services* (IIS).

IIS instalēšana:

1. Control Panel.
2. Add or Remove Programs.
3. Add/Remove Windows Components.



Lai ir disks ar mapi Windows (piemēram, **D:**)

Tiks izveidota mape **D: \Inetpub\wwwroot**.

Mapes iekšā var izveidot papildu mapi ASP skriptiem (piemēram, **MyHost**)

D: \Inetpub\wwwroot\MyHost.

Skripta adresēšana programmā

http: // localhost / myhost / my . asp

Elementārais ziņojums:

<h1>

<% Response.Write "Hello, World !" %>

</h1>

Rezultāts: *pirmā līmeņa* virsraksts.

Tīmekļa lappuses koda analīze (peles labā poga, *View Source*):

```
<h1>Hello, World !</h1>
```

Response ir objekts.

Write() ir viena no objekta metodēm.

Var izmantot saīsinātu *Write* formu.

```
<h1><% = "Hello, World !" %></h1>
```

Cita iespēja palaist skriptu: elements **<script>** ar atribūtu **runat**.

```
<script language="VBScript" runat="Server">  
    Response.Write "<h1>Hello, World !</h1>"  
</script>
```

Var izmantot arī pievienošanas operatoru **With**.

<%

With Response

.Write "h1>"

.Write "Hello, World !"

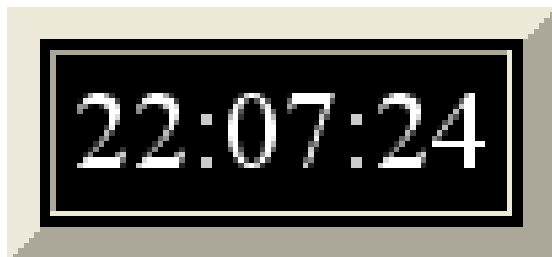
.Write ""

End With

%>

Var pielietot CSS – stilus.

Uzdevums: izveidot elektroniskos pulksteņus.



Tīmekļa lappuses fragments

```
<% @Language = "VBScript" %>
...
<table border="6" cellpadding="3"
  style="color:white;background-color:black">
  <tr><td><% =Time %></td></tr>
</table>
```

Skriptos izmanto arī valodu *JScript*:

```
<% @Language="JScript" %>
<body>
  ...
  <%
    Response.Write("<h1>Hello, World !</h1>")
  %>
</body>
```

VBScript *nav* reģistrjūtīga valoda

```
<% @Language = "VBScript" %>
...
<%
    Response.Write("Hello !")
    Response.wRitE("Hello !")
    REsponse.write("Hello !")
%>
```

JScript *ir* reģistrjūtīga valoda. ASP *objektiem* ir vienīgi pareizs uzrakstīšanas stils. *Locekļu* uzrakstīšana ir brīva.

```
<% @Language="JScript" %>
...
<%
    Response.Write("Hello !")
    Response.wRitE("Hello !")
//    REsponse.Write("Hello !") Kļūda
%>
```

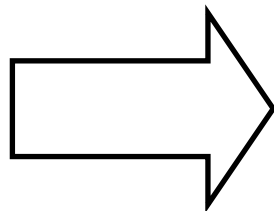

Piemērs ar pulksteņiem

```
<table border="6" cellpadding="6"
  style="color:white;background-color:black">
  <tr>
    <td>
      <%
        var D = new Date();
        Response.Write(D.getHours() + ":" +
          D.getMinutes() + ":" +
          D.getSeconds());
      %>
    </td>
  </tr>
</table>
```

Mainīgie ASP skriptos

Ieplānotais rezultāts

Skripta fragments



Testing
Testing
Testing
Testing
Testing



```
<% For i=20 To 40 Step 5 %>  
    <span style="font-size=<% =i %>" >  
        Testing  
    </span>  
    <br>  
<% Next %>
```

Tīmekļa lappuses *pāradresēšana*

Uzdevums: lietotāju darbību rezultātā tika izvēlēta konkrētā tīmekļa lappuse. Izpildīt pāradresēšanu uz šo lappusi.

<%

```
Lang = "Java" 'programmēšanas valoda  
Response.Buffer = True 'ieteicams rakstīt
```

```
Select Case Lang
```

```
Case "Java"
```

```
Response.Redirect "Java.asp"
```

```
Case "Cpp"
```

```
Response.Redirect "Cplusplus.asp"
```

```
Case "Cs"
```

```
Response.Redirect "Csharp.asp"
```

```
End Select
```

%>

Tīmekļa lappuses *atjaunināšana*

Uzdevums: atjaunināt lappuses saturu vienu reizi sekundē.

```
<% Response.AddHeader "Refresh", "1" %>
<%
    =Time() ` atjaunināšanas kontrole
%>
```

Uzdevums: norādīt informācijas novecošanas periodu kešatmiņā. Perioda vērtība: viena minūte.

```
<% Response.Expires ="1" %>
```

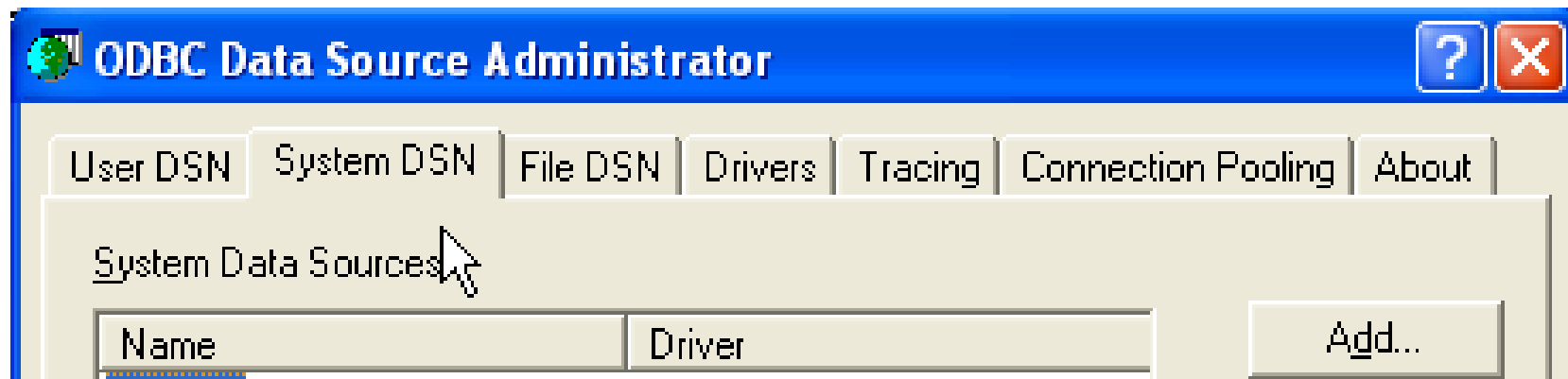
Uzdevums: norādīt informācijas novecošanas laiku kešatmiņā.

```
<%
    Response.ExpiresAbsolute = #September 1, 2010#
%>
```

Datu avota izveidošana

1. *Control Panel -> Administrative Tools -> Data Sources (ODBC).*

2. Ielikums *System DSN*.

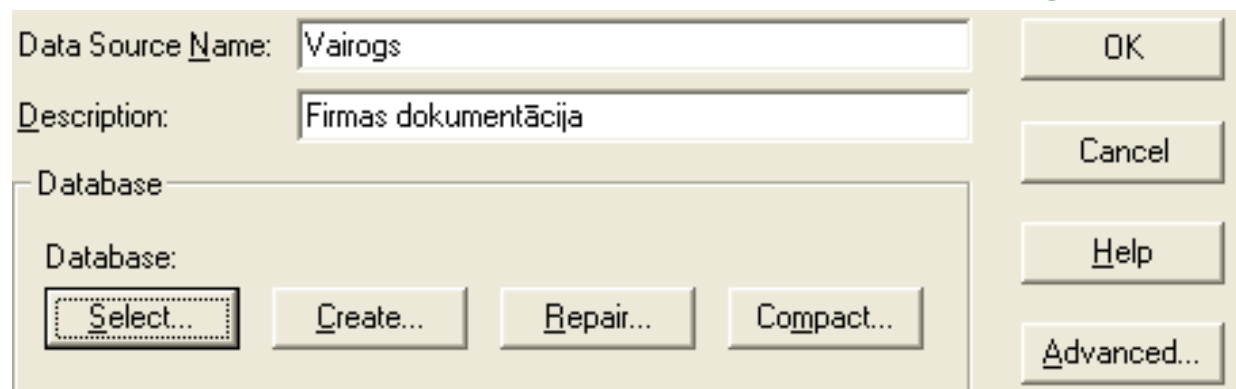


Piezīme: ielikumu *User DSN* nevar izmantot darbā ar *Web* avotiem.

3. Nospieš pogu *Add*. Norādīt draiveri. Nospieš *Finish*.



4. Nosaukt datu avotu. Lai būs **Vairogs**.



5. Nospieš *Select* un norādīt ceļu **.mdb* failam.

6. Rezultāts: jauns datu avots sarakstā (2. solis)

Uzdevums: izvadīt ekrānā dažu tabulas lauku saturu.

```
<% @Language = "VBScript" %>
<% Option Explicit %>
...
<%
    Dim cn, openstr, sql, rs, i
    Set cn =
        Server.CreateObject("ADODB.Connection")
    openstr = "MyFirm"
    cn.Open openstr

    sql = "SELECT Name, Surname, Duty
        FROM Employees"
    Set rs =
        Server.CreateObject("ADODB.Recordset")
    rs.Open sql, cn, 3, 3
```

On Error Resume Next

```
rs.MoveFirst
```

```
i = 1
```

Do While Not rs.EOF

```
Response.Write("<b>" & i & "</b>" & ". ")
```

```
Response.Write Server.HtmlEncode (_
```

```
rs.Fields("Name") & " " & _
```

```
rs.Fields("Surname"))
```

```
Response.Write("<i>, ")
```

```
Response.Write
```

```
Server.HtmlEncode(rs.Fields("Duty"))
```

```
Response.Write("</i><br>")
```

```
i = i + 1
```

```
rs.MoveNext
```

Loop


```
rs.close  
Set rs=nothing  
cn.close  
Set cn=nothing  
%>
```

Skripta izpildes rezultāts

1. Jānis Strods, *programmētājs*
 2. Sergejs Ivanovs, *operators*
 3. Uldis Zemzars, *operators*
-

Piezīme: var norādīt simbolu kodēšanas tabulu.

```
Response.Charset="Windows-1257"
```

Var arī izmantot elementu **<meta>** no HTML.

Uzdevuma risinājums *JavaScript* valodā

```
//Obligātā rindiņa
<% @Language=JScript %>

//Mainīgo deklarēšana
<%
    var cn, openstr, sql, rs, i;

    //Savienojuma un datu kopas izveidošana
    cn = Server.CreateObject("ADODB.Connection");
    openstr = "MyFirm";
    cn.Open(openstr);
    sql = "SELECT Name, Surname, Duty FROM
        Employees";
    rs = Server.CreateObject("ADODB.Recordset");
    rs.Open(sql, cn, 3, 3);
```

```
//Tabulas apstrāde un informācijas izvade
```

```
i = 1;
```

```
while (!rs.EOF) {  
    Response.Write("<b>" + i + "</b>" + ". ");  
  
    Response.Write(Server.HtmlEncode (  
        rs.Fields("Name") + " " +  
        rs.Fields("Surname")));  
  
    Response.Write("<i>, ");  
    Response.Write(Server.HtmlEncode (  
        rs.Fields("Duty")));  
  
    Response.Write("</i><br>");  
    i++;  
    rs.MoveNext();  
}
```

```
//Savienojuma un ierakstu kopas atcelšana  
rs.close();  
rs=null;  
cn.close();  
cn=null;  
%>
```

Operāciju ar objektu *ADO.Connection* analīze

1. *Radīšana*

```
Set cn=Server.CreateObject("ADODB.Connection")
```

2. *Atvēršana*

```
<Connection>.Open ConnectionString [, UserID,  
Password]
```

Visām piešķirēm ir viens un tāds pats efekts

```
openstr = "MyFirm"
```

```
openstr = "Data Source=MyFirm"      'ADO
```

```
openstr = "DSN=MyFirm"              'ODBC
```

Draivera norādīšana

```
openstr = "driver={Microsoft Access Driver  
(* .mdb) }; " &  
    "dbq=" & Server.MapPath("firm.mdb")
```

Pareizs pilns ceļš līdz failam tiks izveidots automātiski.

3. *Aizvēršana*

```
cn.close
```

4. *Atbrīvošana*

```
Set cn=nothing
```

Operāciju ar objektu *ADO.Recordset* analīze

1. *Radīšana*

```
Set rs = Server.CreateObject("ADODB.Recordset")
```

2. *Atvēršana*

```
<Recordset>.Open Source, ActiveConnection,  
    CursorType, LockType, Options
```

```
rs.Open "Employees", cn, 3, 3
```

Darbam ar konstanšu vārdiem izmanto failu *adovbs.inc*.

Faila izvietošana:

C:\Program Files\Common Files\System\ado

Faila izmantošana skriptā

```
<head>
```

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

```
</head>
```

Dažas *CursorType* vērtības

`adOpenStatic` (3)

Statiskais kursors. Ierakstu kopas kopija, nevar tikt izmainīta. Izmanto datu attēlošanai un meklēšanai.

`adOpenDynamic` (2)

Dinamiskais kursors. Ir iespējami papildinājumi, izmaiņas un dzēšana; atļautas *jebkuras* pārvietošanas starp ierakstiem.

Dažas *LockType* vērtības

`adLockReadOnly` (1) . Tikai *lasīšana*.

`adLockPessimistic` (2) . *Pesimistiskais* bloķējums.
Notiek uzreiz pēc ieraksta modificēšanas sākuma.

`adLockOptimistic` (3) . *Optimistiskais* bloķējums.
Ieraksti netiks bloķēti līdz metodes *Update()* izsaukumam.

`adLockBatchOptimistic` (4) .

Optimistiskais *pakešu* bloķējums.

ADO saglabā visas izmaiņas ierakstu kopā un pielieto pēc metodes *BatchUpdate()* izsaukuma.

Dažas *Options* vērtības

`adCmdText (1) .`

Avotu definē *komandas teksts* (pēc noklusēšanas).

```
rs.Open "SELECT * FROM Employees", cn, 3, 3, 1
```

`adCmdTable (2) .`

Avots ir *tabulas* vārds.

```
rs.Open "Employees", cn, 0, 3, 2
```

`adCmdStoredProc (3) .`

Avots ir *glabājamās procedūras* vārds.

Pārvietošana starp ierakstiem

Objektam *Recordset* ir *piecas* metodes

1. `MoveFirst()` . Pārvietoties uz *pirmo* ierakstu.
2. `MoveLast()` . Pārvietoties uz *pēdējo* ierakstu.
3. `MoveNext()` . Pārvietoties uz *nākošo* ierakstu.
4. `MovePrevious()` . Pārvietoties uz *iepriekšējo* ierakstu.

Piezīme: ja ierakstu kopas atvēršanas procesā bija norādīts `adOpenForwardOnly`, notiks kļūda.

5. `Move()` . Norādīt jauno ierakstu.

Uzdevums: izmainīt ierakstu apstrādes secību.

```
rs.MoveLast
```

```
i = 1
```

```
Do While Not rs.BOF
```

```
...
```

```
  i = i + 1
```

```
  rs.MovePrevious
```

```
Loop
```

Cits risinājums:

```
rs.Move 0, 2
```

```
Do While Not rs.BOF
```

```
...
```

```
  rs.Move -1, 0
```

```
Loop
```

Problēma: **ne visa** programmatūra atbalsta metodi *Move()*.

Daža programmatūra atbalsta īpašību *AbsolutePosition*.

`pāriet uz otro ierakstu (numuri 1, 2,...)
rs.AbsolutePosition = 2

Ierakstu daudzums tabulā

Response.Write "Total records: " &
rs.RecordCount & "
"

Grāmatzīmes

rs.Move 1, 1	`otrais ieraksts
Dim Second	`grāmatzīmes deklarācija
Second = rs.Bookmark	`grāmatzīmes saņemšana
rs.Move 0, 1	`pāriet uz citu ierakstu
...	
`atgriezties pēc aizlikšanas	
rs.Bookmark = Second	

Lauku apstrāde

Ierakstu kopa *Recordset* satur kolekciju *Fields*.

Katram ieraksta laukam atbilst objekts *Field*.

Objekti *Field* tiks automātiski izveidoti objekta *Recordset* atvēršanas procesā.

Lauku daudzums

```
Response.Write rs.Fields.Count & "<br>" ` 3
```

Piekļuves sintakse

```
<Recordset>.Fields(Index).Property
```

Index - lauka *vārds* vai *pozīcija*

Īpašība *Field.Value*

Īpašība “pēc noklusēšanas”.

Visām apakšējām rindiņām ir *viens un tāds pats* efekts.

```
rs.Fields("Name").Value
```

```
rs.Fields("Name")
```

```
rs("Name").Value
```

```
rs("Name")
```

```
rs(0).Value           `Lauka indekss
```

```
rs(0)
```

Piezīme: nav ieteicams orientēties uz *skaitliskiem* indeksiem.
Lauku *secība* var izmainīties.

Uzdevums: izvadīt informāciju par visiem ierakstiem tabulas veidā.

Dim i

Response.Write "<table border='2'>"

Do While Not rs.EOF

Response.Write "<tr>"

For i=0 **To** rs.Fields.Count-1

Response.Write "<td>" &
rs.Fields(i).Value & "</td>"

Next

Response.Write "</tr>"

rs.MoveNext

Loop

Response.Write "</table>"

Cita iespēja apstrādāt visus laukus: cikls **For – Each**.

```
Do While Not rs.EOF
    With Response
        .Write "<p>"
        For Each F In rs.Fields
            .Write F & " "
        Next
        .Write "</p>"
    End With
    rs.MoveNext
Loop
```

Iegūtais rezultāts:

```
<p>Jānis Strods programmētājs </p>
<p>Sergejs Ivanovs operators </p>
<p>Uldis Zemzars operators </p>
```


Citas objekta *Field* īpašības

1. *Name*. Lauka vārds.

```
Response.Write rs.Fields(1).Name 'Surname
```

2. *DefinedSize*. Maksimālais lauka izmērs.

```
Response.Write rs.Fields(1).DefinedSize '50
```

3. *ActualSize*. Lauka izmērs.

```
Response.Write rs.Fields(1).ActualSize '6
```

4. *Type*. Lauka tips, skaitliskā vērtība.

```
Response.Write rs.Fields(1).Type '202
```

Dažas konstantes:

adTinyInt: 16 (8 bitu vesels skaitlis ar zīmi)

adSmallInt: 2 (16 bitu vesels skaitlis ar zīmi)

adInteger: 3 (32 bitu vesels skaitlis ar zīmi)

adBigInt: 20 (64 bitu vesels skaitlis ar zīmi)

adUnsignedTinyInt: 17 (8 bitu bez zīmju vesels skaitlis)

` v ē l t r ī s b e z z ī m j u v e s e l i e s k a i t l ī

adSingle: 4 (32 bitu reālais skaitlis)

adDouble: 5 (64 bitu reālais skaitlis)

adDate: 7 (8 bitu datums)

adDBDate: 133 (datums formātā `yyyymmdd`)

adDBTime: 134 (laiks formātā `hhmmss`)

Darbs ar glabājamiem vaicājumiem

Visos gadījumos SQL operators bija *iebūvēts* tīmekļa lappusēs.

Priekšrocība: viss kods atrodas vienā vietā.

Trūkumi:

1. Vienu un to pašu *SQL* – kodu bieži *jāatkārto* vairākās tīmekļa lappuses.
2. *SQL* – operatora modifikācijas gadījumā jāapstrādā *visas tīmekļa lappuses*.
3. Datubāzi var apkalpot speciālisti *SQL* – kodā, kuri tikai raksta vaicājumus un vispār neko nezina par ASP un ADO.

Objekts *ADO.Command*

1. Savienojumu (*Connection*) izveido kā parasti.

```
Set cn = Server.CreateObject("ADODB.Connection")  
openstr = "MyFirm"  
cn.Open openstr
```

2. Objekta *Command* izveidošana.

```
sql = "SELECT Name, Surname, Duty FROM " &_  
      "Employees"  
Set cmd = Server.CreateObject("ADODB.Command")  
Set cmd.ActiveConnection = cn  
  
cmd.CommandText = sql  
cmd.CommandType = adCmdText
```

3. Izveidot un atvērt objektu *Recordset*.

```
Set rs = Server.CreateObject("ADODB.Recordset")  
rs.CursorType = adOpenForwardOnly  
rs.LockType = adLockReadOnly  
Set rs = cmd.Execute()
```

Objekta *Recordset* atvēršana **bez** *Command*: 3 operatori.

Objekta *Recordset* atvēršana **ar** *Command*: 9 operatori.

Secinājums: objektu *Command* parasti neizmanto vienkāršos vaicājumos. Visbiežāk runa ir par diviem gadījumiem:

1. Komandas rezultāts nav ierakstu kopa (visas *SQL* komandas, izņemot **SELECT**).
2. Komanda ir parametrizētais vaicājums.

Uzdevums: iegūt informāciju par darbinieku, izmantojot viņa uzvārdu kā parametru.

Parametrizētā vaicājuma *šablons* un *teksts*.



The screenshot shows a Microsoft Access query design grid. At the top, there is a box labeled 'Employees' containing a list of fields: *, Name, Surname, Duty, and Salary. Below this, the design grid is visible with the following fields and properties:

	Field:	Table:	Sort:	Show:	Criteria:
	Employees.*	Employees		<input checked="" type="checkbox"/>	
				<input type="checkbox"/>	[Surname:]

```

SELECT Employees.*
FROM Employees
WHERE (Employees.Surname=[Surname:]) ;
    
```

Parametrizētā vaicājuma izmantošana skriptā:

1. ADO savienojuma (*ADO Connection*) atvēršana.
2. Objekta *ADO Command* izveidošana glabājamā vaicājuma izsaukumam.
3. Objekta (objektu) *ADO Parameter* izveidošana parametra vārda un vērtības glabāšanai.
4. Objekta (objektu) *ADO Parameter* pievienošana objektam *ADO Command*.
5. Objekta *ADO Recordset* izveidošana ar objekta *ADO Command* metodes *Execute()* palīdzību.

Skripta kods

```
` mainīgo deklarēšana
Dim cn, rs, cmd, openstr, Empl, pSur, i

` savienojuma veidošana
Set cn = Server.CreateObject("ADODB.Connection")
openstr = "MyFirm"
cn.Open openstr

` objekta Command veidošana
Set cmd = Server.CreateObject("ADODB.Command")
Set cmd.ActiveConnection = cn

` parametrizētā vaicājuma norādīšana
cmd.CommandText = "GetInfo"
cmd.CommandType = adCmdStoredProc
```


` parametra radīšana

Empl = "Strods" ` vērtība no formas

Set pSur = Server.CreateObject("ADODB.Parameter")

pSur.Name = "Surname"

pSur.Type = adVarChar

pSur.Direction = adParamInput

pSur.Size = Len(Empl)

pSur.Value = Empl

` parametra pievienošana

cmd.Parameters.Append pSur

` ierakstu kopas radīšana

Set rs = cmd.Execute()

```
` pārbaude: vai ir atrasti ieraksti ?  
If rs.EOF Then  
    Response.Write  
        "Employee with surname: '<b>' &  
        Empl & "</b>' not found."  
End If
```

Kļūdu apstrāde: savienojuma pārbaudes piemērs.

```
If Err.Number<>0 Then  
    Response.Write  
        "<p> Cannot connect to database</p>"  
    Flag = false  
End If
```

Err.Description – kļūdas apraksts.

Err.Source – kļūdas avots.

Ir arī cita iespēja izveidot parametru (bez `Server.CreateObject`)

```
Empl = "Strods" 'vērtība no formas
```

```
Set pSur = cmd.CreateParameter("Surname",  
adVarChar, adParamInput, Len(Empl), Empl)
```

```
cmd.Parameters.Append pSur
```

Metodes `CreateParameter` argumenti:

```
Command.CreateParameter(Name, Type, Direction,  
Size, Value)
```

Dažas citas *Direction* vērtības

`adParamOutput` – parametrs izvadei (SQL Server)

`adParamInputOutput` – parametrs ievadei/izvadei

Darbs ar atslēgto objektu *Recordset*

Iepriekš apskatītos gadījumos objekts *Recordset* bija atkarīgs no objekta *Connection*.

Objekta *Connection* aizvēršana automātiski iznīcina visus meitas objektus.

Modernās ASP versijās objektu *Recordset* var atslēgt.

Objekts *Connection* būs vajadzīgs tikai datu nolasīšanas procesā.

Rezultātā tiks paaugstināts ražīgums.

Atslēgtā režīma piemērs

```
openstr = "driver={Microsoft Access Driver  
(* .mdb) };" & "dbq=" & Server.MapPath("firm.mdb")  
  
Set cn = Server.CreateObject("ADODB.Connection")  
  
cn.Mode = adModeRead  
cn.ConnectionString = openstr  
cn.CursorLocation = adUseClient  
cn.Open  
  
Set rs = cn.Execute("SELECT * FROM Employees", ,  
    adCmdText)  
  
Set rs.ActiveConnection = Nothing  
cn.close  
Set cn=Nothing  
'recordset apstrāde
```

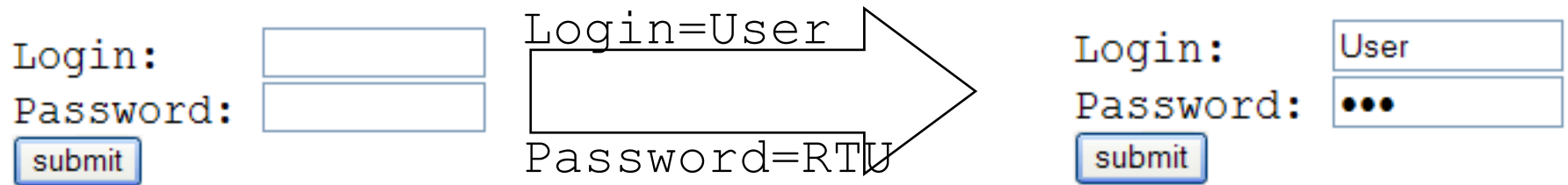
Formu apstrāde

Ir divas informācijas pārraides metodes: **Get** un **Post**.

1. **Get** gadījumā pārraidāmās informācijas apjoms ir ierobežots.
2. **Get** gadījumā lietotājs redzēs mainīgo vārdus un citu informāciju adreses rindā.

Lai ir formas fragments:

```
<form name="Check" method="get" ...>  
  ...<input type="text" name="Login"  
    size="12"><br>  
  ...<input type="password" name="Password"  
    size="12"><br>  
  <input type="submit" value="submit">  
</form>
```



Adreses rinda pēc *Submit*:

`Info.htm?Login=User&Password=RTU`

Adreses rindā redzama *pat parole* (formā bija “zvaigznītes”).

Lai formā norādīta metode `Post`.

```
<form name="Check" method="post" ...>
```

Adreses rinda pēc *Submit* nesatur nekādu informāciju.

`Post` gadījumā ir parametru saraksts.

Ieteicams orientēties uz metodi `Post`.

Pēc noklusējuma: metode `Get`.

Uzdevums: forma satur informāciju par lietotāja *vārdu*, *vecumu*, *pieredzi* noteiktajā nozarē un *izglītību* (var būt izvēlētas vairākas vērtības).

Name:

Age: <20 ☐ 20-40 ☒ >40 ☐

Level:

Diploma:
Master
Doctor

User:

UserName: Jānis Strods

Age: 20-40

Level: Professional

Diploma: Bachelor, Engineer

Apstrādāt informāciju par lietotāju *ASP* – skriptā.

Darbam izmantot *kadru struktūru*.

Kadru struktūra (fails *form_frame.htm*) .

```
<frameset cols="2*, 3*">  
  <frame src="MyForm.htm">  
  <frame src="TestUser.htm" name="TestForm">  
</frameset>
```

Formas atribūti

1. **Name**. Formas vārds.
2. **Method**. Informācijas pārraides metode.
3. **Action**. Apstrādes skripts.
4. **Target**. Rezultātu izvade. Piemēram, var organizēt kadru (*frames*) sistēmu.

Tīmekļa lappuses fragments

```
<!-- Formas deklarācija -->
```

```
<form
```

```
    name      = "User"
```

```
    method    = "Post"
```

```
    action     = "http://localhost/MyHost/TestUser.asp"
```

```
    target     = "TestForm"
```

```
>
```

```
<!-- Formas struktūra -->
```

```
<table cellpadding="3">
```

```
  <tr>
```

```
    <td><b>Name:</b></td>
```

```
    <td>
```

```
      <input name="UserName" type="text">
```

```
    </td>
```

```
  </tr>
```

```
<!-- Formas struktūra (turpinājums) -->
```

```
<!-- Informācija par vecumu -->
```

```
<tr>
```

```
  <td><b>Age: </b></td>
```

```
  <td>
```

```
    &lt;20<input type="radio" name="Age"
      value="below20">
```

```
    20-40<input type="radio" name="Age"
      value="20-40">
```

```
    &gt;40<input type="radio" name="Age"
      value="above40">
```

```
  </td>
```

```
</tr>
```

```
<!-- Formas struktūra (turpinājums) -->
<!-- Informācija par pieredzi nozarē -->

<tr>
  <td><b>Level:</b></td>
  <td>
    <select name="Level">
      <option name="b">Beginner
      <option name="p">Professional
      <option name="e">Expert
    </select>
  </td>
</tr>
```

```
<!-- Formas struktūra (turpinājums) -->
<!-- Informācija par diplomu -->

<tr>
  <td><b>Diploma:</b></td>
  <td>
    <select name="Diploma" multiple>
      <option>Bachelor
      <option>Engineer
      <option>Master
      <option>Doctor
    </select>
  </td>
</tr>
```

```
<!-- Formas struktūra (nobeigums) -->

<!-- Poga formas nosūtīšanai -->
<tr>
  <td colspan="2" align="right">
    <input type="submit" value="send">
  </td>

</tr>
</table>
</form>
```

Faila *TestUser.htm* fragments (vēlāk kadrā būs rezultāti)

```
<body>
  <h2>User:</h2>
</body>
```

Formas apstrādes pamati

1. Informācijas iegūšanu no pārlūkprogrammas nodrošina *ASP* objekts *Request*.
2. *Request.Form* ir datu kolekcija metodes **POST** izmantošanas gadījumā.
3. Kolekcijā saglabāti datu pāri formātā „*atslēga = vērtība*”.
4. Ja atslēgai ir vairākas vērtības, tās tiks ierakstītas sakārtotajā formā.

Piekluve elementiem

`Request.Form.Item(<Element Name>)`

`Request.Form.Item(<Element Index>)`

Vērtība *Item* ir vērtība pēc noklusēšanas.

Visām apakšējām koda rindiņām ir viens efekts

```
Request.Form.Item("UserName")
```

```
Request.Form("UserName")
```

```
Request.Form.Item(1)
```

Elementu daudzums kolekcijā *Form*.

```
Request.Form.Count
```

Skripta fragments (mainīgo deklarēšana)

```
<%
```

```
Response.Write "<h2>User:</h2>"
```

```
Dim i, S
```

```
S = ""
```


Skripta fragments (apstrādes cikls)

```
For i = 1 To Request.Form.Count
    S = S & "<b>" & Request.Form.Key(i) & _
        "</b>" & ": " & Request.Form.Item(i) & _
        "<br>"
Next
Response.Write S
%>
```

Bieži izmantojamus objektus ieteicams saglabāt mainīgajos.

```
Dim UserName
...
UserName = Request.Form("Username")
...
Response.Write UserName
```

Metodes *Get* izmantošana

Izmanto datu kolekciju *QueryString*.

```
Request.QueryString.Item("UserName")
```

```
Request.QueryString("UserName")
```

```
Request.QueryString.Item(1)
```

Formas deklarācija:

```
<form
```

```
    name      = "User"
```

```
    method    = "Get"
```

```
    action    = "http://localhost/MyHost/TestGet.asp"
```

```
    target     = "TestForm"
```

```
>
```

Apstrādes skriptā *Form* vietā būs *QueryString*.

Ar *vienu atslēgu* var būt saistītas *vairākas vērtības*.

Piemērs: cilvēkam var būt vairāki diplomī.

```
For i=1 To Request.Form("Diploma").Count  
    Response.Write i & ". " &  
        Request.Form("Diploma")(i) & "<br>"
```

Next

Informācijas nodošanas *metodes pārbaude*

```
If Request.ServerVariables("REQUEST_METHOD") =  
    "GET" Then  
    ... ` formas apstrāde
```

End If

Alternatīvais variants: pārbaudīt pogu *Submit*.

Lai formas pogai ir vārds:

```
<input type="submit" name="submit">
```

```
If Request.QueryString("Submit") <> "" Then
```

Iespējamā problēma: mēģinājums saīsināt metodes pārbaudes tekstu.

```
If Request ("REQUEST_METHOD") = "GET" Then
```

Piemērā netika minētā kolekcija `ServerVariables`.

Visbiežāk rezultāts būs pareizs. ASP pārbaudīs *visas* iespējamās kolekcijas. Bet tas aizņem kādu laiku.

Tomēr *dažos* gadījumos var būt konflikti.

Piemēram, var būt sakritība ar kādu *cookie*. Tad nav skaidrs, par ko ir runa: formas lauku vai *cookie*.

Secinājums: ieteicams vienmēr norādīt kolekciju.

Ierakstu modificēšana

Lai failā *firm.mdb* jābūt papildu tabula *Hits* ar informāciju par tīmekļa lappuses apmeklēšanu. Tabulas struktūra:

1. Attiecīgais *URL* (lauks *URL*).
2. Apmeklēšanu daudzums (lauks *Counter*).
3. Pēdējās apmeklēšanas datums un laiks (lauks *LastHit*).

Hits : Table			
	Field Name	Data Type	
PK	URL	Text	
	Counter	Number	
	LastHit	Date/Time	

Pēc katras apmeklēšanas tabulā modificē laukus *Counter* un *LastHit*.

Informācija tabulā *Hits*:

Hits : Table			
	URL	Counter	LastHit
	/MyHost/MyHits.asp	69	2003.09.28. 21:54:57
▶		0	

Aktuālā URL iegūšana

```
CurrURL = Request.ServerVariables("PATH_INFO")
```

Piezīme: var apstrādāt kolekciju un iegūt informāciju par visiem servera mainīgajiem.

```
Dim Var
For Each Var In Request.ServerVariables
    Response.Write Var & " -> " & _
    Request.ServerVariables(Var) & _
    "<br>"
Next
```

Servera mainīgie: apstrādes rezultāts.

```
PATH_INFO -> /MyHost/Test.asp  
SERVER_NAME -> localhost  
SERVER_PORT -> 80  
SERVER_PROTOCOL -> HTTP/1.1  
SERVER_SOFTWARE -> Microsoft-IIS/5.1  
REQUEST_METHOD -> GET
```

1. Ierakstu kopas *modifikācija*: metode *Update*.
 2. Ierakstu kopas *atvēršana*: kursora tips *adOpenDynamic*.
-

```
CurrURL = Request.ServerVariables("PATH_INFO")  
CurrURL = UCase(CurrURL)  
Response.Write "<h1>Current page: " & CurrURL &  
"</h1>"
```

```
Set cn = Server.CreateObject("ADODB.Connection")
openstr = "MyFirm"
cn.Open openstr

sql = "SELECT URL, Counter, LastHit FROM Hits " & _
      "WHERE UCase(URL) = '" & CurrURL & "'"
Set rs = Server.CreateObject("ADODB.Recordset")
rs.Open sql, cn, adOpenDynamic, adLockPessimistic,
      adCmdText

If rs.EOF Then
    Response.Write "<h2>No hits.</h2>"
Else
```



```
Temp = rs("Counter") + 1
rs("Counter") = Temp
rs("LastHit") = Now()
rs.Update
Response.Write "<h2>Total hits:" & Temp & _
               ".<br> Last hit: " & rs("LastHit") & _
               "</h2>"
```

End If

```
rs.close
Set rs=nothing
cn.close
Set cn=nothing
```

Current page: /MYHOST/MYHITS.ASP

Total hits:79.

Last hit: 2003.09.28. 22:25:59.

Piezīme: tabulu *Hits* var apstrādāt bez vaicājuma.

```
Set rs = Server.CreateObject("ADODB.Recordset")
rs.Open "Hits", cn, adOpenDynamic,
    adLockPessimistic, adCmdTable

Do While Not rs.EOF
    If (UCase(rs("URL")) = CurrURL) Then
        Temp = rs("Counter") + 1
        ...
    Exit Do
End If
Rs.MoveNext
Loop

If rs.EOF Then
    Response.Write "<h2>No hits.</h2>"
End If
```

Komponents *Page Counter*

Iespēja saskaitīt tīmekļa lappušu apmeklējumus.

Komponents saistīts *tikai ar vienu* tīmekļa lappusi.

Tas periodiski saglabā informāciju par lappuses apmeklējumu daudzumu teksta failā *HitCnt.cnt*.

Metodes:

1. *PageHit* : skaitītāja palielināšana par 1.
2. *Hits*: apmeklējumu daudzums.
3. *Reset* : piešķirt skaitītājam vērtību 0.

Skaitītāja palielināšana:

```
Dim pc
```

```
Set pc = Server.CreateObject("MSWC.PageCounter")
```

```
pc.PageHit
```

```
...
```

```
<div>Total visits: <b><% =pc.Hits %></b>.</div>
```

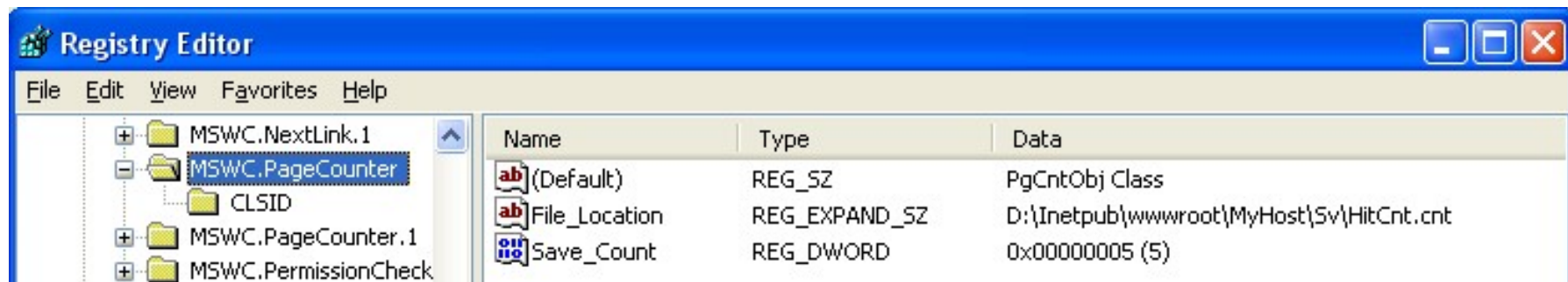
Rezultāts:

Total visits: 17.

Piešķirt skaitītājam vērtību 0:

```
<% pc.Reset %>
```

Objektam *Page Counter* izveido atslēgu *sistēmas reģistrā*:
HKEY_CLASSES_ROOT\MSWC.PageCounter



Atslēgai ir divas vērtības:

1. *File_Location* – teksta rindiņa.

Satur faila *HitCnt.cnt* katalogu.

2. *Save_Count* – vērtība *DWORD*.

Apmeklējumu daudzums (“sliexnis”), pēc kura
sasniegšanas rezultāti tiks saglabāti failā.

Faila *HitCnt.cnt* fragments (divas kolonas, apmeklējumu daudzums un tīmekļa lappuse).

17 /MyHost/Sv/main.asp

9 /MyHost/Sv/sub.asp

Ierakstu grupas modificēšana

Ja modifikāciju pārāk daudz, labāk atvērt objektu *Recordset* grupas *modifikācijas* režīmā.

Visas izmaiņas tiks izpildītas *datu kopijā*, izmaiņas *datubāzē* notiks tikai pēc metodes *BatchUpdate* lietošanas.

Uzdevums:

1. Piešķirt visiem apmeklēšanas skaitītājiem vērtību 0.
2. Attīrīt datuma lauku visiem ierakstiem.

'ierakstu kopas atvēršana darbam pakešu režīmā.

'Bloķējums adLockBatchOptimistic

```
Set rs = Server.CreateObject("ADODB.Recordset")  
sql = "SELECT Counter, LastHit FROM Hits;"  
rs.Open sql, cn, adOpenStatic,  
    adLockBatchOptimistic, adCmdText
```

'ierakstu apstrāde ciklā.

'Metodes Update un UpdateBatch

```
Do While Not rs.EOF  
    rs("Counter") = 0  
    rs("LastHit") = null  
    rs.MoveNext
```

Loop

```
rs.UpdateBatch
```

Ierakstu pievienošana

1. Izveidot jaunu ierakstu (metode *AddNew*).
2. Modificēt attiecīgo lauku vērtības.
3. Modificēt ierakstu kopu (metode *Update*).

Lai datubāze *firm.mdb* nesatur informāciju par kādu *URL*, un jāieraksta informācija par to.

Izmainītais programmas fragments (datu avots – *SQL* vaicājums).

```
If rs.EOF Then  
    rs.AddNew  
    rs("URL") = CurrURL  
    Temp = 1  
Else  
    Temp = rs("Counter") + 1  
End If
```



```
rs("Counter") = Temp  
rs("LastHit") = Now()  
rs.Update
```

```
Response.Write "<h2>Total hits:" & Temp &  
    ".<br> Last hit: " & rs("LastHit") & ".</h2>"
```

Ierakstu dzēšana

Darbības: atrast nevajadzīgus ierakstus un pielietot metodi *Delete*.

Uzdevums: izdzēst visus ierakstus par tīmekļa lappusēm, kuras nebija apmeklētas pēdēju 7 dienu laikā.

```
sql = "SELECT URL, Counter, LastHit " &  
      "FROM Hits WHERE (Now()-LastHit)>7"
```

```
Set rs = Server.CreateObject("ADODB.Recordset")
```

```
rs.Open sql, cn, adOpenDynamic,  
        adLockPessimistic, adCmdText
```

```
Do While Not rs.EOF
```

```
    rs.Delete
```

```
    rs.MoveFirst
```

```
Loop
```

Transakcijas konceptija

1. Jebkurš operators, kurš izmaina sākotnēju informāciju, savā būtībā ir *transakcija* (**UPDATE**, **DELETE**, **INSERT**).
2. Vienā ASP transakcijā var apvienot vairākus operatorus.
3. Objekta `Connection` metodes:
 - a. *BeginTrans* – transakcijas sākums.
 - b. *CommitTrans* – transakcijas apstiprināšana.
 - c. *RollbackTrans* – transakcijas anulēšana.

Transakcijas implementēšana

```
cn.BeginTrans
```

```
sql = "UPDATE Employees SET Salary = Salary + 15"
```

```
cn.Execute sql,, adCmdText + adExecuteNoRecords
```

```
...
```

```
... `citas operācijas un vaicājumi
```

```
cn.Execute sql,, adCmdText + adExecuteNoRecords
```

```
If cn.Errors.Count = 0 Then
```

```
    cn.CommitTrans
```

```
Else
```

```
    cn.RollbackTrans
```

```
End If
```

Informācijas apmaiņa: *servera* skripts ↔ *klienta* skripts

Mērķis: samazināt noslodzi uz serveri (klients arī piedalās informācijas apstrādē).

Uzdevums:

1. Iegūt informāciju par *Web* serveri, izmantojot *ASP* skriptu.
2. Demonstrēt šo informāciju lietotājam, izmantojot klienta skriptu. *Piezīme*: abu skriptu valodas var *atšķirties*.

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

```
...
```

```
<body>
```

```
<%
```

```
    SrvName =
```

```
        Request.ServerVariables("SERVER_NAME")
```

```
    SrvSoft =
```

```
        Request.ServerVariables("SERVER_SOFTWARE")
```

```
%>
```

```
<h5>Information about Web server:</h5>
<script language="JScript">
  var Srv
  Srv = "<b>Server:</b> <% =SrvName %>." +
    "<br><b>Software:</b> <% =SrvSoft %>."
  document.write(Srv)
</script>
</body>
```

Rezultāts:

Information about Web server:

Server: localhost.

Software: Microsoft-IIS/5.1.

Kolekcijas *Cookies* (sīkdatnes)

1. *Response.Cookies* (tikai ierakstīšana).
2. *Request.Cookies* (tikai lasīšana).

Vienkāršākajā gadījumā *cookie* satur *vārdu* (atslēgu) un *vērtību*.

Uzdevums: saglabāt klienta pārlūkprogrammā informāciju par pēdēju tīmekļa lappuses apmeklēšanu.

<%

```
Response.Cookies("VEF_LastVisit") = Time
```

%>

Saglabātas informācijas analīze:

<%

```
If Request.Cookies("VEF_LastVisit") <> "" Then
```

```
    Response.Write "Hello, visitor ! " & _  
        "Your last visit: " & _  
        Request.Cookies("VEF_LastVisit")
```

```
Else
```

```
    Response.Write "Hello, new visitor !"
```

```
End If
```

%>

Citas tīmekļa lappuses izsaukums

To var izpildīt ar metodes *Server.Execute* palīdzību.

Uzdevums: izveidot tīmekļa lappusi, kura veido dokumenta kājēni formātā: *<organizācija> <aktuālais laiks>*.

Izsaucošās tīmekļa lappuses fragments

```
<h1>Home Page</h1>
```

```
...
```

```
<% Server.Execute "footer.asp" %>
```

Izsaucamā tīmekļa lappuse

```
<table width="100%"  
  style="position:relative;margin-top:50pt">  
  <tr>  
    <td>RTU</td>  
    <td align="Right">  
      <% =Now %>  
    </td>  
  </tr>  
</table>
```

Piezīme: to pašu rezultātu var iegūt arī citādi.

```
<h1>Home Page</h1>  
<!-- #include file="footer.asp" -->
```

Pēdējā laikā cita faila pieslēgšana nav populāra.

Globālo mainīgo radīšana

ASP objekts `Application`, kolekcija `Contents`, īpašība `Value`.

Uzdevums: saglabāt informāciju par izvēlēto lidostu kā globālo mainīgo.

```
Application.Value("Airport") = "RIX"
```

```
Application("Airport") = "RIX"
```

Informācijas lasīšana *citā* tīmekļa lappusē:

```
Response.Write(Application.Contents("Airport"))
```

Kolekcijas elementa attīrīšana:

```
Application("Airport") = ""
```

```
Application.Value("Airport") = Empty
```

Elementu daudzums kolekcijā Contents

```
Application.Contents.Count
```

Kolekcijas elementu izvade (*atslēga* – vērtība)

```
For Each Key In Application.Contents  
    Response.Write Key & " " &  
        Application.Value(Key)
```

Next

Kolekcijas elementu izvade (*numurs* – vērtība)

```
For i=1 To Application.Contents.Count  
    Response.Write i & " " &  
        Application.Contents(i)
```

Next

Piezīme: pēdējā gadījumā **neder** variants:

...

```
Application.Value(i) ' 1
```

Application ir vienīgais objekts, kuru *vienlaicīgi* izmanto *vairāki* lietotāji.

Objekta *modifikācijas* laikā ieteicams izmantot *bloķēšanu*

```
Application.Lock
```

```
Application.Value("Airport") = "RIX"
```

```
Application.Unlock
```

Atslēgas *eksistēšanas* pārbaude

```
If Application("Airport") <> Empty Then
```

Lai kādu mainīgo izmantos *vairākas reizes*.

Tad ieteicams izveidot mainīgā *kopiju*.

```
Dim Airp
```

```
Airp = Application("Airport")
```

```
`Airp apstrāde
```

Objektā `Application` var saglabāt masīvu.

Masīva *modifikācijas* procesā *obligāti* jāizmanto mainīgais-starpnieks.

```
Langs = Array("C++", "Java")
```

```
Application.Lock
```

```
Application("Langs") = Langs
```

```
Application.Unlock
```

```
Langs = Application("Langs")
```

```
Langs(0) = "C#"
```

```
Application("Langs")(0) = "C#" `nav izmaiņu
```

```
Application.Lock
```

```
Application("Langs") = Langs
```

```
Application.Unlock
```

Standartizēta shēma: bloķēšana, modificēšana, atbloķēšana.

Objekts *Dictionary* (vārdnīca)

Vārdnīcas *radīšana*

```
Set d =  
    Server.CreateObject("Scripting.Dictionary")
```

Jauno elementu *pievienošana* (tiešā metode)

```
d.Add "cpp", "C++"  
d.Add "cs", "C#"
```

Jau eksistējošu atslēgu *nevar* pievienot divas reizes.

```
d.Add "cpp", "C" `Kļūda
```

Atslēgas *eksistēšanas* pārbaude un elementa *dzēšana*

```
If d.Exists("cpp") Then  
    d.Remove("cpp")  
End If
```

Piezīme: to pašu paņēmieni ieteicams izmantot *elementu pievienošanai*.

Objekts sastāv no diviem masīviem (tips `Variant`).

Vārdnīcas atslēga **nevar** būt masīvs.

Salīdzinājumā ar vairākām citām kolekcijām, vārdnīca nodrošina *ātru informācijas meklēšanu* pēc atslēgas.

Dictionary ir COM objekts, *kompilētais* kods izpildās ātrāk.

Objektu **nevar** saglabāt mainīgajā `Session`. Pretējā gadījumā var iegūt *pavedienu konfliktu* un tīmekļa mezgla bloķēšanu.

Vērtības lasīšana pēc atslēgas

```
Response.Write d.Item("cpp")
```

```
Response.Write d("cpp")
```


Piezīme: pēdējā gadījumā tika izmantota netiešā adresēšana.

Nav ieteicams izmantot to, īpaši atslēgas pārbaudes procesā.

```
If (d("pas")="Pascal") Then
```

```
    . . .
```

```
End If
```

Pārbaudes rezultātā atslēga *tika pievienota* vārdnīcai.

Vārdnīcas izvade

```
For Each K In d.Keys
```

```
    Response.Write K & " = " & d(K) & "<br>"
```

```
Next
```

Rezultāts (atslēgai *pas* ir tukšā vērtība)

```
cpp = C++
```

```
cs = C#
```

```
pas =
```

Komponents *AdRotator*

Reklāmas demonstrēšanas automatizācija.

Lietotāja pāradresēšana uz attiecīgo URL resursu pēc peles klikšķinājuma.

Komponenta funkcionēšanai nepieciešami *trīs* faili:

1. ASP fails ar reklāmas sludinājumu.
2. Rotatora saraksta fails (*Rotator Schedule file*).
3. Pāradresēšanas fails (*Redirection file*).

Piezīme: rotatora saraksta fails ir parastais teksta fails ar divām nodaļām.

1. Pāradresēšanas fails un reklāmas attēlu formatēšana.
2. Informācija par katru reklāmas bloku.

1. ASP fails ar rotatoru

```
<%  
    Set AdR = Server.CreateObject("MSWC.AdRotator")  
%>  
  
<% =AdR.GetAdvertisement("list.txt") %>
```

Piezīme: pēdējā rindīnā obligāti jābūt simbols =.

Objektam ir *vienīga* metode: `GetAdvertisement()`.

Metode nolasa informāciju no *rotatora saraksta*.

2. Rotatora saraksta fails

`Redirect URL` – pāradresēšana

`Width` – attēla platums

`Height` – attēla augstums

`Border` – attēla rāmīša biezums (0 pēc noklusēšanas)

Pēc formatēšanas daļas jābūt *.

Katram sludinājumam var būt *četri* parametri

1. `adURL` – fails ar reklāmas attēlu
2. `adHomePageURL` – tīmekļa lappuses adrese
3. `text` – alternatīvais teksts
4. `impressions` – demonstrēšanas laika procents

Piezīme: ja nav tīmekļa lappuses (2), norāda –.

Rotatora saraksta faila piemērs (`list.txt`)

```
Redirect Control.asp
```

```
Width 200
```

```
Height 100
```

```
Border 3
```

```
*
```

RVR.jpg

RVR.asp

Rīgas vagonu rūpnīca

2

VEF.jpg

VEF.asp

Valsts elektrotehniskā fabrika

3

Piezīme: summārais demonstrēšanas laiks ir $2+3=5$.

RVR attēls tiks demonstrēts 40%, bet VEF 60%.

3. Pāradresēšanas fails (ASP)

```
<% Response.Redirect(Request.QueryString("URL")) %>
```

Pēc klikšķinājuma uz attēla VEF notiks pāreja uz VEF.asp.

```
<A HREF="Control.asp?url=RVR.asp&image=RVR.jpg">  
  <IMG SRC="RVR.jpg" ALT="Rīgas vagonu rūpnīca"  
    WIDTH=200 HEIGHT=100 BORDER=3></A>
```

Komponents *ContentRotator*

HTML teksta virkņu rotācija

1. ASP skripti

<%

```
Set ConR =
```

```
    Server.CreateObject("MSWC.ContentRotator")
```

```
Response.Write ConR.ChooseContent("list.txt")
```

%>

Metode ChooseContent izvēlās vienu HTML virkni no faila.

2. Teksta faila fragments

```
%% // Programming languages
```

```
%% #2
```

```
<h1>C++</h1>
```

```
%% #3
```

```
<h1>Java</h1>
```

Piezīme: katram faila ierakstam ir *divas* daļas.

1. %% <svara koeficients>

2. Teksts HTML formātā. Var aizņemt vairākas rindiņas

Lai skripts (1) ielikts HTML tabulā:

```
<table>
  <tr><td>
    <%
      skripts
    %>
  </td></tr>
</table>
```

Rezultāts: tabulas šūnā ielādes laikā būs dažāds teksts.

Metode GetAllContent () izvada ekrānā visas teksta virknes.

1. Эспозито Дино. Знакомство с Microsoft ASP.NET 2.0. Microsoft Press, Санкт-Петербург, издательство “Питер”, 2006. 512 с.

2. Андерсон Ричард, Фрэнсис Брайан, Хомер Алекс. ASP.NET для профессионалов. Москва, Издательство “Лори”, 2005. 630 с.

3. Аньен Фриц. Основы ASP .NET. с примерами на Visual Basic .NET. Москва, “Бином. Лаборатория знаний”, 2005. 400 с.

4. Рассел Джонс. Программирование ASP. NET средствами VB.NET. Полное руководство. Киев, "Век+", 2008. 784 с.

5. Мак-Дональд Мэтью, Шпушта Марио. Microsoft ASP.NET 3.5 с примерами на C# 2008 для профессионалов. 2-е издание. Москва, “Вильямс”, 2008. 1424 с.

ASP .NET

Elementārais skripts (fails *Hello.aspx*):

```
<% Response.Write("Hello !") %>
```

Piezīme: **neder** ASP variants.

```
<% Response.Write "Hello !" %>
```

Iegūsim kompilatora ziņojumu: *Method arguments must be enclosed in parentheses.*

Compiler Error Message: BC30800: Method arguments must be enclosed in parentheses.

Source Error:

```
Line 1:  <% Response.Write "Hello !" %>
```

Uzdevums: izvadīt informāciju par darbiniekiem.

Piezīme: tiks izmantota savietojamība ar klasisko ASP un parasto ADO.

1. Tīmekļa lappuses sākumā ir direktīva:

```
<% @Page aspcompat=true %>
```

2. Visām funkcijām ar parametriem ir apaļas iekavas.

```
cn.Open("MyFirm")
```

3. *Nav Set.*

```
cn = Server.CreateObject("ADODB.Connection")
```

4. Laukiem (*Fields*) obligāti norāda īpašību *Value*.

```
rs.Fields("Name").Value
```

5. Koda rindiņas pārnesums: pirms `_` ir atstarpe.

```
Response.Write(... & _
```

Skripta fragmenti

```
<% @Page aspcompat=true %>
<html>
<%
    cn = Server.CreateObject("ADODB.Connection")
    cn.Open("MyFirm")
    sql = "SELECT Name, Surname, Duty FROM Employees"
    rs = Server.CreateObject("ADODB.Recordset")
    rs.Open(sql, cn, 3, 3)
    ...
    Response.Write(rs.Fields("Name").Value & " " & _
        rs.Fields("Surname").Value & " " & _
        rs.Fields("Duty").Value)
    ...
    rs.close
    rs=nothing
    cn.close
    cn=nothing
%>
</html>
```

Informācija par aktuālo datumu un laiku (C# valoda)

```
<% @Page Language="C#" %>
<script runat="server">
    protected String GetTime()
    {
        return DateTime.Now.ToString();
    }
</script>

<html>
    <body>
        <% ="Current data and time: " + GetTime() %>
    </body>
</html>
```

ASP un ASP.NET

1. Programmēšanas valodu atbalsts

ASP galvenokārt orientēts uz *divām* skripta valodām: JScript un VBScript.

Abos gadījumos runa ir par valodām bez *stingras tipizācijas* un *kompilācijas*.

.NET platforma atbalsta *vairākas* kompilējamas valodas ar stingro tipizāciju: C#, VB.NET, . . .

Valoda JScript.NET atbalsta stingro tipizāciju.

Komponentus var rakstīt vienā programmēšanas valodā, bet lietot citā valodā (tajā skaitā mantot no komponentiem).

2. Iekšēja tīmekļa lappušu struktūra

ASP tehnoloģijā koda fragmenti un HTML fragmenti *nav* atdalīti.

Tādā tekstā grūti orientēties, īpaši ja tīmekļa lappuses izstrādātāji zina tikai *skriptus*, tikai *datubāzes*, vai tikai *HTML*.

ASP.NET gadījumā programmas kods atdalīts no HTML teksta.

Kodu var arī izvietot citā failā.

Vairākos gadījumos var vispār atteikties no “klasiskās” programmēšanas (īpaši pēc ASP.NET 2.0).

3. Vadības elementu standartizācija

HTML: teksta ievadei ir *vairāki* elementi.

1. *Viena* teksta virkne

```
<input type="text" ... >
```

2. *Vairākas* teksta virknes

```
<textarea rows="3" cols="4" ... >
```

ASP skripti apstrādā šos elementus.

ASP.NET visos gadījumos izmanto *vienīgo* servera elementu

```
<asp:textbox runat="server" ... />
```

Elementa lietošanas režīmu regulē atribūti (viena virkne, vairākas virknes, parole, ...).

Uzdevums: pēc tīmekļa lappuses ielādes izvadīt informāciju teksta lodziņā Message.

1a. Kods valodā VB.NET

```
<script language="VB" runat="server">  
    Public Sub Page_Load(Sender As Object, _  
        E As EventArgs)  
        Message.Text = "Hello!"  
    End Sub  
</script>
```

1b. Kods valodā C#

```
<script language="c#" runat="server">  
    public void page_load(Object Sender,  
        EventArgs E) {  
        Message.Text = "Hello!";  
    }  
</script>
```


2a. Forma ar servera elementu ASP.NET stilā.

```
<form runat="server">  
    <asp:textbox runat="server" id="Message" />  
</form>
```

Pēc peles LP, View Source:

```
<input name="Message" type="text" value="Hello!"  
    id="Message" />
```

2b. Forma ar servera elementu HTML stilā.

```
<form runat="server">  
    <input type="text" runat="server"  
        id="Message" />  
</form>
```

Piezīme: šajā gadījumā skriptos izmanto atribūtu Value.

```
Message.Value = "Hello!"
```

Uzdevums: ievadīt informāciju reģistrēšanai uz serveri.

Izmantot *servera* vadības elementus.

Login:

1. Formas fragments.

```
<form name="User" runat="server">
```

Login:

```
<asp:textbox runat="server" id="Login"  
    text="User" />
```

...

```
<asp:button runat="server" id="Info"  
    text="Send" onclick="Info_Click"/>
```

```
</form>
```

2. Informācijas apstrāde.

```
<head>  
  <script language="vb" runat="server">  
    Public Sub Info_Click(Sender As Object, _  
      E As EventArgs)  
      Response.Write(Login.Text)  
    End Sub  
  </script>  
</head>
```

3. Var norādīt skripta valodu pašā sākumā:

```
<% @Page Language="VB" %>  
<html>  
  ...  
</html>
```

Piezīmes par ASP.NET vadības elementiem:

1. Visur ir aizveroša *slīpa svītra*. **Nepareizi:**

`<asp:button runat="server" ... >`

2. Simbolu reģistrs ir *patvaļīgs*. Ieteicams apakšējais reģistrs.

`<asp:button ... >`

`<ASP:Button ... >`

3. Nav atribūta *name*, bet ir atribūts *id*. **Nepareizi:**

`<asp:textbox runat="server" name="Login" ... />`

4. Nav atribūta *value*, bet ir atribūts *text*. **Nepareizi:**

`Response.Write(Login.Value)`

Piemērā var arī izmantot HTML elementu **<input>**.

```
<input type="text" runat="server" id="Login" value="User">
```

Bet servera elementos ir *standartizētie* īpašību vārdi.

Vadības elementi:

1. Deklarēti uz *serveri*.
 2. Programmēti *serverim*.
 3. Apstrādā *klienta* notikumus.
-

Formā netika norādīts apstrādes skripts (*Action*), kā arī informācijas nodošanas metode (*Method*).

Informācijas apstrāde notiek *aktuālajā* tīmekļa lappusē.

Apstrādātāja *Info_Click* parametri:

1. Vadības elements, kur bija notikums.
 2. Papildus parametri.
-

Piemērā izmantots „iegultā koda” (*code inline*) modelis.

Kods izvietots tīmekļa lappuses iekšā, bet ir atdalīts no HTML elementiem.

Alternatīva: „fona koda” (*code behind*) modelis.

Kods atdalīts no HTML iezīmēšanas un atrodas ārējā failā.

1. *Code behind* modelis: tīmekļa lappuse *ChkLogin.aspx*.

```
<% @Page
    Language="VB"
    Inherits="ChkLogin"
    Src="ChkLogin.vb"
%>

<html>
    ...
    <form runat="server">
        Login:<asp:textbox runat="server" id="Login"
            text="User" />
        ...
        <asp:button runat="server" id="Info"
            text="Send" onclick="Info_Click" />
    </form>
    ...
</html>
```

2. Code behind modelis: fails *ChkLogin.vb*.

```
Imports System
Imports System.Web.UI 'User Interface
Imports System.Web.UI.WebControls
Public Class ChkLogin
  Inherits System.Web.UI.Page
  Public Dim Login As TextBox

  Public Sub Info_Click(Sender As Object, _
    E As EventArgs)
    Response.Write(Login.Text)
  End Sub
End Class
```

Piezīme: vadības elementu *TextBox* no tīmekļa lappuses obligāti deklarē kā klases atribūtu.

Var arī mantot no klases *Page*, bez prefiksa.

Lai apstrādātāja kods uzrakstīts C# valodā.

1. Tīmekļa lappuses *.aspx sākums.

```
<%@ Page  
    Language="C#" Inherits="ChkLogin"  
    Src="ChkLogin.cs"  
%>
```

2. Apstrāde (C# fails *ChkLogin.cs*).

```
using System;  
using System.Web.UI;  
using System.Web.UI.WebControls;  
public class ChkLogin: System.Web.UI.Page {  
    public TextBox Login;  
    protected void Info_Click(object sender,  
        EventArgs e) {  
        Response.Write(Login.Text);  
    }  
}
```

Modernajās .NET versijās izmanto *daļējās* klases (*partial classes*)

1. Direktīvā `Page` **ne**izmanto atribūtu `Src`.
2. Failu ar kodu norāda atribūtā `CodeFile`.
3. Koda failu nosaukumos *ieteicams* orientēties uz diviem paplašinājumiem. Piemēram: `ChkLogin.aspx.cs`.
4. Koda failā klasi apraksta kā **`partial`**.
5. Vadības elementus (`Login`) atkārtoti **ne**deklarē.
Rezultātā tiks samazināts koda apjoms.

Visas deklarācijas tiks ģenerētas *automātiski*. Piemēram:

```
protected System.Web.UI.WebControls.TextBox Login;
```

Izmainītais piemērs ar daļējām klasēm

1. Tīmekļa lappuses *.aspx sākums.

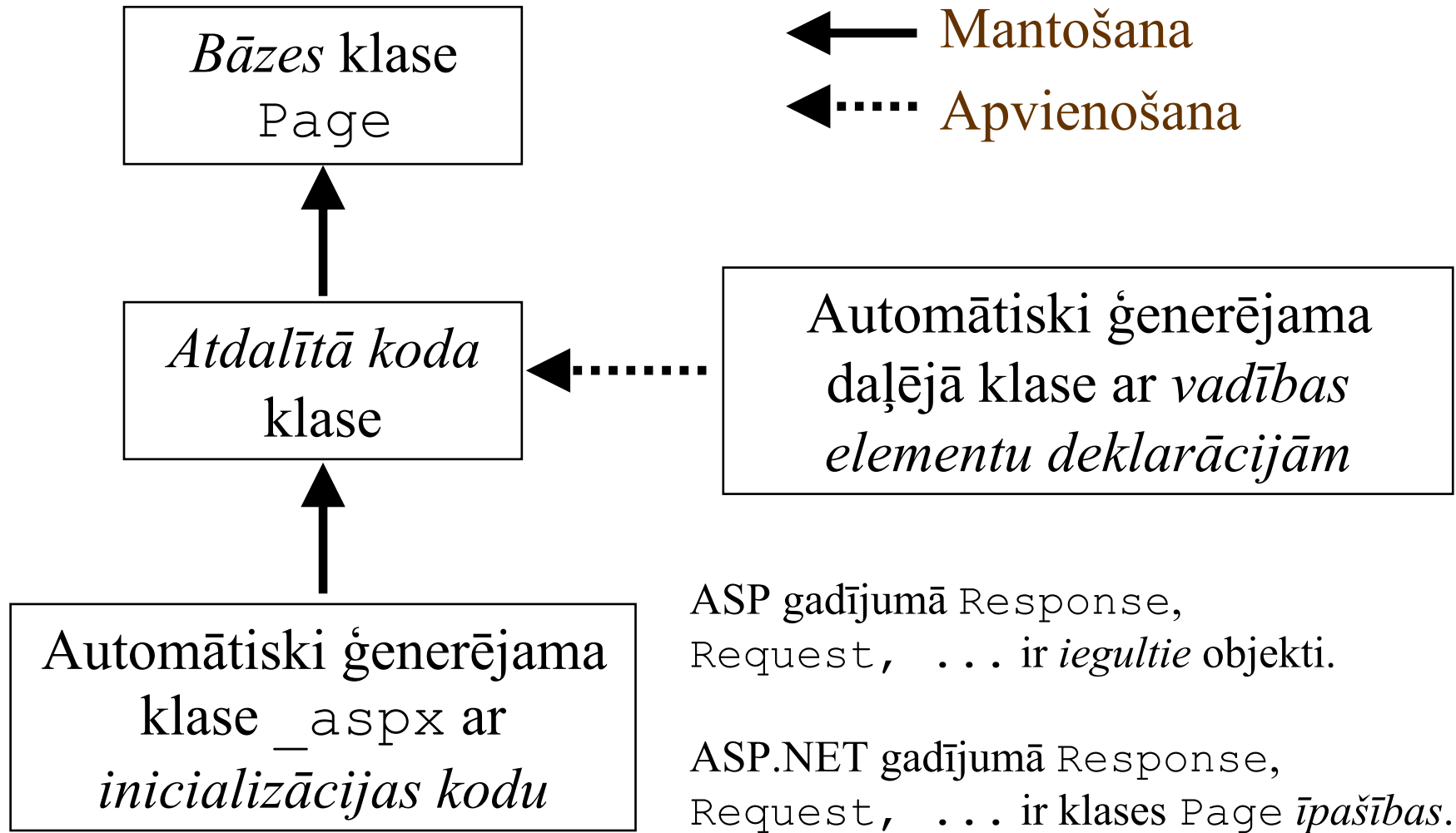
```
<% @Page  
    Language="C#" Inherits="ChkLogin"  
    CodeFile="ChkLogin.aspx.cs"  
%>
```

2. Fails ar C# kodu (ChkLogin.aspx.cs).

```
using System;  
...  
public partial class ChkLogin:  
    System.Web.UI.Page {  
    protected void Info_Click(object sender,  
        EventArgs e) {  
        Response.Write(Login.Text);  
    }  
}
```

Tīmekļa lappuses *konstruēšana*

← Mantošana
←..... Apvienošana



1. Klase Page no .NET klašu bibliotēkas noteic *bāzes funkcionālās iespējas*, kas atļauj:
 - a. apkalpot vadības elementus;
 - b. vizualizēt HTML kodu;
 - c. strādāt ar objektiem ASP stilā (Response, Request, ...).
 2. Klase ar atdalīto kodu *manto* 1. punktā minētās iespējas.
 3. Sistēma automātiski ģenerē kodu ar *aizsargātiem (protected) mainīgajiem* – vadības elementiem.
 4. Kompilators veido *rezultējošo* klasi ar sufiksu: piemēram, ChkLogin_aspx.
- Klasei pievienots inicializācijas kods un galīgā HTML iezīmēšana.

ASP.NET tehnoloģijas pamatā ir *servera* vadības elementi (*server controls*). Vizītkarte: atribūts **runat**="server".

ASP.NET lietošanas rezultāts: *kompilējamais* kods.

Visa tīmekļa lappuse (HTML kods, teksts un cita informācija) tiks kompilēta *klasē*. Ģenerējamas klases superklase: *System.Web.UI.Page*.

Dinamiski konstruētas *Page* klases objekti tiks izveidoti pieprasījuma laikā.

```
<%@ Page ... %>
```

Klases (objekta) izpildes rezultāts: izvaddati, kurus saņem klients.

Runā, ka ASP.NET **<form>** elementi realizē arhitektūru ar atgriešanu serverī (*postback architecture*).

Katrs servera vadības elements tiks kompilēts kā *globāli pieejamais* tīmekļa lappuses objekts. Rezultātā var adresēt elementu tekstu, sarakstu alternatīvas, utt.

Uzdevums: lai tīmekļa lappuses inicializācijas laikā tiks norādītas vērtības pēc noklusēšanas.

```
Sub Page_Load()  
    User.Text = "User"  
    Password.Text = "Password"  
End Sub
```

Piezīme: elementi *User* un *Password* tika noformēti kā:

```
<asp:textbox runat="server" id="User" />
```

Notikumu apstrāde ASP.NET tehnoloģijā

1. Pēc pirmās tīmekļa lappuses ielādes ASP.NET veido *objektu* no lappuses un vadības elementiem.
2. Izpildās inicializācijas kods. Tīmekļa lappusi pārveido HTML kodā un *atgriež klientam*. Lappuses objekti tiks izdzēsti no servera atmiņas.
3. Lietotājs sūta formas datus *serverim atpakaļ (postback)*. Kopā ar formas datiem serveris saņem visu lappusi.
4. ASP.NET saņem atgriezto lappusi un *atjauno* visus objektus (objektiem būs stāvoklis pirms pēdējās nosūtīšanas).

5. ASP.NET noteic operāciju, kas stimulēja atpakaļēju datu nosūtīšanu. Tiks ierosināts attiecīgais notikums (piemēram, `Send.Click`).

6. Tīmekļa lappuses izstrādātāju operācijas (servera datubāzes apstrāde, izmaiņas vadības elementos).

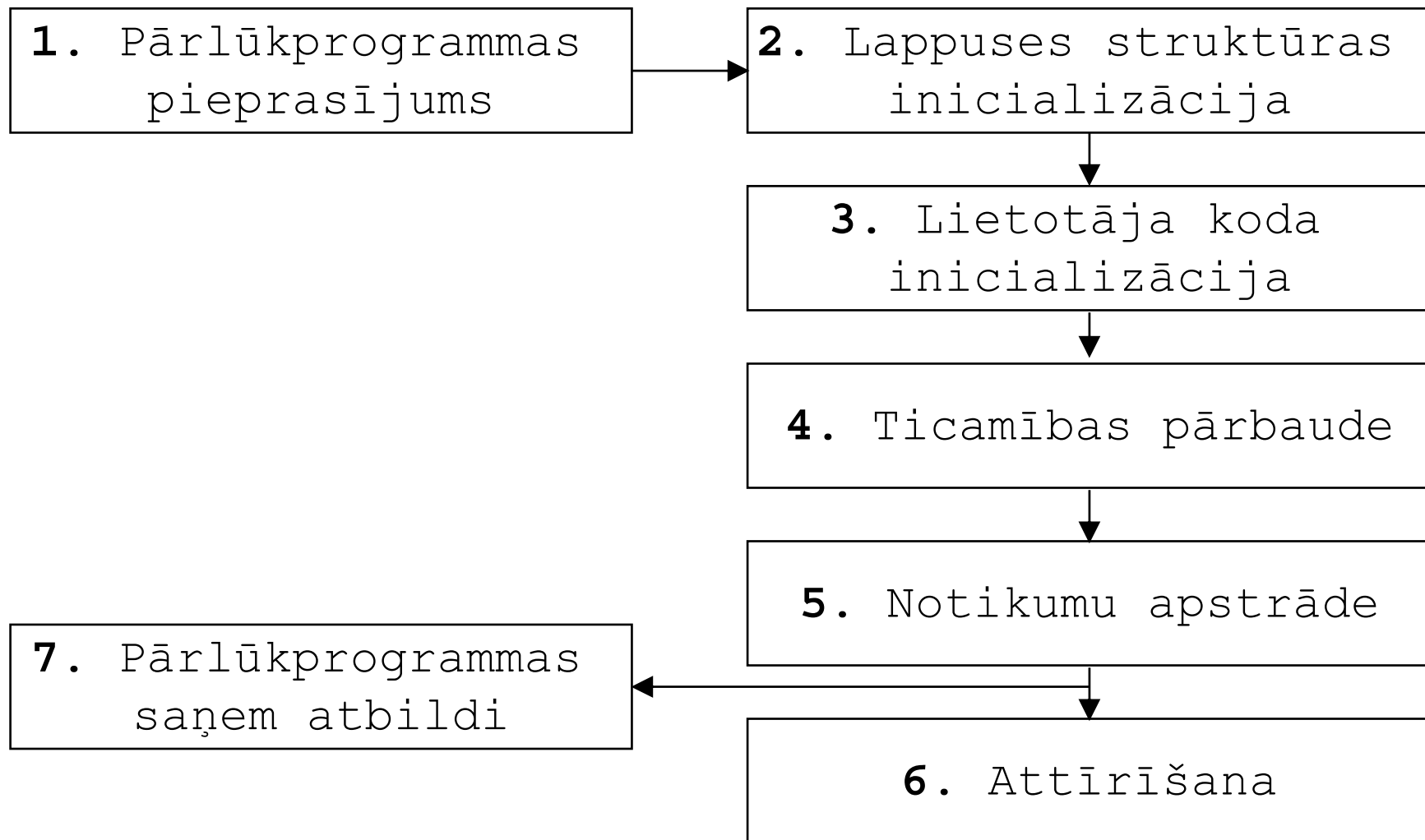
7. Izmainīto lappusi pārveido HTML kodā un atgriež klientam.

Pēc kārtējas pārsūtīšanas tiks atkārtoti soli (2) – (6).

Notikumi tiks ierosināti pēc *jebkurām* izmaiņām.

Lēmumu par attiecīgo reakciju pieņem lappuses izstrādātājs.

ASP.NET tīmekļa lappuses dzīves cikls



Dzīves cikla aprakstīšana

2. Tīmekļa lappuses radīšana, vadības elementu ģenerēšana. Notikums `Page.Init`.
3. Notikums `Page.Load`.
Notikums notiek *vienmēr*: pirmā ielāde vai sūtīšana atpakaļ.
4. Vadības elementi, kas automātiski pārbauda citus elementus.
Var arī pārbaudīt visu lappusi: `Page.IsValid`.
5. Tīmekļa lappuse pilnīgi ielādēta un pārbaudīta.
 - a. Notikumi, kas prasa *momentānu atbildi* (piemēram, `Submit`).
 - b. *Izmaiņu* notikumi. Izmaiņas teksta lodziņos, sarakstos.
6. HTML ģenerēšana. Notikums `Page.Unload`.

Tīmekļa lappuses *virsraksts*

Lai lappuses virsraksts *var mainīties* atkarībā no informācijas, kas tika nolasīta no datubāzes.

1. Elementu **<head>** *obligāti* deklarē kā servera elementu.
2. Kolekcija `Controls` atļauj *pievienot* metadatus.
3. Klase `HtmlMeta` atļauj *veidot* metadatus.

```
Page.Header.Title = "Programming"
```

```
Dim KW As HtmlMeta = New HtmlMeta()
```

```
KW.Name = "keywords"
```

```
KW.Content = "C++, Java, C#"
```

```
Page.Header.Controls.Add(KW)
```

```
<head runat="server" >  
</head>
```

Dinamisko vadības elementu radīšana

Uzdevums: izveidot dinamisko pogu “Calculate”.

```
Public Sub Calc_Click(Sender As Object, _  
    E As EventArgs)  
    Response.Write("Done.")  
End Sub  
  
Public Sub Page_Load(Sender As Object, _  
    E As EventArgs)  
    Dim CalcB As Button = New Button()  
    CalcB.Text = "Calculate"  
    CalcB.Id = "Calc"  
    AddHandler CalcB.Click, AddressOf Calc_Click  
    FormPanel.Controls.Add(CalcB)  
End Sub
```

Panelis pogas pievienošanai

```
<form runat="server">  
    <asp:panel runat="server" id="FormPanel"/>  
</form>
```

Vadības elementu *ieteicams* veidot notikumu apstrādātājā
Page_Load.

C# lietošana (fragmenti)

```
public void Calc_Click(object Sender,  
    EventArgs E) {  
    ...  
}  
...  
Button CalcB = new Button();  
CalcB.Text = "Calculate";  
CalcB.ID = "Calc";  
CalcB.Click += new EventHandler(Calc_Click);  
FormPanel.Controls.Add(CalcB);
```

Visi servera vadības elementi manto no klases *WebControl*.

Dažas īpašības:

`BackColor` = “fona krāsa”

`BorderColor` = „robežas krāsa”

`BorderWidth` = „robežas biezums”

`ForeColor` = "teksta krāsa"

`Height` = “augstums”

...

```
<asp:textbox runat="server"  
    BackColor="lightgray" BorderColor="red"  
    BorderWidth="5" ForeColor="Green" .../>
```

Uzdevums: izvadīt informāciju par vadības elementa stāvokli (vērtību) HTML elementā.

1. Vadības elements.

```
<asp:textbox runat="server"  
    id="Login" text="User" />
```

2. HTML elements **<div>**.

```
<div runat="server" id="Res"></div>
```

3. Apstrādātāja fragments.

```
Res.InnerHTML = Login.Text
```

Piezīme: elementā **<div>** obligāti jābūt atribūts **runat**.

Parastais teksta logs:

```
<asp:textbox runat="server"  
    TextMode="Singleline"...>
```

Uzdevums: izveidot ievades lauku ar norādītu rindiņu un kolonu daudzumu (HTML `<textarea>` ekvivalents).

```
<asp:textbox  
    runat="server"  
    TextMode="Multiline"  
    Rows="3" Columns="30"  
    id="Data" text="User"/>
```

Paroles ievade (lodziņš ar “zvaigznītēm”):

```
<asp:textbox runat="server"  
    TextMode="Password".../>
```

Vadības elements **<asp:label>**

Elementa radīšana:

```
<asp:label runat="server" id="Res"  
text="Result:" />
```

Elementa adresēšana:

```
Res.Text = Data.Text
```

Elements ir elementa **** analogs.

Elements **<div>** automātiski pārnes tekstu uz nākamo rindiņu.

'Result:' ir teksts ekrānā.

Dinamiskā interfeisa radīšana

Uzdevums: dinamiski noformēt rezultāta izvadi.

```
<% @Page Language="VB" %>
<% @Import Namespace="System.Drawing" %>
...
<%
    Dim Counter=0
    ...
    Result.ForeColor = Color.yellow
    Result.BackColor = Color.black
    Result.Text = " Result: " & Counter
%>
...
<asp:label runat="server" id="Result"/>
```

Attiecīgo ASP.NET kodu var rakstīt *jebkurā* vietā.
Rezultātā visu svarīgo kodu var izvietot *vienā* vietā.
Tas īpaši svarīgi *lielos* projektos.

Alternatīvais risinājums DHTML stilā

```
Dim Res
```

```
Dim Counter=0
```

```
...
```

```
Res = "<span style=" & _  
    "'color:yellow;background-color:black'>" & _  
    "Result: " & Counter & "</span>"
```

```
Response.Write(Res)
```

Nepieciešams domāt par izvades operatora izvietošanu.

Vadības elements **<asp:LinkButton>**

Ieplānotais rezultāts:

Login: [Send](#)

```
<asp:LinkButton runat="server" text="Send"
onclick="Info_Click"/>
```

Vadības elements **<asp:ImageButton>**

Ieplānotais rezultāts:

Login: 

```
<asp:ImageButton runat="server"
ImageURL="Send.jpg" AlternateText="Send"
onclick="Info_Click"/>
```

Piezīme: notikuma apstrādātājam būs citi parametri.

```
Public Sub Info_Click  
    (Sender As Object, E As ImageClickEventArgs)
```

Elements **<asp:Image>**

```
<asp:Image runat="server" id="i"  
    ImageURL="Foto.jpg" AlternateText="Foto"... />
```

Piezīme: šeit un citos gadījumos jābūt atribūts *id*.

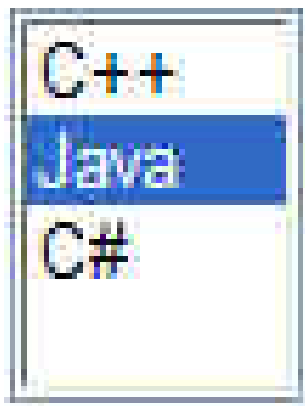
Elements **<asp:HyperLink>**

```
<asp:HyperLink runat="server" id="d"  
    NavigateURL="Dtls.htm" Text="Details" />
```

Rezultāts: teksta hipersaite *Details* uz tīmekļa lappusi *Dtls.htm*.

Vadības elements `<asp:ListBox>`

Lai sarakstā ir trīs programmēšanas valodas: *C++*, *Java*, *C#*.



```
<asp:ListBox runat="server" id="Langs">  
  <asp:ListItem text="C++" value="cpp"/>  
  <asp:ListItem text="Java" value="java"  
    selected="True"/>  
  <asp:ListItem text="C#" value="cs"/>  
</asp:ListBox>
```

Rezultāta saņemšana (*cpp*, *java* vai *cs*):

```
Res.InnerHTML = Langs.Text
```

Piezīme: `ListItem` nav servera vadības elementi.

Līdz ar to atribūts `runat` nav vajadzīgs.

Var neizmantot atribūtu **text**:

```
<asp:ListItem value="cpp">C++</asp:ListItem>
```

```
<asp:ListItem value="java"  
    selected="True">Java</asp:ListItem>
```

```
<asp:ListItem value="cs">C#</asp:ListItem>
```

Lai ir iespēja norādīt vairākas valodas:

```
<asp:ListBox runat="server" id="Langs"  
    rows="2" SelectionMode="Multiple" >
```

...

```
</asp:ListBox>
```

Piezīme: tiks redzamas *tikai divas* saraksta rindiņas.

Iegūto rezultātu apstrāde:

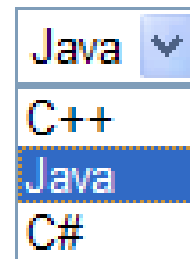
```
Public Sub Info_Click  
    (Sender As Object, E As EventArgs)  
    Dim S As String, Item As ListItem  
    S = "Result: "  
    For Each Item In Langs.Items  
        If Item.Selected Then  
            S = S & " " & Item.Text  
        End If  
    Next  
    Res.InnerHTML = S  
End Sub
```

Piezīme: pēc noklusēšanas **SelectionMode**="Single".

Vadības elements `<asp:DropDownList>`

Iepriekšēja uzdevuma izpilde:

```
<asp:DropDownList runat="server" id="Langs">  
    <asp:ListItem text="C++" value="cpp" />  
    ...  
</asp:DropDownList>
```



Izvēlētais elements: *Langs.Text*.

Izvēlēta elementa indekss: *Langs.SelectedIndex* (0, 1, ...).

Ja nav norādīta atribūta `value` vērtība, par rezultātu būs `text` vērtība.

```
<asp:ListItem text="C++" />
```

Rezultāts: *C++*.

Vadības elements `<asp:CheckBox>`

Norādīt *C++*, *Java* vai *C#*:

```
<asp:CheckBox runat="server" id="cpp"
  text="C++" />
```

```
<asp:CheckBox runat="server" id="java"
  text="Java" checked />
```

```
<asp:CheckBox runat="server" id="cs"
  text="C#" />
```



Rezultāta analīze:

```
Res.innerHTML = cpp.Checked & " " & java.Checked
& " " & cs.Checked
```

Iezīmju izlīdzināšana: `TextAlign= {"Left", "Right"}.`

```
<asp:CheckBox ... TextAlign="Left" />
```

Vadības elements `<asp:CheckBoxList>`

Norādīt *C++*, *Java* vai *C#*:



```
<asp:CheckBoxList runat="server"
  id="Langs">
```

```
  <asp:ListItem text="C++" />
```

```
  <asp:ListItem text="Java" selected />
```

```
  <asp:ListItem text="C#" />
```

```
</asp:CheckBoxList>
```

Izvēles rūtiņu saraksta apstrādei var izmantot parasta saraksta apstrādes kodu (*SelectionMode="Multiple"*).

Sarakstu var izvietot horizontāli:

```
<asp:CheckBoxList ... RepeatDirection="Horizontal">
```

Cita iespēja apstrādāt sarakstu:

```
S = "Result: "  
For i=0 To Langs.Items.Count-1  
    If Langs.Items(i).Selected Then  
        S = S & " " & Langs.Items(i).Text  
    End If  
Next  
Res.InnerHTML = S
```

Vadības elements **<asp:RadioButton>**

```
<asp:RadioButton runat="server" id="cpp"  
    text="C++" />  
<asp:RadioButton runat="server" id="java"  
    text="Java" />  
<asp:RadioButton runat="server" id="cs"  
    text="C#" />
```

Ir *četri* vadības elementu veidi.

1. *Sarakstu* elementi.

Gandrīz visi vadības elementi ar nosaukuma daļu `...List`.

Izņēmums: `DataList`.

Tie ir pašpietiekamie vadības elementi, tos viegli atkārtoti izmantot.

Elementi gandrīz nav saistīti ar tīmekļa lappusi, galvenais – kods, kuru izmanto datu iegūšanai.

Nevar izmainīt elementu vizualizāciju.

Kādas iespējas nodrošina CSS un HTML formatēšana, bet tās ir “ārējas” iespējas.

2. *Iteratīvie* elementi.

Elements apstrādā saistītā datu kopuma elementus un pielieto pie tiem ASP.NET šablonu.

Ir divi iteratīvie elementi: `Repeater` un `DataList`.

Šablons ir iezīmēšanas koda fragments. Faktiski, tā ir forma ar vietām datu izvadei. Formu iebūvē vadības elementā.

Rezultātā elements ir “elastīgs” un nodrošina vairākas iespējas. Strādāt ar tādiem elementiem sarežģītāk.

Problemātiski izmantot šo kodu atkārtoti.

Šablona informācija visbiežāk atrodas aktuālajā tīmekļa lappusē.

3. *Tabulas* elementi.

Datu izvade lappusēs, informācijas kārtošana.

Elementi `GridView`, `DataGrid`.

Būtībā, runa ir par iteratīvo elementu specializētām versijām.

4. *Ierakstu* vizualizēšanas elementi.

Elementi `DetailsView`, `FormView`.

Atšķirība no tabulas elementiem: darbs ar atsevišķiem ierakstiem.

Dažreiz (3) un (4) grupu sauc: reprezentēšanas elementi.

ASP .NET versijas konfigurēšana

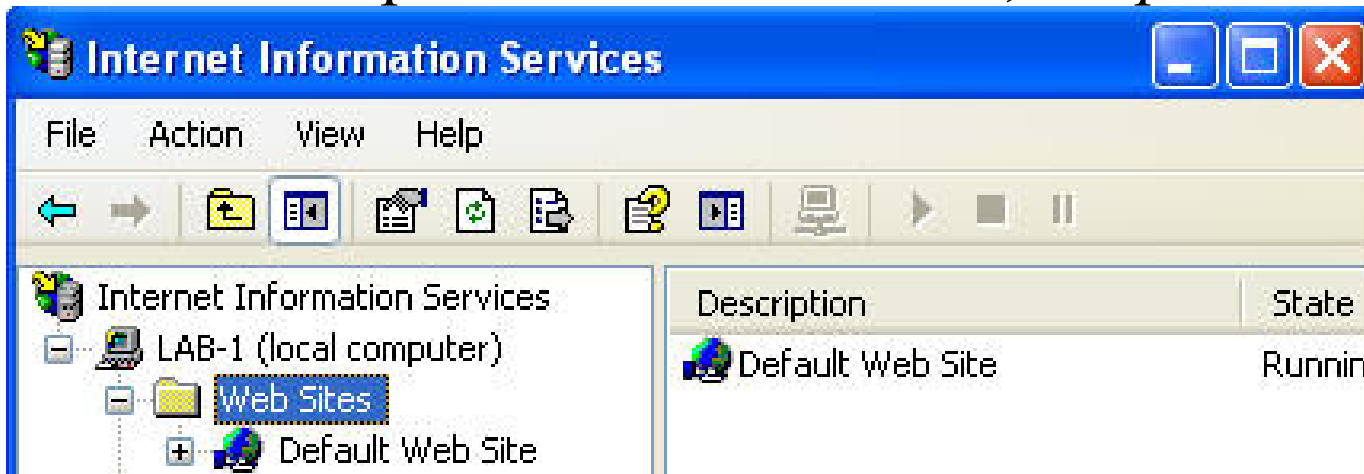
1. Pārlūkprogramma: versija “pēc noklusēšanas” (1.1.432...).

Version Information: Microsoft .NET Framework Version:1.1.4322.2407; ASP.NET Version:1.1.4322.2407

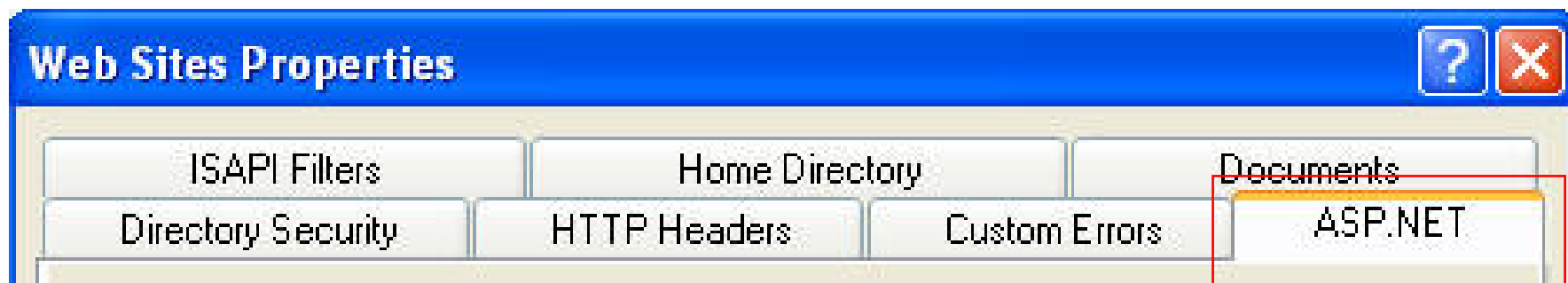
2. *Control Panel, Internet Information Services.*



3. Iezīmēt mapi *Web Sites*. Peles LP, *Properties*.



4. Izvēlēties ielikumu ASP.NET.



5. Norādīt jaunu ASP.NET versiju.



6. Pārlūkprogramma: jauna versija (2.0.507...).

Version Information: Microsoft .NET Framework Version:2.0.50727.1433; ASP.NET Version:2.0.50727.1433

Uzdevums: iegūt informāciju par firmas darbinieku vārdiem.

```
<% @Page Language="VB" Debug="True" %>
<!-- Vārdu telpu importēšana -->
<% @Import Namespace="System.Data" %>
<% @Import Namespace="System.Data.OleDb" %>
```

Skripts tīmekļa lappuses **<head>** daļā:

```
Sub Page_Load(Source As Object, E As EventArgs)
    ' Mainīgo deklarēšana
    Dim Cn As OleDbConnection
    Dim Cmd As OleDbCommand
    Dim Rd As OleDbDataReader
    Dim openstr, SQL As String
```

Piekluve datubāzei:

```
openstr = _
    "Provider=Microsoft.Jet.OLEDB.4.0;" & _
    "Data Source=" & Server.MapPath ("firm.mdb")
```

```
` Savienojuma atvēršana
Cn = New OleDbConnection(openstr)
Cn.Open()

SQL = "SELECT * FROM Employees"

` SQL vaicājuma izpilde
Cmd = New OleDbCommand(SQL, cn)
Rd = Cmd.ExecuteReader()

` Datu iegūšana
` Names ir kombinētais saraksts
Names.DataSource = Rd
Names.DataBind()

Rd.Close()
Cn.Close()
```

End Sub

```
<form runat="server">  
  <!-- Elementa Names deklarēšana -->  
  <asp:ListBox runat="server" id="Names"  
    DataTextField="Name"  
    DataValueField="Name" />  
</form>
```

Rezultāts:



Piezīme: darbā ar *MS SQL Server* izmanto citus objektus:

```
<% @Import Namespace="System.Data.SqlClient" %>
...
Dim Cn As SqlConnection
Dim Cmd As SqlCommand
Dim Rd As SqlDataReader
```

Kā redzams, *OleDb* vietā ir *Sql*.

Turpmāk runa būs par jēdzieniem *Connection*, *Command*, *DataReader* un citiem.

Objekti *OleDbConnection* un *SqlConnection* ir līdzīgi ADO objektam *Connection*. Populārās metodes:

1. *Open()*. Atvērt savienojumu ar datu avotu.
2. *Close()*. Aizvērt savienojumu ar datu avotu.
3. *BeginTransaction()*. Uzsākt transakciju.

ADO: objekti *Connection* (*Command*) un *Recordset*.

ADO.NET: objekti *Connection* (*Command*) un *DataReader* (*DataSet*).

Objekts *RecordSet* neeksistē.

DataReader nodrošina datu lasīšanu un pārvietošanu vienā virzienā.

Datubāzes modifikācija: abos gadījumos objekti *Connection* un *Command*.

Tālāk: SQL operatoru vai glabājamās procedūras izpilde.

OleDB... pielieto ar OLE-DB.

Sql... izmanto *Tabular Data Services* (TDS) ar MS SQL Server.

Informācijas iegūšana ciklā:

```
Dim Res As String
```

```
Dim i As Integer
```

```
...
```

```
Rd = Cmd.ExecuteReader()
```

```
i=0
```

```
Do While Rd.Read()
```

```
    i = i + 1
```

```
    Res &= i & ". " & Rd("Name") & " " & _  
        Rd("Surname") & " " & Rd("Duty") & "<br>"
```

```
Loop
```

```
Rd.Close()
```

```
Cn.Close()
```

```
Response.Write(Res)
```

Rezultāts:

1. Jānis Strods programmētājs
2. Sergejs Ivanovs operators
3. Uldis Zemzars operators

Datu avoti ADO un ADO.NET arhitektūrās

ADO: *vispārināta* objektu kopa.

Visām datubāzēm (Oracle, SQLServer, ...) izmanto vienu un to pašu klasi `Connection`.

ADO.NET: *datu piegādātāju* (data provider) modelis.

Datu piegādātāji ir klases, kas nodrošina piekļuvi datubāzei, SQL komandu izpildi un datu iegūšanu.

`Connection` – savienojums ar datubāzi.

`Command` – SQL komandu un glabājamo procedūru izpilde.

`DataReader` – ātra piekļuve datiem tikai lasīšanai.

`DataAdapter` – savstarpēja iedarbība ar `DataSet`.

Katrs piegādātājs nodrošina specifisko realizāciju *visām* četrām augstākminētām klasēm.

Tas atļauj nodrošināt ADO.NET *paplašināmību*.

.NET ietvars saistīts ar *četriem* kopumiem no piegādātājiem.

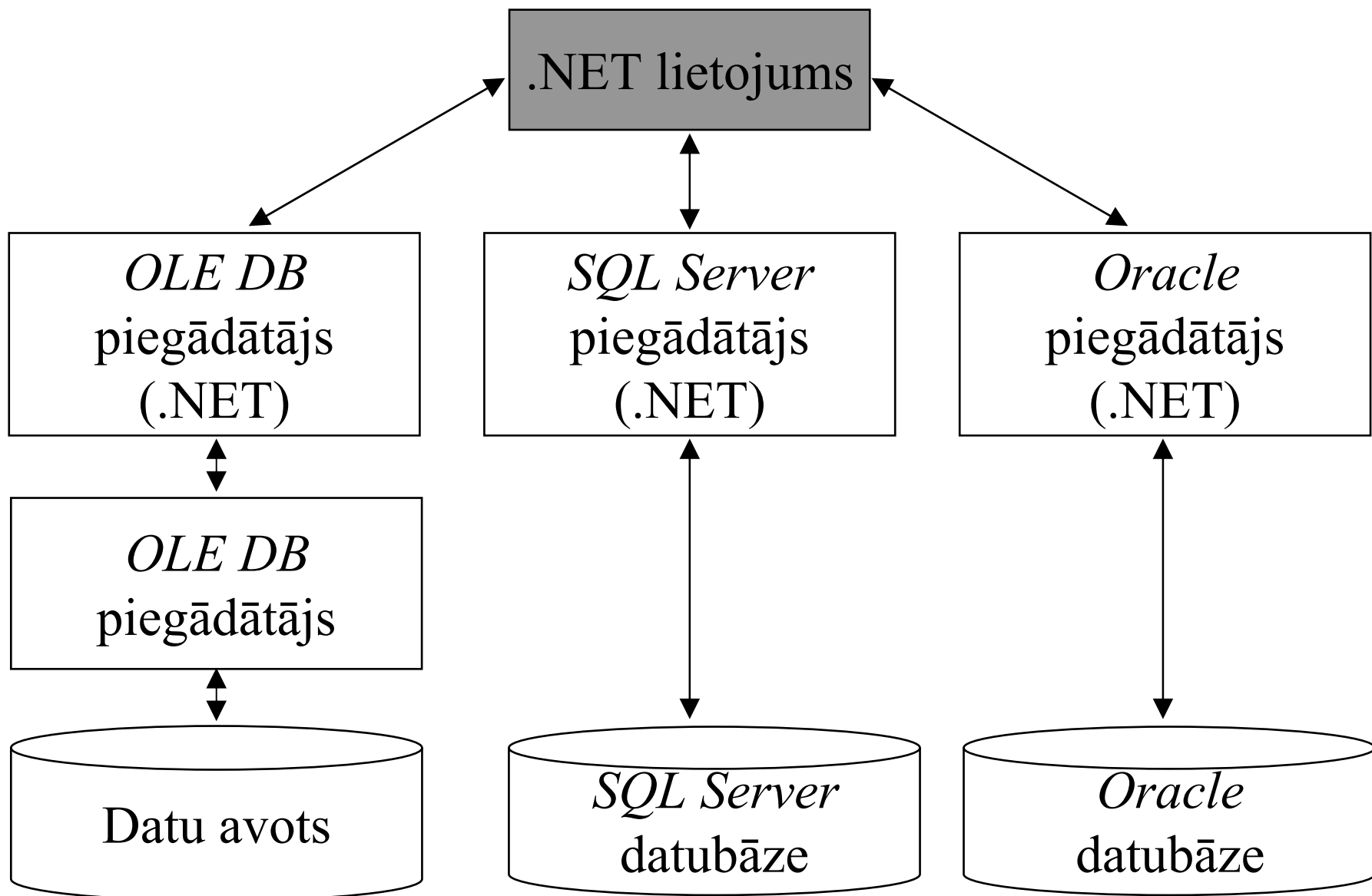
Visos gadījumos nodrošināta optimizēta piekļuve datubāzēm.

1. *OLE DB*. Piekļuve *jebkuram* datu avotam ar OLE DB draiveri, tajā skaitā vecām SQL versijām (līdz 7.0).

2. *SQLServer* (7.0 un augstāk).

3. *Oracle*.

4. *ODBC*. Piekļuve *jebkuram* datu avotam ar ODBC draiveri.



OLE DB ir ADO sastāvdaļa un to aktīvi izmanto.

Gandrīz visiem datu avotiem paredzēti OLE DB draiveri (SQL Server, Oracle, MS Access, MySQL, ...).

Ieteicams izmantot *specializēto* piegādātāju.

Faktiski, *piegādātājs* ir četrus ADO.NET klašu kopums: `Connection`, `Command`, `DataReader`, `DataAdapter`.

Visi piegādātāji kādā veidā *standartizē* savas klases.

Piemēram, `Connection` realizē interfeisu `IDbConnection`.

Šis interfeiss noteic metodes `Open()` un `Close()`.

ADO.NET vārdu telpas

1. `System.Data`. Konteineru klases. Tabulas, kolonnas, ...
2. `System.Data.OleDb`. Piekļuve *OLE DB* piegādātājam.
3. `System.Data.SqlClient`. Piekļuve *Microsoft SQL Server* datubāzei.
4. `System.Data.OracleClient`. Piekļuve *Oracle* datubāzei.
5. `System.Data.Odbc`. Piekļuve pie ODBC draiveriem.
6. `System.Data.Common`. Dažas abstraktās klases.
7. `System.Data.SqlTypes`. SQL Server struktūru atbalsts.

Uzdevums: iegūt informāciju par firmas darbinieku vārdiem, lietojot ODBC.

Piezīme: parādītas galvenokārt jaunas (izmainītas) rindiņas.

```
<% @Import Namespace="System.Data.Odbc" %>
```

```
Sub Page_Load(Source As Object, E As EventArgs)
```

```
    ' Mainīgo deklarēšana
```

```
    Dim Cn As OdbcConnection
```

```
    Dim Cmd As OdbcCommand
```

```
    Dim Rd As OdbcDataReader
```

```
    ...
```

Piekluve datubāzei:

```
openstr = _
```

```
"Driver={Microsoft Access Driver (*.mdb)};" & _
```

```
"DBQ=" & Server.MapPath("firm.mdb")
```

```
Cn = New OdbcConnection(openstr)
Cn.Open()

SQL = "SELECT * FROM Employees"
Cmd = New OdbcCommand(SQL, Cn)
Rd = Cmd.ExecuteReader()

Names.DataSource = Rd

...

Rd.Close()
Cn.Close()
```

Piezīme: šajā gadījumā savienojuma rinda *pilnīgi sakrīt* ar savienojumu rindu, kas tika izmantota ASP tehnoloģijā.

Uzdevums: iegūt informāciju par firmas darbinieku vārdiem, lietojot ODBC. Programmēšanas valoda: C#.

Piezīme: visos gadījumos jāraksta *tikai* `Odbc` .

```
<% @Import Namespace="System.Data.Odbc" %>
```

```
protected void Page_Load(Object Source,
    EventArgs E) {
    OdbcConnection Cn;
    OdbcCommand Cmd;
    OdbcDataReader Rd;

    string openstr, SQL;

    openstr = "driver={Microsoft Access Driver" +
        " (*.mdb) };" +
        "dbq=" + Server.MapPath("firm.mdb");
```



```
Cn = new OdbcConnection(openstr);  
Cn.Open();  
SQL = "SELECT * FROM Employees";  
Cmd = new OdbcCommand(SQL, Cn);  
Rd = Cmd.ExecuteReader();  
  
Names.DataSource = Rd;  
Names.DataBind();  
  
Rd.Close();  
Cn.Close();
```

Piezīme: pārējās metodēs arī nepieciešams ievērot reģistru.

Open(), Close(), ExecuteReader(), ...

Ir iespēja *dinamiski* veidot piekļuves rindiņu.

Tādiem nolūkiem lieto klasi `OdbcConnectionStringBuilder`.

```
OdbcConnectionStringBuilder CSB = new  
    OdbcConnectionStringBuilder();  
CSB.Add("Dbq", Server.MapPath("firm.mdb"));  
CSB.Add("Uid", "");  
CSB.Add("Pwd", "");  
openstr = CSB.ConnectionString;
```

Piezīme: mūsu piemērā `Uid` un `Pwd` inicializācija nebija obligātā.

Ir iespējama cita sintakse:

```
CSB["Dbq"] = Server.MapPath("firm.mdb");  
CSB["Uid"] = "";  
CSB["Pwd"] = "";
```

To pašu pieeju var realizēt, strādājot ar OleDb...

Izmanto klasi OleDbConnectionStringBuilder.

```
Dim CSB As New OleDbConnectionStringBuilder()  
CSB.Add("Data Source", Server.MapPath("firm.mdb"))  
CSB.Add("Provider", "Microsoft.Jet.OLEDB.4.0")  
CSB.Add("Jet OLEDB:Database Password", "")
```

Ir iespējama cita sintakse:

```
CSB("Data Source") = Server.MapPath("firm.mdb")  
CSB("Provider") = "Microsoft.Jet.OLEDB.4.0"  
CSB("Jet OLEDB:Database Password") = ""
```

Ir vairākas citas metodes. Piemēram, savienojuma attīrīšana.

```
CSB.Clear()
```

Pieslēgšanai var izmantot konfigurācijas failu `web.config`.
Tad nevajag izvietot savienojuma rindiņu programmas kodā.
Tas ir *ieteicamais* stils.

Datubāzes pārvietošanas gadījumā nevajag pārkompilēt vairākus koda fragmentus (analoģija ar DSN lietošanu ASP).

Faila *fragments*:

```
<configuration>
```

```
...
```

```
<connectionStrings>
```

```
<add name="FirmConnection"
```

```
  connectionString="Provider=Microsoft.Jet.
```

```
    OLEDB.4.0;Data Source=d:\InetPub\WWWRoot
```

```
    \MyHost\firm.mdb;
```

```
    User ID=;Password=;"/>
```

```
</connectionStrings>
```

```
</configuration>
```

Programmā izmanto *divas* klases:

1. `ConnectionStringSettings`.

Savienojuma rindas parametri.

2. `ConfigurationManager`.

Piekluve konfigurācijas failam (tajā skaitā savienojuma rindas nolasīšana).

Skripta fragments:

```
Dim CStrSet As ConnectionStringSettings
```

```
CStrSet =  
    ConfigurationManager.ConnectionStrings  
        ("FirmConnection")
```

```
openstr = CStrSet.ConnectionString
```

Ir papildu iespējas (piemēram, informācija par savienojumu).

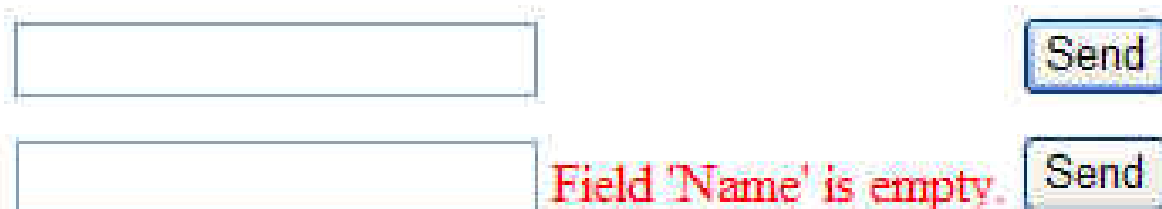
```
Response.Write(CStrSet.Name) 'FirmConnection
```

Papildu komentāri par `DataReader` lietošanu.

1. Piekļuve datiem bez attālās apstrādes (nav atslēgta savienojuma).
2. Tas ir kursors - “*firehouse*” (ugunsdzēsēja šļūtene): tiešais savienojums, visātrākā piekļuve un datu iegūšana.
3. To bieži lieto **SELECT** operatora izpildei.
4. To var lietot **INSERT**, **UPDATE**, **DELETE** operatoru izpildei (pateicoties `Command`).
5. To var lietot darbā ar glabājamām procedūrām (pateicoties `Command`).
6. To lietderīgi izmantot darbā ar lieliem datu apjomiem.

Datu pareizības pārbaude

Uzdevums: teksta laukā jābūt kāda informācija.



Field 'Name' is empty.

Teksta logs un satura kontrole:

```
<asp:TextBox runat="server" id="Name"/>
```

```
<asp:RequiredFieldValidator
```

```
    runat="server" id="ValidName"
```

```
    ControlToValidate="Name"
```

```
    ErrorMessage="Field 'Name' is empty."
```

```
    ForeColor="Red"/>
```

Piezīme: piemērā tika rezervēta brīva telpa iespējamam ziņojumam (attālums starp teksta lauku un pogu).

Var atteikties no rezervēšanas:

```
<asp:RequiredFieldValidator... Display="Dynamic"/>
```

Rezultāts: elements **** ar stilu **display:none**.

Elementa izmērs netiks pieņemts vērā formas radīšanas procesā.

Pēc noklusēšanas ir vērtība *Static*.

```
<asp:RequiredFieldValidator... Display="Static"/>
```

Rezultāts: elements **** ar stilu
visibility:hidden.

Uzdevums: pārbaudīt ievadītu vecumu. Vecums jābūt diapazonā 18..70.

```
<asp:TextBox runat="server" id="Age" />
```

```
<asp:RangeValidator
```

```
    runat="server" id="Val_Age"
```

```
    ControlToValidate="Age"
```

```
    MinimumValue="18"
```

```
    MaximumValue="70"
```

```
    Type="Integer"
```

```
    EnableClientScript="false"
```

```
    Text="Age must be in range 18 to 70"
```

```
    ForeColor="Red" />
```

Uzdevums: pārbaudīt ievadītu vecumu. Vecums jābūt vismaz 18 gadi.

```
<asp:Textbox runat="server" id="Age" />
```

```
<asp:CompareValidator
```

```
    runat="server" id="Val_Age"
```

```
    ControlToValidate="Age"
```

```
    Operator="GreaterThanOrEqualTo"
```

```
    ValueToCompare="18"
```

```
    Type="Integer"
```

```
    EnableClientScript="false"
```

```
    Text="Age must be at least 18 years"
```

```
    ForeColor="Red"
```

```
/>
```

Uzdevums: pārbaudīt visas ievadītas informācijas pareizību.

Lai ir teksta logi *Name* un *Surname* kopā ar attiecīgo pareizības pārbaudi.

```
<asp:RequiredFieldValidator  
    runat="server" Display="None" InitialValue=""  
    ControlToValidate="Surname"  
    ErrorMessage="Field 'Surname' is empty.".../>  
...
```

Rezultējošā pārbaude:

```
<asp:ValidationSummary  
    runat="server" id="VS"  
    ShowSummary="false"  
    ShowMessageBox="true"  
    HeaderText="Errors:"/>
```

Lai abi teksta logi ir tukši. Rezultāts:



Lai atribūts *ShowSummary* netika izmantots. Rezultāts: papildus ziņojums ekrānā.

Surname:

Errors:

- Field 'Name' is empty.
- Field 'Surname' is empty.

Uzdevums: ievadīt teksta lodziņā nepārskaitli.

1. ASP vadības elementi.

```
<asp:textbox runat="server" id="Odd" />  
<asp:CustomValidator  
    runat="server"  
    ControlToValidate="Odd"  
    Display="Dynamic"  
    OnServerValidate="CheckOdd"  
    ErrorMessage="Please enter ODD number" />
```

Piezīme: datu pareizības pārbaude notiek *servera* pusē.

2. Servera skripts.

```
Sub CheckOdd(Src As Object,  
    e As ServerValidateEventArgs)
```

2. Servera skripts (turpinājums).

```
e.IsValid = False  
Try  
    Dim x As Integer = Convert.ToInt32(e.Value)  
    If x Mod 2 <> 0 Then  
        e.IsValid = True  
    End If  
    Catch ex As Exception  
    End Try  
End Sub
```

Piezīmes: pēc noklusēšanas informācija satur kļūdu.

e.IsValid – pārbaudes rezultāts.

Ja lodziņā nav skaitļa, tiks ierosināts izņēmums.

3. Iespējama pārbaude klienta pusē.

Vadības elements:

<asp:CustomValidator

```
...  
ClientValidationFunction="CheckOdd"  
.../>
```

Skripts (*JavaScript*):

<script>

```
function CheckOdd(src, e) {  
    e.IsValid = (e.Value == parseInt(e.Value)) &&  
                (e.Value % 2) != 0  
}
```

</script>

Vadības elementu saturu var kontrolēt ar *regulāro izteiksmju* palīdzību (*Regular Expressions*).

To atļauj validators **RegularExpressionValidator**.

Uzdevums: pārbaudīt elektroniskā pasta adreses pareizību.

Adresē *obligāti* jābūt simboli ‘@’ un ‘.’; atstarpju nav.

```
<asp:TextBox id="Email" runat="server"/>
```

```
<asp:RegularExpressionValidator
```

```
  id="EmailREValid" runat="server"
```

```
  ControlToValidate="Email"
```

```
  ValidationExpression="\S+@\S+\.\S+"
```

```
  Display="Dynamic" Text="*" />
```

Piezīmes: \S ir jebkurš simbols, izņemot atstarpi.

+ ir kvantifikators – *vismaz* viena sakritība (a@b.cd).

Visiem validatoriem ir viena superklase: **BaseValidator**.

Vārdu telpa: `System.Web.UI.MobileControls`.

Svarīgie atribūti:

1. **ControlToValidate** = ID

Pārbaudāmais vadības elements.

2. **EnableClientScript** = {**true**, **false**}

Pareizības pārbaude *klienta* pusē.

3. **SetFocusOnError** = {**true**, **false**}

Pozicionēšana elementā *ar nepareizo* informāciju (ASP.NET 2.0).

Fokuss pēc noklusēšanas: nosūtīšanas poga (līdzīgais elements).

Lai ir vairāki vadības elementi ar vērtību `true`.

Tad notiek pozicionēšana pirmajā elementā.

4. **Enabled** = {**true**, false}

Validatoru funkciju pieejamība.

Validatoru var *dinamiski* pieslēgt/atslēgt skriptā.

5. **Text** = "kļūda"

Ziņojums par kļūdu *blakus* ar vadības elementu.

6. **ErrorMessage** = "kļūda"

Ziņojums elementam **ValidationSummary** (ja tas eksistē).

Ja atribūta **Text** nav, būs efekts atribūtam **ErrorMessage**.

7. **Display** = {**Dynamic**, Static, None}

Ziņojuma par kļūdu pozicionēšana blakus ar vadības elementu.

Ziņojumu var arī atslēgt (None).

Uzdevums: pārbaudīt ievadītā vecuma pareizību ar vairāku validatoru palīdzību (tajā skaitā ar *lappuses validatoru*).

Ieplānotie rezultāti:

1. Forma. 2. *Tukšais* lodziņš/nav nosūtīts.
3. Lodziņā *nav prasāmā skaitļa* /nav nosūtīts.
4. Lodziņā *viss ir kārtībā* /nosūtīts.

Enter your age (at least 7):

Submit

(1)

Enter your age (at least 7): *

Submit

Not submitted.

(2)

• Enter a value.

Enter your age (at least 7): *

Submit

Not submitted.

(3)

• Enter at least 7.

Enter your age (at least 7):

Submit

Submit OK!

(4)

1. *Tīmekļa lappuses pareizības kontrole.*

Reakcija uz nosūtīšanas pogas nospiešanu.

```
<script runat="server">  
    Sub SubmitBtn_Click(ByVal sender As Object,  
        ByVal e As EventArgs)  
        If Page.IsValid Then  
            SubmitResultLbl.Text = "Submit OK!"  
        Else  
            SubmitResultLbl.Text = "Not submitted."  
        End If  
    End Sub  
</script>
```

2. Forma servera pusē un teksta lodziņš.

```
<form runat="server">  
    Enter your age (at least 7):  
    <asp:textbox id="Age" runat="server"/>
```

3. Validators *tukšā lodziņa* pārbaudei.

```
<asp:RequiredFieldValidator  
    id="AgeRqrValid" runat="server"  
    ControlToValidate="Age"  
    EnableClientScript="False"  
    Display="Dynamic"  
    ErrorMessage="Enter a value." Text="*" />
```

4. Validators *vērtības* pārbaudei.

```
<asp:CompareValidator  
    id="AgeCmpValid" runat="server"  
    ControlToValidate="Age"  
    EnableClientScript="False" Display="Dynamic"  
    Type="Integer" Operator="GreaterThanOrEqualTo"  
    ValueToCompare="7"  
    ErrorMessage="Enter at least 7." Text="*" />
```

5. *Poga* informācijas nosūtīšanai.

```
<asp:button  
    id="SubmitBtn" runat="server"  
    Text="Submit" onclick="SubmitBtn_Click"/>
```

6. *Iezīme* nosūtīšanas rezultāta izvadei.

```
<asp:label id="SubmitResultLbl" runat="server"/>
```

7. *Summārais* validators.

```
<asp:ValidationSummary id="Errors" runat="server"/>
```

Ja nav summārā validatora, kļūdu gadījumā tiks redzamas tikai *sarkanās “zvaigznītes”*.

Pats summārais validators **neizpilda** nekādu pārbaudi.

Bez atribūta **Text** ziņojums **ErrorMessage** būs *divās vietās*.

Lai vienā formā ir *vairākas* pogas.

Pēc *kādas* pogas nospiešanas tiks pārbaudīti tikai *konkrētie* vadības elementi (nevajag aktivizēt *visus* validatorus).

Uzdevums: forma sastāv no *divām* loģiskām daļām.

Pēc pogas `Submit` nospiešanas uz servera puses tiks apstrādātā formas daļa ar *informāciju par lietotāju*.

Pēc pogas `Define Status` nospiešanas tiks izpildītas papildu operācijas, bet to rezultāts var būt interesants tikai pašam lietotājam.

Uzdevuma izpildei tiks izmantoti parastie validatori ar atribūtu **ValidationGroup**.

```
<form runat="server">
```

```
...
```

```
Age: <asp:textbox id="Age" runat="server"/>
```

```
...
```

```
<asp:RequiredFieldValidator
```

```
    id="AgeRqrValid" runat="server"
```

```
    ControlToValidate="Age"
```

```
    Display="Dynamic" Text="*"
```

```
    ValidationGroup="ClientGroup" />
```

```
...
```

```
<asp:button
```

```
    id="SubmitBtn" runat="server"
```

```
    Text="Submit"
```

```
    ValidationGroup="ClientGroup" />
```

```
...
```



```
Status: <asp:textbox id="Status" runat="server"/>
```

```
...
```

```
<asp:RequiredFieldValidator
```

```
  id="StatusRqRValid" runat="server"
```

```
  ControlToValidate="Status"
```

```
  Display="Dynamic" Text="*" 
```

```
  ValidationGroup="StatusGroup" />
```

```
...
```

```
<asp:button
```

```
  id="StatusBtn" runat="server"
```

```
  Text="Define Status"
```

```
  ValidationGroup="StatusGroup" />
```

Piezīme: lai kādai pogai *nav* norādīta vadības elementu grupa. Tad tiks apstrādāti tikai elementi *bez* kādas grupas.

Elements `<asp:bulletedlist>` (ASP.NET 2.0)

Programmēšanas valodu saraksts:

- Java
- C#

```
<asp:bulletedlist runat="server"
    BulletStyle="square">
    <asp:listitem Text="Java"></asp:listitem>
    <asp:listitem>C#</asp:listitem>
</asp:bulletedlist>
```

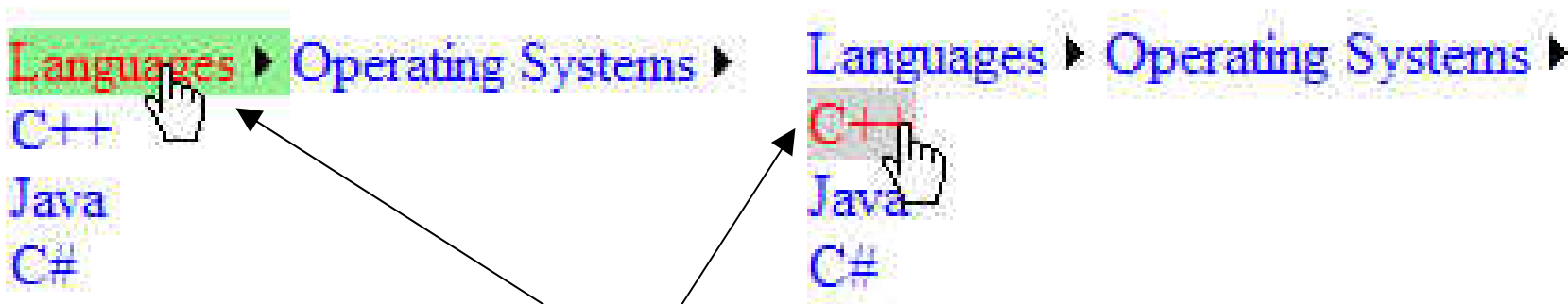
Elements `<asp:panel>` (ASP.NET 2.0)

```
<asp:panel runat="server"
    Height="100pt" Width="200pt"
    Scrollbars="vertical"
    style="background-color:gray">
    <asp:label runat="server">C#</asp:label>
</asp:panel>
```



Izvēlnes veidošana (ASP 2.0)

Uzdevums: izveidot hierarhisko izvēlni programmatūras izvēlei.



Izvēlnē tiks izmantoti stili. Tad:

```
<head runat="server">
</head>
```

Piezīme: izvēlni izviesto formas iekšā.

```
<form runat="server">
...
</form>
```

Izvēlnes aprakstīšana:

```
<asp:Menu id="Soft" runat="server"
  Orientation="Horizontal">

  <StaticHoverStyle BackColor="LightGreen"
    ForeColor="Red" />

  <DynamicHoverStyle BackColor="LightGray"
    ForeColor="Red" />

  <Items>
    <asp:MenuItem Text="Languages">
      <asp:MenuItem Text="C++"
        NavigateURL="cpp.aspx" />
      <asp:MenuItem Text="Java"
        NavigateURL="Java.aspx" />
      <asp:MenuItem Text="C#"
        NavigateURL="cs.aspx" />
    </asp:MenuItem>
```

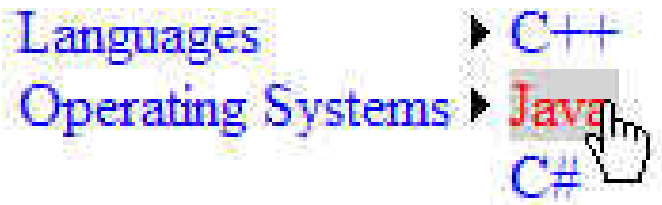
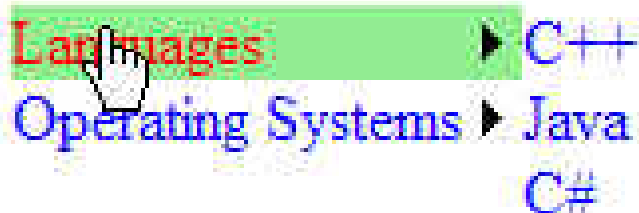
Izvēlnes aprakstīšana (turpinājums)

```

<asp:MenuItem Text="Operating Systems">
    <asp:MenuItem Text="Windows"
        NavigateURL="win.aspx" />
    <asp:MenuItem Text="Unix"
        NavigateURL="unix.aspx" />
    <asp:MenuItem Text="Linux"
        NavigateURL="linux.aspx" />
</asp:MenuItem>
</Items>
</asp:Menu>

```

Izvēlne ar vertikālo orientāciju (*Orientation="Vertical"*):



ASP.NET 2.0 nodrošina sasaistīšanu ar datiem *bez koda palīdzības*.

Attiecīgais kods paslēpts ASP.NET platformā.

Datu avotu identificē pēc vārda.

Vairākos vadības elementos ir jauns atribūts *DataSourceId*.

Datu atlases komandu norāda atribūtā *SelectCommand*.

Sasaistīšanu var izpildīt ar vairākiem objektiem, tajā skaitā ar XML dokumentiem un MS Excel lapām.

Microsoft Access datubāzu apstrāde (ASP.NET 2.0)

Uzdevums: izvadīt informāciju par darbinieku vārdiem un uzvārdiem. Izmantot *tikai vadības elementus*.

```
<form runat="server">
  <asp:AccessDataSource
    runat="server" id="Firm"
    DataFile="Firm.mdb"
    SelectCommand = "SELECT * FROM Employees"/>
  <asp:Repeater runat="server" id="R"
    DataSourceId="Firm">
    <ItemTemplate>
      <%# Eval("Name") %>
      <%# Eval("Surname") %> <br>
    </ItemTemplate>
  </asp:Repeater>
</form>
```

Jānis Strods
 Sergejs Ivanovs
 Uldis Zemzars

Elementu `AccessDataSource` var izmantot arī *citām* operācijām ar datiem.

1. `DeleteCommand` – dzēšana.
2. `InsertCommand` – ielikšana.
3. `UpdateCommand` – atjaunināšana.

Elementu `AccessDataSource` izdevīgi izmantot darbā ar vadības elementiem `GridView`, `FormView`.

Elementu **ne**var izmantot, ja *MS Access* datubāzei ir parole. Jāizmanto `SqlDataSource`.

```
<asp:SqlDataSource
    id="Firm" runat="server"
    providerName="System.Data.OleDb"
    connectionString="Provider=Microsoft.Jet.OLEDB.4.0
;Data Source=D:\Inetpub\...\firm.mdb"
    SelectCommand = "SELECT * FROM Employees"/>
```


Uzdevums: pieslēgt datubāzi ar web.config palīdzību.

1. web.config saturs.

```
<connectionStrings>
```

```
  <add name="FirmConnection"
```

```
    connectionString="Provider=Microsoft.Jet.OLEDB.
```

```
4.0;Data Source=d:\InetPub\...\firm.mdb;"
```

```
    providerName="System.Data.OleDb"/>
```

```
</connectionStrings>
```

2. Tīmekļa lappuses fragments.

```
<asp:SqlDataSource
```

```
  id="Firm" runat="server"
```

```
  connectionString =
```

```
    "<%$ ConnectionStrings:FirmConnection %>"
```

```
  providerName=
```

```
    "<%$ ConnectionStrings:FirmConnection.
      ProviderName %>"
```

```
  SelectCommand = "SELECT * FROM Employees"/>
```

Uzdevums: pieslēgt datubāzi ar ODBC palīdzību.

Lai jau eksistē datu avots *System DSN* ar vārdu VEF.

```
<asp:SqlDataSource  
  id="Firm"  runat="server"  
  ProviderName = "System.Data.Odbc"  
  ConnectionString = "dsn=VEF"  
  SelectCommand = "SELECT * FROM Employees"/>
```

AccessDataSource vadības elementā **neizmanto** ConnectionString. To vietā ir DataFile.

Līdz ar to nav iespējas pieslēgties datubāzei ar paroli, jo savienojuma parametrus norāda atribūtā ConnectionString.

AccessDataSource elements orientēts tikai uz vienu piegādātāju: System.Data.OleDb.NET.

Uzdevums: iegūt tabulu ar *Repeater* palīdzību.

```
<asp:Repeater runat="server" id="R"
    DataSourceId="Firm">
    <HeaderTemplate>
        <table border="2" cellpadding="3">
    </HeaderTemplate>
    <ItemTemplate>
        <tr>
            <td>
                <%# DataBinder.Eval(Container.DataItem,
                    "Name") %>
            </td>
            <td>
                <%# DataBinder.Eval(Container.DataItem,
                    "Surname") %>
            </td>
        </tr>
    </ItemTemplate>
</asp:Repeater>
```

Jānis	Strods
Sergejs	Ivanovs
Uldis	Zemzars

```
        </td>
    </tr>
</ItemTemplate>

<FooterTemplate>
    </table>
</FooterTemplate>

</asp:Repeater>
```

Piezīme: to pašu rezultātu var iegūt, **nelietojot** DataBinder.

```
<tr>
    <td>
        <%# Eval ("Name") %>
    </td>
    <td>
        <%# Eval ("Surname") %>
    </td>
</tr>
```

To pašu uzdevumu var izpildīt ar *GridView* palīdzību:

```
<asp:AccessDataSource.../>
```

```
<asp:GridView
```

```
    runat="server" id="R"
```

```
    DataSourceId="Firm"
```

```
    AutoGenerateColumns="true"/>
```

Rezultāts:

Name	Surname	Duty	Salary
Jānis	Strods	programmētājs	1100
Sergejs	Ivanovs	operators	900
Uldis	Zemzars	operators	850

Interfeisi datubāzes apstrādē

Iepriekšējos piemēros var izmantot interfeisus:

```
Dim Cn As IDbConnection
```

```
Dim Cmd As IDbCommand
```

```
Dim Rd As IDataReader
```

Pārējie operatori netiks izmainīti (runa ir par *OleDbConnection* un *OleDbCommand*).

```
Cn = New OleDbConnection(openstr)
```

Neder kods:

```
Cn = New IDbConnection(openstr)
```

Nevar izveidot interfeisa piemēru.

Uzdevums: attēlot datubāzes saturu tabulas formā.

```
Cn = New OleDbConnection(openstr)
```

```
Cn.Open()
```

```
SQL = "SELECT * FROM Employees"
```

```
Cmd = Cn.CreateCommand()
```

```
Cmd.CommandText = SQL
```

```
Rd = Cmd.ExecuteReader()
```

```
Data.DataSource = Rd
```

```
Data.DataBind()
```

```
<asp:DataGrid id="Data" runat="server"/>
```

Name	Surname	Duty	Salary
Jānis	Strods	programmētājs	1100
Sergejs	Ivanovs	operators	900
Uldis	Zemzars	operators	850

Piezīme: par datu avotiem var būt vairākas kolekcijas, kas nav saistītas ar ADO.NET.

Piemēram, var izmantot *ArrayList*, *SortedList*, *Queue*,...

Saraksta veidošana:

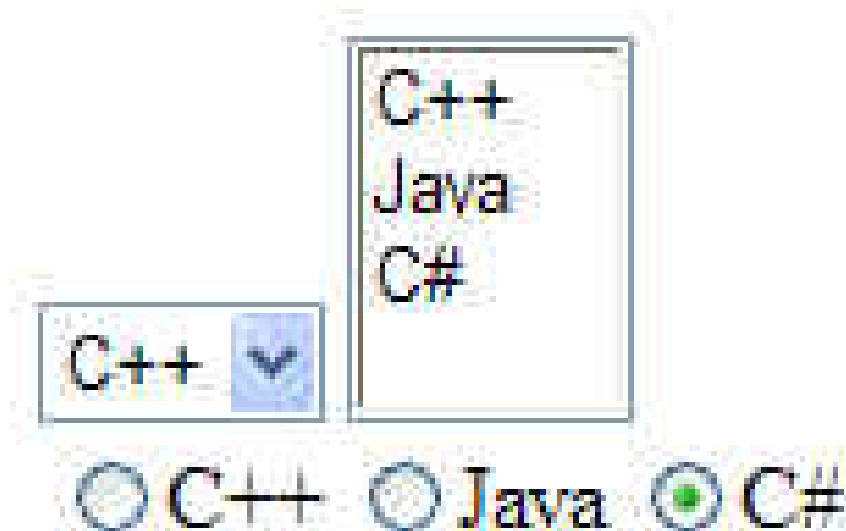
```
Dim Langs As ArrayList = New ArrayList()  
Langs.Add("C++")  
Langs.Add("Java")  
Langs.Add("C#")
```

Datu sasaistīšana:

```
DDL.DataSource = Langs `DropDownList  
LB.DataSource = Langs `ListBox  
RBL.DataSource = Langs `RadioButtonList  
DataBind()
```


ASP vadības elementi

```
<form runat="server">  
  <asp:DropDownList runat="server" id="DDL"/>  
  <asp:ListBox runat="server" id="LB"/>  
  <asp:RadioButtonList runat="server" id="RBL"  
    RepeatDirection="Horizontal"/>  
</form>
```



Objekts DataSet

Uzdevums: izveidot sarakstu no darbinieku vārdiem.

```
Dim Cn As OleDbConnection
```

```
Dim Da As OleDbDataAdapter
```

```
Dim Ds As DataSet
```

```
Dim openstr, SQL As String
```

```
Cn = New OleDbConnection(openstr)
```

```
SQL = "SELECT * FROM Employees"
```

```
Da = New OleDbDataAdapter(SQL, Cn)
```

```
Ds = New DataSet()
```

```
Da.Fill(Ds, "Employees")
```

```
LB.DataSource = Ds  
LB.DataTextField = "Name"  
DataBind()
```

ASP vadības elements

```
<form runat="server">  
    <asp:ListBox runat="server" id="LB"/>  
</form>
```

Galvenā atšķirība starp *DataSet* un *DataReader*: *DataSet* veido lokālu datu *kopiju*.

Programmā netika izmantots operators `Cn.Close()`.

Savienojums tika **automātiski** pārtraukts pēc operatora `Da.Fill(ds, "Employees")`.

Uzdevums: izvadīt informāciju par darbiniekiem, kuru vārds sākas ar burtu 'J'.

```
Dim Dv As DataView
Dv = New DataView(Ds.Tables("Employees"))

Dv.RowFilter = "Name like 'J%'"
Dv.Sort = "Name"

DDL.DataSource = Dv

DDL.DataTextField = "Name"

DataBind()
```

ASP vadības elements

```
<form runat="server">
    <asp:DropDownList runat="server" id="DDL"/>
</form>
```

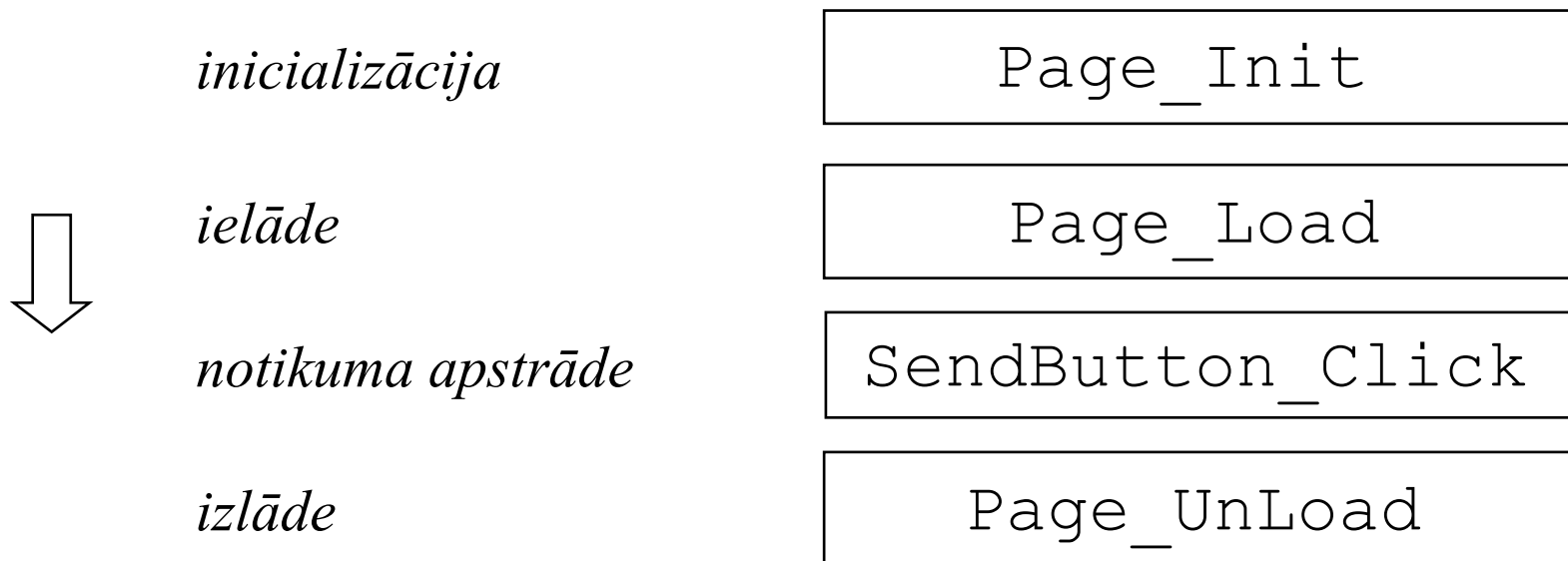
Vienā tīmekļa lappusē var būt *tikai viena* forma ar atribūtu `runat="server"`. **Nepareizi:**

```
<form id="f1" runat="server"></form>
```

```
<form id="f2" runat="server"></form>
```

Pēc noklusējuma atribūta `method` vērtība ir `post`.

Formas dzīves cikls (notikums un apstrādātājs):



Lai ir skripta fragments:

```
Sub Page_Init(Sender As Object, e As EventArgs)  
    Response.Write("Init.")
```

```
End Sub
```

```
Sub Page_Load(Sender As Object, e As EventArgs)  
    Response.Write("Load.")
```

```
End Sub
```

```
Sub Page_PreRender(Sender As Object,  
    e As EventArgs)  
    Response.Write("PreRender.")
```

```
End Sub
```

Rezultāts:

Init.Load.PreRender.

Notikums *PreRender* tiks iniciēts pēc visu servera notikumu iniciēšanas, bet pirms HTML faila saņemšanas klienta pusē.

Visos gadījumos sistēma automātiski meklē Page_... metodes.

Var atteikties no tādas shēmas:

```
<% @Page ... AutoEventWireup="false" %>
```

AutoEventWireup – automātiska notikumu sasaistīšana.

Iepriekšējā piemērā visas Page_... metodes netiks izpildītas.

Koda lapas numurs (pozitīvais vesels skaitlis):

```
<% @ ... CodePage="1257" %>
```

Izmantojama kodēšanas sistēma:

```
<% @ ... ResponseEncoding="Unicode" %>
```

Trasēšanas ieslēgšana:

```
<% @ ... Trace="true" %>
```

Trasēšanas rezultāti

Request Details

Session Id: dihw4jvlfvfczi45fbclnsnr
Time of Request: 02.11.2008 21:32:54
Request Encoding: Unicode (UTF-8)

Request Type: GET
Status Code: 200
Response Encoding: Unicode

Trace Information

Category	Message
aspx.page	Begin PreInit
aspx.page	End PreInit
aspx.page	Begin Init
aspx.page	End Init

Kompilācijas aizliegšana:

```
<% @Page ... CompilationMode="never" %>
```

Tas palielina *ātrdarbību*.

Piezīme: tādās tīmekļa lappusēs *nevar eksistēt skripti*.

Pēc noklusējuma: `Always` - kompilēt vienmēr.

Ir iespējams variants `Auto`: atkarībā no lappuses satura.

Obligātā mainīgo deklarācija *Visual Basic* skriptā:

```
<% @Page ... Explicit="True" %>
```

Pēc noklusējuma: `False`.

Citas valodas *ignorē* šo atribūtu.

Savienojuma kļūdas apstrāde: izņēmums *OleDbConnection*.

Try

```
cn = New OleDbConnection(openstr)
```

Catch Exc **As** OleDbException

```
Response.Write("Connection error.")
```

```
Response.Write(Exc.Message)
```

End Try

Iespējamie rezultāti:

```
Connection error.Could not find installable ISAM.
```

```
Connection error.Could not find file...
```

Piezīme: **n**eder operators:

Catch E **As** OleDbException `ir parametrs E

Vairāku izņēmumu apstrāde (*VB .NET*)

Try

...

Catch Exc **As** OleDbException

Response.Write("Connection error.")

Catch Exc **As** Exception

Response.Write("Unknown error.")

Finally

Response.Write("Done.")

End Try

Izņēmums *Exception* tiks ierosināts citu kļūdu gadījumā (piemēram, ja blokā notiks dalīšana ar 0).

Bloks *Finally* tiks izpildīts vienmēr.

Mūsu gadījumā ziņojums *Done.* tiks izvadīts visos gadījumos.

Klases radīšana VB .NET valodā

Class Human

Private Dim _Name **As String**

Private Dim _Surname **As String**

`Konstruktors bez parametriem

Public Sub New()

 _Name = ""

 _Surname = ""

End Sub

`Konstruktors ar parametriem

Public Sub New(Name **As String**,

 Surname **As String**)

Me._Name = Name

Me._Surname = Surname

End Sub

```
`Informācijas izvades metode
Public Overrides Function ToString() As String
    Return "Name:" & _Name & _
        ", surname:" & _Surname
End Function

`Īpašības radišana
Public Property Name() As String
    Get
        Return _Name
    End Get
    Set
        _Name = Value
    End Set
End Property
End Class

Dim H As New Human("Uldis", "Strods")
H.Name="Juris"
Response.Write(H)
```

Apakšklases radīšana

```
Class Employee
```

```
  Inherits Human
```

```
  Private Dim _Salary As Double
```

```
  Public Sub New()
```

```
    MyBase.New()
```

```
    _Salary = 0
```

```
  End Sub
```

```
  Public Sub New(Name As String, _
```

```
    Surname As String, Salary As Double)
```

```
    MyBase.New(Name, Surname)
```

```
    Me._Salary = Salary
```

```
  End Sub
```

```

Public Overrides Function ToString() As String
    Return MyBase.ToString() & _
        ", salary:" & _Salary
End Function

```

```

End Class

```

```

Dim Emp As New Employee("Ivars", "Celms", 700)
Response.Write(Emp)

```

Interfeisa radīšana:

```

Interface IHuman
    Property Name() As String
End Interface

```

Interfeisa realizēšana:

```

Class Human
    Implements IHuman
    ...
    Public Property Name() As String _
        Implements IHuman.Name

```

VB.NET vairs neatbalsta tipu *Variant*. **Nepareizi:**

```
Dim O As Variant
```

Jāizmanto datu tips *Object*:

```
Dim O As Object
```

```
O = New Human()
```

```
Response.Write( CType(O, Human) )
```

Rezervēto vārdu **Empty** un **Null** vietā lieto vārdu **Nothing**.

Eksistē saīsināta operatoru forma

```
Dim i As Integer = 1
```

```
i += 1
```

```
Dim S As String = "Hello"
```

```
S &= ", World!"
```

Nepareizi:

```
i++
```


Izmaiņas valodā JScript.NET (vismaz JScript 8.0 versija)

```
<% @Page Language="JScript" %>
```

```
<% @Page Language="Javascript" %>
```

Datiem ir tipi:

```
function Page_Load (Sender: Object, E: EventArgs) {
    var S : String
    S = "Hello."
    Response.Write (S)
}
```

Daži citi datu tipi:

```
var i:int, s:short, d:double, f:float
```

```
var sb: sbyte, ui: uint, us: ushort
```

Tipu pārveidošana notiek automātiski:

```
S = i    // String ← int
```

```
S += 2   // 12
```

Klases radīšana valodā JScript.NET

// Klases un atribūtu deklarācija

```
class Human {  
    private var _Name : String  
    private var _Surname : String  
  
    // Konstruktors pēc noklusēšanas  
    public function Human() {  
        _Name=""  
        _Surname=""  
    }  
  
    // Konstruktors ar parametriem  
    public function Human(Name:String,  
        Surname:String) {  
        _Name=Name  
        _Surname=Surname  
    }  
}
```

```
// Piekļuve vārdam: īpašība get()
function get Name() {
    return _Name;
}

// Piekļuve vārdam: īpašība set()
function set Name(Name : String) {
    _Name = Name;
}

// Informācijas par objektu izvade
public override function ToString() {
    return "Name:" + _Name +
        " Surname:" + _Surname
}

}
```

Piezīme: **n**eder **t**oString().

Nav tādas pārdefinējamas metodes.

Programmas fragments

```
// Objektu radīšana
var H1 = new Human(),
    H2 = new Human("Uldis", "Strods")

// Informācijas par objektiem izvade
Response.Write(H1)      //Name: Surname:
Response.Write(H2)      //Name:Uldis Surname:Strods

// Darbs ar īpašībām
H2.Name="Aldis"
Response.Write(H2.Name) //Aldis
Response.Write(H2)      //Name:Aldis Surname:Strods
```

Piezīme: objektu (mainīgo) deklarācija ir *obligāta*.

```
H3 = new Human() // kļūda: objekts nav deklarēts
i=1               // kļūda: mainīgais nav deklarēts
```

Apakšklases radīšana valodā JScript.NET

// Klases un atribūta deklarēšana

```
class Employee extends Human {  
    private var _Salary : double
```

//apakšklases konstruktors

```
public function Employee (Name:String,  
    Surname:String, Salary:double) {
```

```
    super (Name, Surname)
```

```
    this._Salary = Salary
```

```
}
```

//informācijas izvade

```
public override function ToString() {
```

```
    return super.ToString() +
```

```
        " Salary:" + _Salary
```

```
}
```

```
}
```

Piezīmes:

1. Apakšklases konstruktorā var *neizsaukt* superklases konstruktoru.

Tas atšķir situāciju no *Java* vai *C#* valodām

```
//apakšklases konstruktors  
public function Employee (Name:String,  
    Surname:String, Salary:double) {  
    this.Name=Name  
    this.Surname=Surname  
    this._Salary=Salary  
}
```

2. Apakšklases metodes *nevar* strādāt ar *privātiem* superklases atribūtiem. Tika izmantotas īpašības get/set.

3. Ir arī piekļuves specifikators **protected**.

Vārdnīcas radīšana lietojot *paplašināšanu*

```
expando class Dictionary {  
    public override function ToString() : String {  
        var S : String = ""  
        for (var Key : String in this)  
            S += Key + " : " + this[Key] + "<br>"  
        return S  
    }  
}
```

Programmas fragments:

```
var D : Dictionary  
D = new Dictionary()  
D["B"] = "Bold"  
D["I"] = "Italic"  
Response.Write(D)
```

Piezīme: **neder** atribūtu adresēšana ar punktu.

Objekta pārbaude uz *piederību klasei*

```
var H:Human = new Human(),  
    HE:Human = new Employee(),  
    E:Employee = new Employee()  
Response.Write(H instanceof Human)           //True  
Response.Write(H instanceof Employee)         //False  
Response.Write(HE instanceof Human)           //True  
Response.Write(HE instanceof Employee)         //True  
Response.Write(E instanceof Human)            //True  
Response.Write(E instanceof Employee)         //True
```

Statiskie klases locekļi. Konstantes.

```
class Ph {  
    public static const G = 9.8  
}  
...  
Response.Write(Ph.G) //9,8
```


Uzskaitāmie tipi

```
enum Colors {Red = 1, Green = 2, Blue = 3}  
var C1:Colors = Colors.Red  
var C2:Colors = "Blue"  
Response.Write(C1 + " " + int(C1)) //Red 1
```

Ja pārveidošana nav iespējama, notiek izņēmums

```
var C3:Colors  
try {  
    C3 = "Orange"  
}  
catch (ArgumentException) {  
    Response.Write("Color doesn't exist")  
}
```

Vērtības eksistēšanas pārbaude:

```
if (Enum.IsDefined(Colors, "Orange")) {
```

Vērtības *pārveidošana*

```
var C4 = Enum.Parse(Colors, "Red")
```

Visu iespējamo vērtību izvade

```
for(var i : int in Enum.GetValues(Colors))  
    Response.Write(Enum.GetValues(Colors).  
        GetValue(i) + " ")
```

Cits izvades variants:

```
for(var i : int in Enum.GetNames(Colors))  
    Response.Write(Enum.GetNames(Colors).  
        GetValue(i) + " ");
```

Klases un interfeisi:

```
interface DataControl {  
    ...  
}  
class CheckForm implements DataControl {  
    ...  
}
```

Notikumu apstrādātāju radīšana

1. Speciāla atribūta lietošana: `onEvent`, kur `Event` ir notikuma nosaukums.

```
<asp:button ... onclick="Send"/>
```

2. Darbs ar delegātu formā *EventHandler*.

```
Sub Send(Sender As Object, E As EventArgs)
```

```
...
```

```
End Sub
```

```
Sub Page_Init(Sender As Object, E As EventArgs)
```

```
    AddHandler SndBtn.Click, _
```

```
        New EventHandler(AddressOf Send)
```

```
End Sub
```

```
<asp:button id="SndBtn" runat="server"
```

```
    Text="Send"/>
```

Piezīmes:

1. **New** EventHandler – delegāta radīšana.
 2. AddressOf Send – delegāta inicializēšana ar funkciju *Send*.
 3. Kā redzams, elementā `<asp:button>` netika izmantots atribūts `onclick`.
-

Izvades procesa demonstrēšana

```
<% Info.Text = "Done!" %>
<form runat="server">
    <asp:label id="Info" runat="server"/>
</form>
```

Rezultāts: *Done!*

Lai informācijas izvade notiek *pēc* iezīmes:

```
<asp:label id="Info" runat="server" />
```

...

```
<% Info.Text = "Done!" %>
```

Rezultāts: ziņojums **netiks** izvadīts.

Viss servera kods `<% . . . %>` tiks pievienots metodei *Render*.

Pirms tam servera elements `<asp:label...>` jau tika izvietots atbildes buferī.

Elementa iekšējais teksts ir tukšā rindiņa, kuru nevar izmainīt.

To pašu efektu iegūsim, strādājot servera pusē ar `` un citiem elementiem.

```
<span id="Info" runat="server" />
```

CSS stilu lietošana

1. Atribūts `style`.

```
<asp:label id="Info" runat="server"
  style="color:red"/>
```

2. Īpašība *Style*.

```
Sub Page_Init(Sender As Object, E As EventArgs)
  Info.Style("color")="red"
End Sub
```

Piezīme: **neder** variants

```
document.all.Info.style.color = "red"
```

Jēdziens "document" neeksistē.

Lai CSS īpašības vārds sastāv no vairākiem vārdiem:

```
Info.Style("background-color")="green"
```

3. Elements `<style>`.

```
<style type="text/css">
```

```
  #Info {color:red}
```

```
</style>
```

```
<asp:label id="Info" runat="server"/>
```

Papildu informācijas saglabāšana

Parastais ASP: objekts *Session*.

ASP.NET: objekts *ViewState* (attēlošanas stāvoklis).

ViewState ir stāvokļa elementu kolekcija.

Pamatā ir *vārdnīcas* princips.

Kolekcijā var izmantot patstāvīgi izveidotas atslēgas.

Rezultāts: datu saglabāšana starp pieprasījumiem.

Būtībā, *ViewState* ir mehānisma *State Bag* realizācija.

ViewState lietošanas piemērs

```
Sub Page_Load(Sender As Object, E As EventArgs)

    Dim Count As Integer

    If (ViewState("VisCount") Is Nothing) Then
        Count = 1
    Else
        Count = ViewState("VisCount") + 1
    End If

    ViewState("VisCount") = Count

    Response.Write("Total visited:" & _
        ViewState("VisCount"))

End Sub
```


Uzdevums: nodrošināt iespēju saglabāt visu teksta lodziņu saturu un atjaunot to nepieciešamības gadījumā.

Svarīgie jēdzieni:

1. Klase `ControlCollection`. Servera vadības elementu kolekcija.
2. Īpašība `Controls`. Atgriež attiecīgo kolekciju.

Piemērs: lai ir servera forma `User`. Tad formas vadības elementi veido kolekciju `User.Controls`.

3. Datu tipu pārveidošana VB.NET valodā:
`CType(<izteiksme>, <tipa vārds>)`.

Piemērs:

`CType(C, TextBox)`

Atslēga: unikālais identifikators ID.

1. Forma ar diviem lodziņiem un divām teksta iezīmēm.

```
<form runat="server" id="User">
```

```
  <asp:label runat="server"
    id="lName" text="Name:" />
```

Name:

Surname:

Save

Restore

```
  <asp:textbox runat="server" id="Name" /><br>
```

```
  <asp:label runat="server"
    id="lSurname" text="Surname:" />
```

```
  <asp:textbox runat="server" id="Surname" /><br>
```

```
  <asp:button runat="server" id="Save"
    text="Save" onclick="Save_Click" />
```

```
  <asp:button runat="server" id="Restore"
    text="Restore" onclick="Restore_Click" />
```

```
</form>
```

2. Vadības elementu analīze un teksta saglabāšana.

```
Private Sub SaveText (Controls As ControlCollection,  
    Nested As Boolean)  
    ' Visu vadības elementu apstrāde  
    For Each C As Control In Controls  
        ' Teksta lodziņu meklēšana  
        If (TypeOf (C) Is TextBox) Then  
            ' Stāvokļa saglabāšana  
            ViewState (C.ID) = CType (C, TextBox).Text  
        End If  
        ' Rekursija, ja ir jāapstrādā ielikta elementus  
        If C.Controls.Count <> 0 And Nested  
            SaveText (C.Controls, True)  
        End If  
    Next  
End Sub
```

3. Vadības elementu analīze un teksta atjaunošana.

```
Private Sub RestoreText(Controls As
    ControlCollection, Nested As Boolean)
    For Each C As Control In Controls
        If (TypeOf(C) Is TextBox) Then
            ` vai elementa stāvoklis tika saglabāts?
            If (ViewState(C.ID) <> Nothing) Then
                CType(C, TextBox).Text = ViewState(C.ID)
            End If
        End If

        If C.Controls.Count <> 0 And Nested
            RestoreText(C.Controls, True)
        End If

    Next
End Sub
```

4. Podziņu notikumu apstrāde.

```
Public Sub Save_Click(Sender As Object, _  
    E As EventArgs)  
    SaveText(User.Controls, True)  
End Sub
```

```
Public Sub Restore_Click(Sender As Object, _  
    E As EventArgs)  
    RestoreText(User.Controls, True)  
End Sub
```

Piezīme: papildu funkcijas *SaveText(...)* un *RestoreText(...)* tika uzrakstītas *rekursijas* dēļ.

Kolekcija `ViewState` strādā ar klases `Object` eksemplāriem.

Ja atslēgas nav, tiks ierosināts izņēmums `NullReferenceException`.

Saglabājot *objektus*, nepieciešams izmantot atribūtu `Serializable`.

Jābūt iespēja pārveidot objektu uz *baitu plūsmu*.

Baitu plūsma tiks izvietota tīmekļa lappusē, paslēptajā apgabalā.

```
<Serializable()> Class Human
```

```
...
```

```
End Class
```

```
Dim H As New Human("Uldis", "Strods")
```

```
ViewState("1") = H
```

```
R.Text = CType(ViewState("1"), Human).Name
```

C# valodas sintakse:

```
[Serializable]
```

```
class Human {...}
```

```
ViewState["1"] = H;
```

```
R.Text = ((Human) ViewState["1"]).Name;
```

Vienreizēja operācijas izpildīšana

Servera vadības elementi saglabā savu stāvokli starp tīmekļa lappušu nosūtīšanām.

Var noteikt: vai tīmekļa lappusi ielāde pirmo reizi?

```
Sub Page_Load(Sender As Object, E As EventArgs)
    If Not Page.IsPostBack Then
        Langs.Items.Add("Java")
        Langs.Items.Add("C#")
        ...
    End If
End Sub
```

Rezultāts: saraksts netiks „dublēts” pēc kārtējas ielādes.

Piezīme: var nerakstīt *Page*.

```
If Not IsPostBack Then
```

Ja formu atgriež serverim, īpašībai *IsPostBack* ir vērtība *true*.

Īpašība atļauj minimizēt kodu, kurš izpildās tīmekļa lappuses ielādes procesā.

Piemēram, sākumā var nolasīt informāciju no datubāzes un saglabāt atmiņā.

Turpmāk var strādāt ar saglabātiem datiem un neapstrādāt datubāzi.

Mūsu gadījumā saraksta alternatīvas varētu būt iegūtas no datubāzes, ar SQL vaicājuma palīdzību.

Neatkarīgi no *IsPostBack* vērtības, serveris apstrādā vaicājumu dzīves cikla ietvaros.

Piemēram, nevar izsaukt *Page_Load* pirms *Page_Init*.

Formu radīšana: papildu jautājumi

Lai ir forma (fails *User.aspx*)

```
<form runat="server">  
    <asp:textbox id="Name" runat="server"  
        text="Juris" /><br>  
    <asp:textbox id="Surname" runat="server"  
        text="Strods" />  
</form>
```

Tīmekļa lappuse pārlūkprogrammā (*View Source*).

```
<form name="ctl00"  
    method="post" action="User.aspx" id="ctl00">  
    ...  
</form>
```

1. Automātiski norādīta metode „post”.
2. Automātiski norādīts, ka tīmekļa lappuse sūta datus sev.

Iegūtajā tekstā ir parastie HTML elementi

```
<input name="Name" type="text" value="Juris"  
id="Name" /><br>
```

```
<input name="Surname" type="text" value="Strods"  
id="Surname" />
```

Tekstā ir arī *paslēptie* elementi:

```
<div>
```

```
<input type="hidden" name="__VIEWSTATE"
```

```
id="__VIEWSTATE"
```

```
value="/wEPDwUJODQ0NDUxNTQzZGSFXCa6QSmWE  
nIwU1I5lvVImJPRvA==" />
```

```
</div>
```

Tas atļauj serverim salīdzināt jaunu stāvokli ar iepriekšēju.

Nekādas šifrēšanas nav.

Mehānisma nosaukums: stāvokļu serializācijas mehānisms (*state serialization mechanism*).

Stāvokļa attēlošanas glabāšana palielina tīmekļa lappuses izmēru.

Rezultātā klients maksa *divas reizes*:

1. Saņem lielākā izmēra tīmekļa lappusi.
2. Atgriež serverim informāciju par vadības elementiem.

Ja lieto *sarežģītus* vadības elementus (GridView un citus), stāvokļu attēlošanas informācija aizņem daudz vietas.

Mehānismu var atslēgt, ja vadības elementā norādīt:
`EnableViewState=false`.

Tad jādomā par *atkārtotu* elementa inicializēšanu pirms atpakaļējas nosūtīšanas.

Formāli attēlošanas stāvokļa lauka izmērs *nav ierobežots*.

Problēma: daži ugunsdiri un starpniekserveri bloķē tīmekļa lappusi, ja paslēpto lauku izmērs ir pārāk liels.

Risinājums: paslēpto lauku var dalīt daļās.

Daļu *izmēru* norāda konfigurācijas failā `web.config`.

```
<configuration>
  <system.web>
    <pages maxPageStateFieldLength="10"/>
  </system.web>
</configuration>
```

Šis paņēmiens **nepaaugstina** ražīgumu.

Būs papildus izmaksas serializācijas nodrošināšanai.

Rezultāts:

```
<input type="hidden" name="__VIEWSTATEFIELD_COUNT"
      id="__VIEWSTATEFIELD_COUNT" value="6" />
<input type="hidden" name="__VIEWSTATE"
      id="__VIEWSTATE" value="/wEPDwUKLT" />
<input type="hidden" name="__VIEWSTATE1"
      id="__VIEWSTATE1" value="U1ODczNTQ0" />
<input type="hidden" name="__VIEWSTATE2"
      id="__VIEWSTATE2" value="MmRk/JxECJ" />
<input type="hidden" name="__VIEWSTATE3"
      id="__VIEWSTATE3" value="f5MDF3e8Ut" />
<input type="hidden" name="__VIEWSTATE4"
      id="__VIEWSTATE4" value="APP/Xidly0" />
<input type="hidden" name="__VIEWSTATE5"
      id="__VIEWSTATE5" value="4=" />
```

Vaicājumu **INSERT**, **DELETE**, **UPDATE** izpilde
Attiecīgie vaicājumi **neatgriež** rezultātu – datu kopu.

Vaicājumu izpilde: objekts *OleDbCommand*, metode
ExecuteNonQuery().

Metodes rezultāts: izmainīto ierakstu daudzums.

Ja vaicājums **nav** INSERT, DELETE vai UPDATE, rezultāts
ir -1 (ASP .NET 3.5). *Piezīme*: ASP.NET 2.0 gadījumā ir 0.

Objekts *OleDbDataReader* nav vajadzīgs.

```
Dim Cn As OleDbConnection
```

```
Dim Cmd As OleDbCommand
```

```
Dim OpenStr, SQL As String
```

```
Dim DelCount As Integer
```

```
OpenStr = _  
    "Provider=Microsoft.Jet.OLEDB.4.0;" & _  
    "Data Source=" & Server.MapPath ("firm.mdb")  
  
SQL = "DELETE FROM Employees " & _  
    "WHERE Surname='Strods'"  
  
Cn = New OleDbConnection(OpenStr)  
Cmd = New OleDbCommand(SQL, Cn)
```

Try

```
    Cn.Open()  
    DelCount = Cmd.ExecuteNonQuery()  
    Result.Text = "Deleted " & DelCount & _  
        " record(s)."
```

Catch Exc **As** OleDbException

```
    Result.Text = "Error: " & Exc.Message
```

Finally

```
    Cn.Close()
```

End Try

Rezultāta atgriešana mainīgajā

Vaicājumā var izmantot funkcijas *Sum()*, *Count()* un citas.

Funkciju rezultātu iegūšana: objekts *OleDbCommand*, metode *ExecuteScalar()*.

Rezultāts – objekts; dažās valodās nepieciešama papildu pārveidošana „objekts” \rightarrow *int*.

```
Dim EmplCount As Integer
```

```
SQL = "SELECT Count(*) FROM Employees"
```

```
Cn = New OleDbConnection(OpenStr)
```

```
Cmd = New OleDbCommand(SQL, Cn)
```

```
EmplCount = Cmd.ExecuteScalar()
```

```
Result.Text = "Total " & EmplCount & _  
    " record (s) ."
```


Visos piemēros tika izmantota konstrukcija:

```
Cmd = New OleDbCommand(SQL, Cn)
```

Visbiežāk tas ir **labākais** variants.

Alternatīva: patstāvīgi norādīt trīs īpašības.

```
Cmd = New OleDbCommand()
```

```
Cmd.Connection = Cn
```

```
Cmd.CommandType = CommandType.Text
```

```
Cmd.CommandText = "SELECT Name FROM Employees"
```

Pēc noklusējuma runa ir par komandas tipu
CommandType.Text.

Darbā ar glabājamām procedūrām izmanto vērtību
CommandType.StoredProcedure.

Savienojuma aizvēršana objektā *DataReader*

Izmanto pārlādētu metodi *ExecuteReader(...)* un uzskaitāmo tipu *CommandBehavior*.

```
Dim Rd As OleDbDataReader
...
Cn.Open()
Rd = Cmd.ExecuteReader _
    (CommandBehavior.CloseConnection)
...
Rd.Close()
```

Rezultāts: savienojums *Cn* tiks aizvērts automātiski.

Šo paņēmieni izdevīgi izmantot, ja objekts *DataReader* tiks nodots citai metodei kā parametrs.

Aizverot *DataReader* citā metodē, aizvērsim *Connection* sākotnējā metodē.

Savienojuma atcelšana blokā **using**

Koncepcija: iznīcināmo objektu izmanto īso laiku.

Pēc bloka **using** pabeigšanas CLR automātiski izsauks metodi *Dispose()*.

Objektam *Connection* tas ir *Close()* ekvivalents.

```
Using Cn As New OleDbConnection(openstr)
    Cn.Open()
    ...
    S = "Version: " & cn.ServerVersion & "<br>"
    S &= "State: " & cn.State & "<br>"

End Using

Result.Text = S
-----
Version: 04.00.0000
State: 1
```

Objektu *Cn* var aizvērt pirms bloka pabeigšanas.

```
...  
S &= "State (begin): " & cn.State & "<br>"  
Cn.Close  
S &= "State (end): " & cn.State & "<br>"
```

End Using

Rezultāts:

```
State (begin): 1  
State (end): 0
```

Objekts *Cn* eksistē tikai blokā **using**.

Aizvērt (vai vienkārši izmantot) savienojumu *Cn* aiz bloka robežām *nav iespējams*.

Nav nepieciešamības rakstīt bloku **finally** – savienojums tiks aizvērts pat neapstrādātā izņēmuma gadījumā.

To pašu stilu var ievērot C# valodā.

```
OleDbConnection cn;  
...  
cn = new OleDbConnection(openstr);  
using (cn) {  
    cn.Open();  
    s = "State (begin): " + cn.State + "<br>";  
}  
s += "State (end): " + cn.State + "<br>";
```

```
State (begin): Open  
State (end): Closed
```

Mūsu piemērā objekts *cn* tika deklarēts pirms bloka **usage**.
Līdz ar to *cn* var izmantot aiz bloka robežām.

Vecais variants:

```
using (OleDbConnection cn = new  
    OleDbConnection(openstr)) {
```

Informācijas izvades paātrināšana

Ieteicams izmantot klasi *StringBuilder*, nevis parasto teksta rindiņu konkatenāciju.

Katra konkatenācijas operācija iznīcina veco objektu un veido jauno objektu. Tas samazina apstrādes ātrumu.

Klase *StringBuilder* atrodas vārdu telpā `System.Text` (faktiski, klase tiks „pieslēgta” automātiski).

```
Dim Info As StringBuilder
Info = New StringBuilder("")
...
Info.Append("<tr><td><b>" & i & ".</b></td>")
Info.Append("<td>" & _
    Server.HtmlEncode(Rd("Name")) & "</td>")
...
Result.Text = Info.ToString()
```

Lauku adresēšana *pēc indeksiem*

Izvadīt informāciju formātā:

Name -> Jānis
Surname -> Strods
Duty -> programmētājs
Salary -> 700

...

```
Dim FInd As Integer
```

```
Do While Rd.Read()
```

```
    For FInd = 0 To Rd.FieldCount-1  
        Info &= Rd.GetName(FInd) & " -> "  
        Info &= Rd.GetValue(FInd)  
        Info &= "<br>"
```

```
    Next
```

```
    Info &= "<br>"
```

Loop

Informācijas apstrāde pēc indeksiem notiek ātrāk, bet ir samazināta koda lasāmība un universalitāte.

Parametrizētu vaicājumu lietošana

Pamatprincips: vaicājuma modificēšana.

Parastais vaicājums:

```
SELECT * FROM Employees WHERE Duty = 'operators'
```

Modificētais vaicājums:

```
SELECT * FROM Employees WHERE Duty = @amats
```

Konkrētas vērtības norāda kolekcijā Parameters.

```
Cmd = New OleDbCommand _  
    (" SELECT * FROM Employees " & _  
    " WHERE Duty = @amats", Cn)
```

```
Cmd.Parameters.Add("@amats", "operators")
```

Parametrizēšanas efekts: daļēja aizsardzība no uzbrukumiem (piemēram, SQL injekcijas).

Glabājamie vaicājumi

Lai parametrizētais vaicājums *FindDuty* atrodas MS Access datubāzē:

```
SELECT Employees.*  
FROM Employees  
WHERE Employees.Duty=[Duty:] ;
```

Parametra (amata) nodošana:

```
Cn = New OleDbConnection(OpenStr)  
Cmd = New OleDbCommand("FindDuty", Cn)  
Cmd.CommandType = CommandType.StoredProcedure  
  
Dim ParmDuty As OleDbParameter = _  
    Cmd.Parameters.Add("@Duty", _  
        SqlDbType.NVarChar, 20)  
ParmDuty.Value = "operators"
```

ADO.NET transakcijas

Transakcijas palaišana: objekts *Connection*, metode *BeginTransaction()*.

Rezultāts: kāds objekts *Transaction*.

Varianti: *OleDbTransaction*, *SqlTransaction*, *OracleTransaction*...

Klasei *Transaction* ir divas galvenās metodes:

1. *Commit()*. Transakcijas pabeigšana. Izmaiņas saglabātas datu avotā.
2. *RollBack()*. Visas izmaiņas atceltas.

Piezīme: metodi *RollBack()* ieteicams pielietot izņēmumu gadījumā.

Uzdevums: vienā transakcijā izdzēst informāciju par diviem darbiniekiem.

```
Dim T As OleDbTransaction
```

```
Cn = New OleDbConnection(OpenStr)
```

```
Cmd = New OleDbCommand()
```

```
Try
```

```
    Cn.Open()
```

```
    T = Cn.BeginTransaction()
```

```
    Cmd.Transaction = T
```

```
    Cmd.Connection = Cn
```

```
    Cmd.CommandText = "DELETE FROM Employees" & _  
        "WHERE Surname='Strods' "
```

```
    Cmd.ExecuteNonQuery()
```

```
    Cmd.CommandText = "DELETE FROM Employees" & _  
        "WHERE Surname='Zemzars' "
```

```
    Cmd.ExecuteNonQuery()
```

```
T.Commit()  
Catch Exc As Exception  
    T.Rollback()  
Finally  
    Cn.Close()  
End Try
```

Datu sasaistīšana un ierakstu atlase

Uzdevums: izvadīt ekrānā *darbinieku sarakstu* un nodrošināt nepieciešamo ierakstu atlasi. Rezultāts: *uzvārdu saraksts*.

1. Saraksts.

```
<form runat="server">  
    <asp:ListBox id="EmplInfo" runat="server"  
        SelectionMode="Multiple"  
        DataTextField="FullInfo"  
        DataValueField="Surname"/>  
</form>
```

2. Saraksta radīšana. Problēma: jālieto vairāki lauki.

```
Cn = New OleDbConnection(OpenStr)
SQL = "SELECT Surname, Name & ' ' & Surname & " & _
      "' ' & Duty & ' ' & " & _
      "Salary AS FullInfo FROM Employees"
Cmd = New OleDbCommand(SQL, cn)
Cn.Open()
Rd = Cmd.ExecuteReader()
EmplInfo.DataSource = Rd
EmplInfo.DataBind()

Rd.Close()
Cn.Close()
```

Rezultāts (fragments)

```
Jānis Strods programmētājs 600
Sergejs Ivanovs operators 700
```

Atlasītas informācijas apstrāde (fragments)

```
Dim S As String = ""  
Dim Item As ListItem  
For Each Item In EmplInfo.Items  
    If Item.Selected  
        S = S & Item.Value & "<br>"  
    End If  
Next
```

Rezultāts: izvēlētu *uzvārdu* saraksts.

Uzvārds kā vērtība tika norādīts SQL vaicājumā (pirms komata).

Datu parametrizēšana datu avotā

Uzdevums: iegūt informāciju par darbinieku ar norādītu uzvārdu.

<input type="text" value="Zemzars"/>			
Name	Surname	Duty	Salary
Uldis	Zemzars	operators	850

1. Pirmais datu avots (*uzvārdu atlasei*).

```
<asp:AccessDataSource
```

```
  id="Firm" runat="server"
```

```
  DataFile="Firm.mdb"
```

```
  SelectCommand =
```

```
    "SELECT DISTINCT Surname FROM Employees" />
```

2. Otrais datu avots (*informācijas par darbiniekiem iegūšana*).

```
<asp:AccessDataSource
  id="Result"  runat="server"
  DataFile="Firm.mdb"
  SelectCommand = "SELECT * FROM Employees
    WHERE Surname=@Surname">

  <SelectParameters>
    <asp:ControlParameter ControlID="DDL_Sname"
      Name="Surname"
      PropertyName = "SelectedValue"/>
  </SelectParameters>
</asp:AccessDataSource>
```

Piezīme: *DDL_Sname* ir uzvārdu saraksta (*DropDownList*) identifikators.

3. Uzvārdu saraksts un rezultātu tabula.

```
<form runat="server">  
  <asp:DropDownList  
    id="DDL_Sname" runat="server"  
    DataSourceId="Firm" DataTextField="Surname"  
    AutoPostBack="True" />  
  
  <asp:GridView  
    id="R" runat="server"  
    DataSourceId="Result"  
    AutoGenerateColumns="true" />  
</form>
```

Piezīme: atribūts *AutoPostBack* garantē, ka tīmekļa lappuse tiks nosūtīta atpakaļ pēc jebkurām izmaiņām uzvārdu sarakstā.

Var runāt par diviem „avotiem”: galveno un pakļauto.

Piemērā var izmantot glabājamus vaicājumus

SQL vaicājums MS Access datu bāzē (*SelectInfo*):

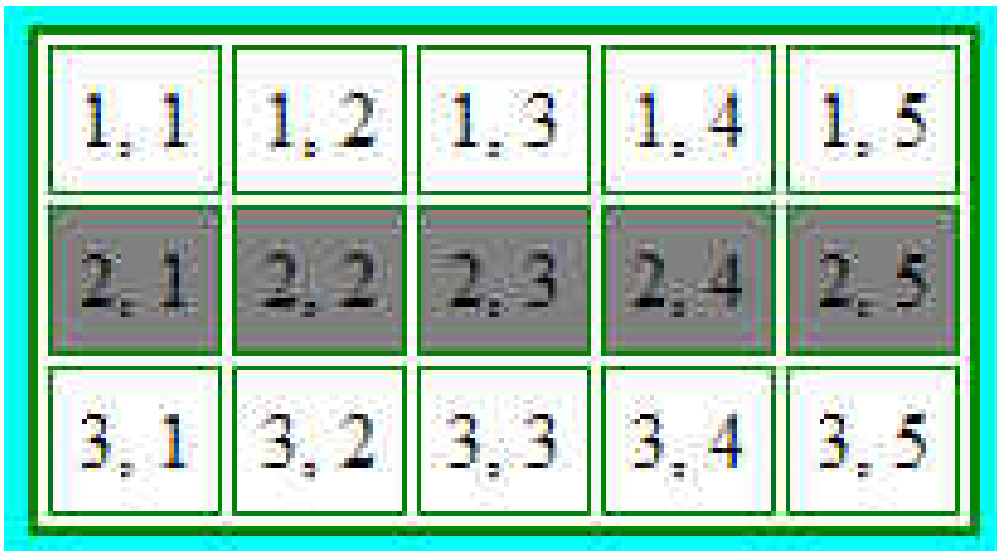
```
SELECT Employees.*  
FROM Employees  
WHERE Employees.Surname=[@Surname]
```

Datu avota fragments:

```
<asp:AccessDataSource  
  id="Result" runat="server"  
  DataFile="Firm.mdb"  
  SelectCommand = "SelectInfo"  
  SelectCommandType = "StoredProcedure" >  
  <SelectParameters>...</SelectParameters>  
</asp:AccessDataSource>
```

Dinamiskas tabulas

Uzdevums: dinamiski izveidot tabulu.



1, 1	1, 2	1, 3	1, 4	1, 5
2, 1	2, 2	2, 3	2, 4	2, 5
3, 1	3, 2	3, 3	3, 4	3, 5

Tīmekļa lappusei ir kolekcija *Controls*.

```
Dim T As HtmlTable = New HtmlTable()
```

```
Dim Row As HtmlTableRow
```

```
Dim Cell As HtmlTableCell
```

```
Dim i, j As Integer
```

```
T.Border = 2
T.CellPadding = 5
T.BorderColor = "Green"
T.BGColor = "White"
For i = 1 To 3
    Row = New HtmlTableRow()
    If (i Mod 2) = 0
        Row.BGColor = "Gray"
    End If
    For j = 1 To 5
        Cell = New HtmlTableCell()
        Cell.InnerHtml = i & ", " & j
        Row.Cells.Add(cell)
    Next
    T.Rows.Add(row)
Next
Me.Controls.Add(T)
```

Pilnfunkcionālie vadības elementi

Tādus elementus bieži programmē kā parastus objektus. Piemēram, elementiem eksistē konkrētie deskriptori.

Vēlāk objekti tiks transformēti uz vairākiem HTML elementiem. Dažreiz tie izmanto klienta *JavaScript* kodu.

Nepilnfunkcionālais vadības elements: *Table*.

Iemesls: tiks izmantoti objekti – HTML elementu čaulas (**<table>**, **<tr>**, **<td>**).

Pilnfunkcionālais vadības elements: *Calendar*.

Dienas, mēneši un gadi nav saistīti ar rezultējošu HTML iezīmēšanu.

Elements *AdRotator*

Elements atļauj radīt reklāmkarogus (banners).

Attēlu demonstrēšana notiek saskaņā ar iepriekšnoteiktiem nosacījumiem, tos deklarē XML failā.

XML faila struktūra:

<Advertisements>...</Advertisements> - saknes elements

<Ad>...</Ad> - attēla parametru deklarēšana.

1. **<ImageUrl>** - attēls (relatīva adrese mapē vai URL adrese internetā).
2. **<NavigateUrl>** - hipersaite, kas tiks aktivizēta pēc attēla izvēles.
3. **<AlternateText>** - papildus teksts (vai teksts attēla vietā).

4. **<Impressions>** - attēla demonstrēšanas frekvence.

Parametra lietošanas rezultāts atkarīgs no citām frekvencēm (proporcijas – piemēram, $6:2 = 3:1$).

5. **<Keyword>** - attēlu filtrēšana. Vēlāk var ierobežot attēlu demonstrēšanu.

Ieplānotais rezultāts:



[Valsts elektrotehniskā fabrika](#)



[Rīgas vagonu rūpnīca](#)

XML fails ar informāciju par attēliem (*adr.xml*)

```
<?xml version="1.0" ?>
```

```
<Advertisements>
```

```
  <Ad>
```

```
    <ImageUrl>
```

```
      VEF.jpg
```

```
    </ImageUrl>
```

```
    <NavigateUrl>
```

```
      http://www.vef.lv
```

```
    </NavigateUrl>
```

```
    <AlternateText>
```

```
      Valsts elektrotehniskā fabrika
```

```
    </AlternateText>
```

```
    <Impressions>
```

```
      2
```

```
    </Impressions>
```

```
    <Keyword>
```

```
      Elektronika
```

```
    </Keyword>
```

```
  </Ad>
```



```
<Ad>
  <ImageUrl>
    RVR.jpg
  </ImageUrl>
  <NavigateUrl>
    http://www.rvr.lv
  </NavigateUrl>
  <AlternateText>
    Rīgas vagonu rūpnīca
  </AlternateText>
  <Impressions>
    2
  </Impressions>
  <Keyword>
    Vagoni
  </Keyword>
</Ad>
</Advertisements>
```

Vadības elementa iebūvēšana

```
<asp:AdRotator
    id="AdR" runat="server"
    Target="_blank" AdvertisementFile="adr.xml"
    OnAdCreated="AdR_Created">
</asp:AdRotator>
<br>
<asp:HyperLink id="HlBnr" runat="server">
    HyperLink
</asp:HyperLink>
```

Skripts:

```
Sub AdR_Created(Source As Object, _
    E As AdCreatedEventArgs)
    HlBnr.NavigateUrl = e.NavigateUrl
    HlBnr.Text = e.AlternateText
End Sub
```

Informācijas filtrēšana:

```
<asp:AdRotator... KeywordFilter="Elektronika"/>
```

Informācijas nosūtīšana starp tīmekļa lappusēm

Īpašība *PostBackURL* atbilst par datu nosūtīšanu starp tīmekļa lappusēm.

Īpašības *PostBackURL* vērtība ir cita tīmekļa lappuse.

Objekts *PreviousPage*: izsaucošā tīmekļa lappuse.

Lai ir tīmekļa lappuse *MainPage.aspx*:

```
<form runat="server">  
    <asp:TextBox id="Name" runat="server" />  
    <asp:Button id="Submit" runat="server"  
        Text="Submit" PostBackURL="SubPage.aspx" />  
</form>
```

Tīmekļa lappuse *SubPage.aspx*

```
Sub Page_Load(Source As Object, E As EventArgs)
    If (Not PreviousPage Is Nothing) Then
        Result.Text = "Hello, " & _
            CType(PreviousPage.FindControl("Name"), _
                TextBox).Text & "."
    End If
End Sub
```

```
<form runat="server">
    <asp:Label runat="server" id="Result"/>
</form>
```

C# pārbaude:

```
if (PreviousPage != null)
```

Elements *GridView*: papildu informācija

Uzdevums: iegūt tabulu ar informāciju par darbiniekiem.

Name	Surname	Duty	Salary
Jānis	Strods	programmētājs	1100
Sergejs	Ivanovs	operators	900
1	2		

Lappušu numuri

Citādi būs
papildu,
automātiski
izveidotas
kolonas

```
<asp:AccessDataSource id="Firm" .../>
<asp:GridView id="GV" runat="server"
  DataSourceID="Firm" GridLines="None"
  AutoGenerateColumns="False"
  AllowPaging="True" PageSize="2"
  CellPadding="5" BorderWidth="4"
  BorderColor="Green">
```

```
<HeaderStyle ForeColor="Black"
    BackColor="Yellow" Font-Bold="True"/>
<RowStyle ForeColor="Green"
    BackColor="#C0C0C0"/>
<Columns>
    <asp:BoundField DataField="Name"
        HeaderText=" Name "/>
    <asp:BoundField DataField="Surname"
        HeaderText=" Surname "/>
    <asp:BoundField DataField="Duty"
        HeaderText=" Duty "/>
    <asp:BoundField DataField="Salary"
        HeaderText=" Salary "/>
</Columns>
</asp:GridView>
```

Daļā *<Columns>* var patstāvīgi definēt tabulas kolonas.

Tas nodrošina papildu formatēšanu katrai kolonai.

Tieši definētas kolonas strādā *ātrāk*, nekā automātiski iegūtas kolonas.

Kolonas slēpšana:

```
<asp:BoundField DataField="Salary" ...  
    Visible="False" />
```

Tukšas vērtības vizualizēšana:

```
<asp:BoundField DataField="Salary"  
    NullDisplayText="Not specified" />
```

Daži kolonu tipi:

CheckBoxField, HyperlinkField, ImageField.

Tabulas kolonu formatēšana: ieliktie elementi *ItemStyle*.

```
<asp:BoundField DataField="Surname" ...>  
    <ItemStyle ForeColor="Magenta"  
        BackColor="White" Width="100pt"/>  
</asp:BoundField>
```

Alternatīvs risinājums:

```
<asp:BoundField DataField="Surname" ...  
    ItemStyle-ForeColor="Magenta"  
    ItemStyle-BackColor="White"  
    ItemStyle-Width="100pt" />
```

Šos paņēmienus visbiežāk lieto kolonas platuma definēšanai.

Katras otrās rindiņas formatēšana:

```
<AlternatingRowStyle BackColor="Red" ... />
```


Dažas citas *GridView* vizuālās īpašības:

Tabulas *virsraksts*:

... **Caption**="Personnel" ...

Tabulas virsraksta *izlīdzināšana*:

... **CaptionAlign**="Bottom" ...

Attālumi šūnu *iekšā* un *starp* šūnām:

... **CellPadding**="10" ... **CellSpacing**="5" ...

Horizontālā tabulas izlīdzināšana lappusē

... **HorizontalAlign**="Center" ...

Virsraksta rindiņas demonstrēšana (lauku nosaukumi):

... **ShowHeader**="False" ...

Piezīme: līdzīgi norāda **ShowFooter**.

Informācijas kārtošana

Uzdevums: kārtot darbiniekus pēc amata.

Secība tiks izmainīta pēc klikšķinājuma *kolonas virsrakstā*.

Employees	<u>Duty</u>	Salary	Employees	<u>Duty</u>	Salary	Employees	<u>Duty</u>	Salary
Jānis Strods	programmētājs	1100	Uldis Zemzars	operators	1000	Jānis Strods	programmētājs	1100
Uldis Zemzars	operators	1000	Jānis Strods	programmētājs	1100	Ivans Petrovs	programmētājs	800
Ivans Petrovs	programmētājs	800	Ivans Petrovs	programmētājs	800	Uldis Zemzars	operators	1000

```
<asp:GridView ... AllowSorting="True" >
```

```
<asp:BoundField DataField="Duty"
    HeaderText="Duty" SortExpression="Duty"/>
```

Kārtošanas izteiksmē var norādīt *vairākus* laukus

```
<asp:BoundField...
    SortExpression="Duty, Surname"/>
```

Lappušu paneļa noskaņošana

Uzdevums: pārvietošanai izmantot hipersaites *Forward* un *Back*.

Name	Surname	Duty	Salary	Name	Surname	Duty	Salary
Jānis	Strods	programmētājs	1100	Uldis	Zemzars	operators	1000
Sergejs	Ivanovs	operators	900	Ivans	Petrovs	programmētājs	800

Forward >

< Back Forward >

Koda fragments (<asp:GridView> daļa):

```
<PagerSettings Mode="NextPrevious"
    PreviousPageText="< Back"
    NextPageText="Forward >" />
<PagerStyle BackColor="Black" ForeColor="Yellow"
    HorizontalAlign="Center" />
```

Parametra *Mode* vērtības

1. *Numeric*. Norādes uz citām lappusēm (1, 2, ...).
2. *NextPrevious*. Norādes „uz priekšu” un „atpakaļ”. Var izmantot attēlus. Tad ...*Text* vietā būs ...*ImageURL*.

```
<PagerSettings Mode="NextPrevious"
  PreviousPageImageURL="ArrBack.jpg"
  NextPageImageURL="ArrForw.jpg" />
```



3. *NextPreviousFirstLast*. Papildus norādes uz pirmo un pēdējo lappusēm.

```
<PagerSettings Mode="NextPreviousFirstLast"
  FirstPageText="<< First"
  PreviousPageText="< Back"
  NextPageText="Forward >"
  LastPageText="Last >>" />
```

Šablonu lietošana

Uzdevums: izvadīt informāciju par darbinieku vārdiem un uzvārdiem vienā kolonā.

Employees	Duty	Salary
Jānis Strods	programmētājs	1100
Sergejs Ivanovs	operators	900

```
<Columns>
```

```
<asp:TemplateField HeaderText="Employees">
```

```
<ItemTemplate>
```

```
<em><%# Eval("Name") %>
```

```
<%# Eval("Surname") %></em>
```

```
</ItemTemplate>
```

```
</asp:TemplateField>
```

```
<asp:BoundField DataField="Duty"
```

```
HeaderText=" Duty " />
```

```
...
```

```
</Columns>
```

Eval() ir klases *System.Web.UI.DataBinder* statiskā metode.

GridView darba algoritms:

1. Apstrādāt šablonu *TemplateField* katram elementam.
2. Aprēķināt visas pārējas datu piesaistīšanas izteiksmes.
3. Pievienot ģenerētu HTML kodu tabulai.

Eval() atļauj nedomāt par datu objekta tipu.

Galvenais – objekta eksistēšana.

Rezultāts: lappuses, kuras *nav* stingri saistītas ar datu piekļuves līmeni.

Var vispār neizmantot `<asp:BoundField>`:

`<Columns>`

`<asp:TemplateField HeaderText="Employees">`

`<ItemTemplate>`

`Name, surname: `

`<%# Eval("Name") %>`

`<%# Eval("Surname") %>
`

`Duty, salary: `

`<%# Eval("Duty") %>`

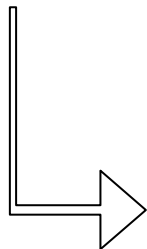
`<%# Eval("Salary") %><hr>`

`</ItemTemplate>`

`</asp:TemplateField>`

`</Columns>`

Rezultāts:
nav kolonu.



Employees

Name, surname: *Jānis Strods*

Duty, salary: *programmētājs 1100*

Name, surname: *Sergejs Ivanovs*

Duty, salary: *operators 900*

Informācijas dzēšana elementā *GridView*

Blakus ar katru ierakstu var izveidot hipersaiti *Delete*.

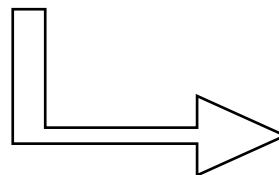
1. Elements `<asp:AccessDataSource>`

```
<asp:AccessDataSource ...
  SelectCommand = "SELECT * FROM Employees"
  DeleteCommand = "DELETE * FROM Employees
    WHERE Surname=@Surname"/>
```

2. Elements `<asp:GridView>`

```
<asp:GridView ...
  AutoGenerateDeleteButton="True"
  DataKeyNames="Surname">
```

```
<Columns>
  ...
</Columns>
</asp:GridView>
```



Employees

Delete	Jānis Strods programmētājs	1100
Delete	Uldis Zemzars operators	1000
Delete	Ivans Petrovs programmētājs	800

Dzēšana ar `<asp:CommandField>` palīdzību

	Employees	Duty	Salary
<input type="button" value="Delete"/>	Jānis Strods	programmētājs	1100
<input type="button" value="Delete"/>	Sergejs Ivanovs	operators	900

```
<asp:AccessDataSource id="Firm" ...
    DataKeyNames="Surname"
    DeleteCommand = "DELETE Employees.*
        FROM Employees WHERE Surname=@Surname"/>

<asp:GridView...>
    ...
    <Columns>
        <asp:CommandField ShowDeleteButton="True"
            ButtonType="Button" DeleteText="Delete"/>
    ...
    </Columns>
</asp:GridView>
```

Elements `<asp:DetailsView>`: viena ieraksta vizualizēšana.

Piezīme: spēkā ir *GridView* iekšējās formatēšanas principi.

Name	Jānis
Surname	Strods
Duty	programmētājs
Salary	1100
1 2 3 4 5 6	



Name	Uldis
Surname	Zemzars
Duty	operators
Salary	1000
1 2 3 4 5 6	



...

```
<asp:AccessDataSource id="Firm" .../>
```

```
<asp:DetailsView id="GV" runat="server"
    DataSourceID="Firm" AllowPaging="True"
    AutoGenerateRows="False">
```

```
<Fields>
```

```
    <asp:BoundField DataField="Name"
        HeaderText=" Name " />
```

```
    ...
```

```
</Fields>
```

```
</asp:DetailsView>
```

Ieraksta ielikšana (neder *GridView*)

Name	Jānis
Surname	Strods
Duty	programmētājs
Salary	1100
New	
1 2 3 4 5 6	

Name	<input type="text"/>
Surname	<input type="text"/>
Duty	<input type="text"/>
Salary	<input type="text"/>
Insert Cancel	

```
<asp:AccessDataSource...
```

```
    SelectCommand = "SELECT * FROM Employees"
```

```
    InsertCommand = "INSERT INTO Employees
```

```
        (Name, Surname, Duty, Salary)
```

```
    VALUES (@Name, @Surname, @Duty, @Salary)"/>
```

```
<asp:DetailsView...
```


```
    AllowPaging="True"
```


```
    AutoGenerateRows="False"
```

```
    AutoGenerateInsertButton="True">
```

Pārvietošana starp ierakstiem *ar šablona palīdzību*

Ieplānotais rezultāts:

Name	Jānis
Surname	Strods
Duty	programmētājs
Salary	1100
 Page 1 of 3	
Previous	

Name	Jānis
Surname	Strods
Duty	programmētājs
Salary	1100
 Page 1 of 3	
Next	

Izvada ekrānā ar skripta palīdzību

1. Elements *DetailsView*.

```
<asp:DetailsView runat="server" id="DV"
    DataSourceID="Firm" AllowPaging="True"
    OnDataBound="DV_DataBound" >
```

2. Šablons *PagerTemplate* (*DetailsView* iekšā)

<PagerTemplate>

```
<asp:LinkButton runat="Server" id="Prev"
    Text="<" CommandName="Page"
    CommandArgument="Prev" ToolTip="Previous"/>
```

```
<asp:LinkButton id="Next" runat="Server"
    Text=">" CommandName="Page"
    CommandArgument="Next" ToolTip="Next"/>
```

 ; ; ; ;

Page

<asp:Label id="PNum Lbl" runat="server" />

of

**<asp:Label id="Tot Lbl" runat="server" /> **

</PagerTemplate>

3. Notikuma *OnDataBound* apstrāde.

```
Sub DV_DataBound(Source As Object,  
    E As EventArgs)  
    Dim PRow As DetailsViewRow = DV.BottomPagerRow  
    Dim Page As Integer  
    Dim Count As Integer  
  
    Dim PNum As Label =  
        CType(PRow.Cells(0).FindControl("PNum_Lbl"),  
            Label)  
  
    Dim TotNum As Label =  
        CType(PRow.Cells(0).FindControl("Tot_Lbl"),  
            Label)  
  
    If (PNum IsNot Nothing) And  
        (TotNum IsNot Nothing) Then
```

```
Page = DV.DataItemIndex + 1
```

```
Count = DV.DataItemCount
```

```
PNum.Text = Page
```

```
TotNum.Text = Count
```

End If

End Sub

Piezīmes:

1. *CommandName*="Page". Obligāta vērtība.
2. *CommandArgument*="Prev". Obligāta vērtība, nav saistīta ar *id* vērtību.
3. *BottomPagerRow* – automātiski pievienota rindiņa (obligāti jābūt *AllowPaging*="True").
4. *DataItemIndex* – ieraksta indekss vadības elementā *DetailsView* (0, 1, ...).

Elements *FormView* (ASP.NET 2.0)

Konceptuāli līdzīgs elementam *GridView*.

Darbs ar *vienu* ierakstu, pamatā *šabloni*.

Nav iespējams strādāt ar tabulas kolonām.

```
<asp:FormView id="FV" runat="server">
```

```
<Columns>
```

```
<asp:BoundField DataField="Name"
  HeaderText="Name" />
```

```
...
```

```
</Columns>
```

```
</asp:FormView>
```

Pareizi:

```
<asp:FormView id="FV" runat="server" ...
```

```
  AllowPaging="True" >
```

```
<ItemTemplate>
```

```
<%# Eval("Name") %> <%# Eval("Surname") %>
```

```
</ItemTemplate>
```

```
</asp:FormView>
```


FormView un *DetailsView*: līdzība un atšķirības

Abos gadījumos ir trīs savstarpēji izslēdzošie režīmi: lasīšana, ielikšana, rediģēšana.

FormView neatbalsta darbu ar `<asp:CommandField>`.

Alternatīvais risinājums: `<asp:Button>` vai `<asp:LinkButton>` ar atribūtu *CommandName*.

`<ItemTemplate>`

```
<%# Eval("Name") %> <%# Eval("Surname") %>
<asp:Button runat="server" Text="Delete"
  CommandName="Delete"/>
```

`</ItemTemplate>`

Obligāti jābūt:

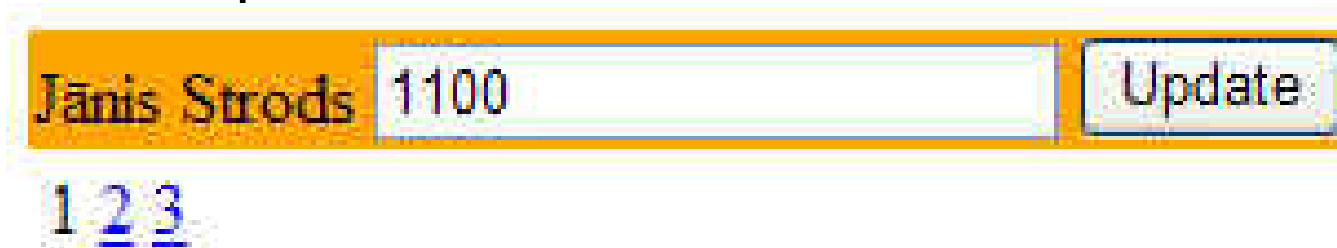
`<asp:AccessDataSource`

```
DeleteCommand = "DELETE * FROM..." />
```

`<asp:FormView DataKeyNames="Surname" ... />`

Informācijas rediģēšana

Uzdevums: nodrošināt iespēju rediģēt darbinieka algu ar šablona palīdzību.



Jānis Strods 1100 Update

1 2 3

1. Datu avots.

```
<asp:AccessDataSource
```

```
  runat="server" id="Firm"
```

```
  DataFile="Firm.mdb"
```

```
  SelectCommand = "SELECT * FROM Employees"
```

```
  UpdateCommand = "UPDATE Employees SET  
    Salary=@Salary WHERE Surname=@Surname" />
```

2. Elements *FormView*.

```

<asp:FormView id="FV" runat="server"
DataSourceID="Firm" DataKeyNames="Surname"
DefaultMode="Edit"
EmptyDataText="No employees found."
AllowPaging="True">
  <EditRowStyle BackColor="Orange"/>
  <EditItemTemplate>
    <%# Eval("Name") %> <%# Eval("Surname") %>
    <asp:TextBox id="Sal" runat="server"
      Text='<%# Bind("Salary") %>' />
    <asp:Button Text="Update"
      runat="server" CommandName="Update" />
  </EditItemTemplate>
</asp:FormView>

```

Pogas *Cancel* radīšana un informācijas dzēšana

Jānis Strods	1100	Update	Cancel
--------------	------	--------	--------

1 2 3

```
<EditItemTemplate>
```

```
...
```

```
<asp:Button Text="Cancel" runat="server"
  CommandName="Cancel"/>
```

```
</EditItemTemplate>
```

```
<asp:AccessDataSource
```

```
  id="Firm" runat="server"
```

```
...
```

```
  DeleteCommand = "DELETE FROM Employees
    WHERE Surname=@Surname"/>
```

```
<EditItemTemplate>
```

```
...
```

```
<asp:Button Text="Delete" runat="server"
  CommandName="Delete"/>
```

```
</EditItemTemplate>
```

Ieraksta ielikšana

Jānis Strods programmētājs 1100

Vasilijs Semjonovs programmētājs 950

1 2 3 1 2 3 4

Name:

Surname:

Duty:

Salary:

1. Datu avots.

```
<asp:AccessDataSource id="Firm" runat="server"
    SelectCommand = "SELECT * FROM Employees"
    InsertCommand = "INSERT INTO Employees
        VALUES (@Name, @Surname, @Duty, @Salary)"/>
```

2. Elementa *FormView* sākums.

```
<asp:FormView id="FV" runat="server"
    DataSourceID="Firm" DataKeyNames="Surname"
    AllowPaging="True">
```

3. Informācijas vizualizēšana (šablons *<ItemTemplate>*).

```
<ItemTemplate>
    <%# Eval("Name") %> <%# Eval("Surname") %>
    <%# Eval("Duty") %> <%# Eval("Salary") %>
    <asp:Button runat="server"
        Text="New" CommandName="New"/>
</ItemTemplate>
```

4. Informācijas ielikšana (šablons *InsertItemTemplate*).

Piezīme: labākai informācijas vizualizēšanai tiks izveidota HTML tabula.

```
<InsertItemTemplate>
```

```
<table>
```

```
<tr>
```

```
<td>Name:</td>
```

```
<td>
```

```
<asp:TextBox id="Name" runat="server"
```

```
Text='<%# Bind("Name") %>' />
```

```
</td>
```

```
</tr>
```

• • •

```
<tr>
```

```
<td>Salary:</td>
```

```
<td>
```

```
<asp:TextBox id="Salary" runat="server"
```

```
Text='<%# Bind("Salary") %>' />
```

```
</td>
```

```
</tr>
```

```
</table>
```

```
<asp:Button Text="Insert" runat="server"
```

```
CommandName="Insert"/>
```

```
</InsertItemTemplate>
```

Vadības elements *MultiView* (ASP.NET 2.0)

Uzdevums: demonstrēt lietotājam informāciju par vairākām programmēšanas valodām *vienā* ekrāna apgabalā.



1. Metodes *Page_Load()* fragments.

```
Sub Page_Load(...)
```

```
...
```

```
If Not IsPostBack Then
```

```
    Langs.ActiveViewIndex = 0
```

```
End If
```

```
End Sub
```

Piezīme: reālajā dzīvē alternatīvas izvēle notiek sarakstā, radiopogās, un tā tālāk.

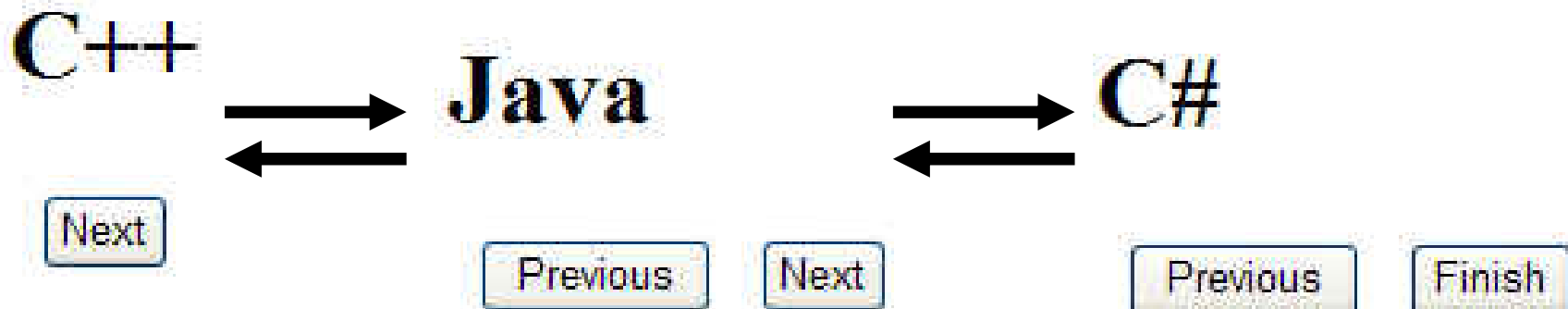
2. Vadības elements.

```
<asp:MultiView runat="server" id="Langs">  
  <asp:View runat="server" id="Cpp">  
    <h1>C++</h1>  
    <asp:Button runat="server"  
      Text="Next" CommandName="NextView" />  
  </asp:View>  
  
  <asp:View runat="server" id="Java">  
    <h1>Java</h1>  
    <asp:Button runat="server"  
      Text="Prev" CommandName="PrevView" />  
    <asp:Button runat="server"  
      Text="Next" CommandName="NextView" />  
  </asp:View>  
  
  . . .  
</asp:MultiView>
```

Vadības elements *Wizard* (ASP.NET 2.0)

Var iegūt *MultiView* elementa efektu, bet galvenais princips – vedņa radīšana.

Ieplānotais rezultāts:



```
<asp:Wizard runat="server" id="Langs">
  <WizardSteps>
    <asp:WizardStep runat="server"
      StepType="Start">
      <h1>C++</h1>
    </asp:WizardStep>
```

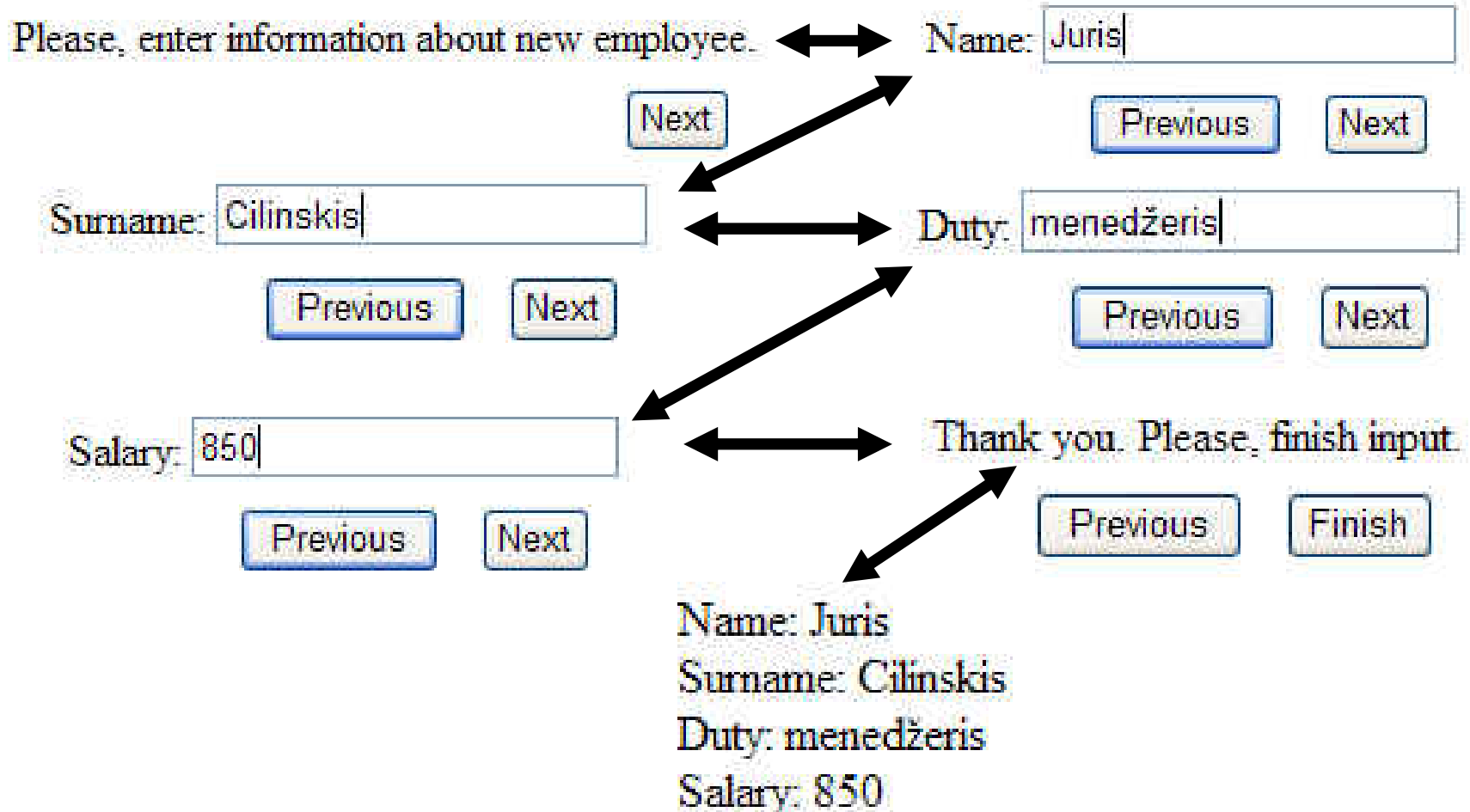
```
<asp:WizardStep runat="server"  
  StepType="Step" >  
  <h1>Java</h1>  
</asp:WizardStep>  
  
<asp:WizardStep runat="server"  
  StepType="Finish">  
  <h1>C#</h1>  
</asp:WizardStep>  
  
</WizardSteps>  
</asp:Wizard>
```

Elementu *Wizard* izdevīgi izmantot secīgai datu ievadei.

Iespējamie soļi: *Start*, *Step*, *Finish*, *Complete*.

Jauna darbinieka pievienošana

Ieplānotais rezultāts:



1. Elements *Wizard*.

```
<asp:Wizard id="Info" runat="server"
  OnFinishButtonClick="Finished">
  <WizardSteps>
    <asp:WizardStep runat="server"
      StepType="Start">
      Please, enter information
      about new employee.
    </asp:WizardStep>
    <asp:WizardStep runat="server"
      StepType="Step">
      Name: <asp:textbox id="Name"
        runat="server"/>
    </asp:WizardStep>
    . . .
    <asp:WizardStep runat="server"
      StepType="Step">
      Salary: <asp:textbox id="Salary"
        runat="server"/>
    </asp:WizardStep>
```

```

<asp:WizardStep runat="server"
    StepType="Finish">
    Thank you. Please, finish input.
</asp:WizardStep>
<asp:WizardStep runat="server"
    StepType="Complete">
    <asp:label id="Res" runat="server" />
</asp:WizardStep>
</WizardSteps>
</asp:Wizard>

```

2. Metode *Finished*.

```

Sub Finished(ByVal sender As _
    Object, ByVal e As WizardNavigationEventArgs)
    Dim S As String = ""
    S &= "Name: " & Name.Text & "<br>"
    ...
    Res.Text = S
End Sub

```

Pogu stila regulēšana



```
<asp:Wizard runat="server" id="Info">
  <StepStyle BackColor="LightGray"/>
  <SideBarButtonStyle BackColor="Green"/>
  <StartNextButtonStyle ForeColor="Green"/>
  <StepNextButtonStyle ForeColor="Blue"/>
  <StepPreviousButtonStyle ForeColor="Blue"/>
  <FinishPreviousButtonStyle ForeColor="Blue"/>
  <FinishCompleteButtonStyle ForeColor="Green"/>
  ...
</asp:Wizard>
```

Darbs ar XML dokumentiem

Lai ir objekts *DataSet* (jau tika apskatīts lekciju konspektā).

Dim Ds **As** DataSet

```
GV.DataSource = Ds  
DataBind()
```

Piezīme: GV ir *GridView* rezultāta kontrolei.

<asp:GridView id="GV" runat="server"...>

XML faila radīšana pašreizējā katalogā:

```
Ds.WriteXML(Server.MapPath("firm.xml"))
```

XML shēmas radīšana:

```
Ds.WriteXMLSchema(Server.MapPath("firm.xsd"))
```


XML faila fragments:

```

<?xml version="1.0" standalone="yes"?>
<NewDataSet>
  <Employees>
    <Name>Jānis</Name>
    <Surname>Strods</Surname>
    <Duty>programmētājs</Duty>
    <Salary>1100</Salary>
  </Employees>
  ...
</NewDataSet>

```

XML shēma:

```

<?xml version="1.0" standalone="yes"?>
<xs:schema id="NewDataSet" xmlns=""
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns:msdata="urn:schemas-microsoft-com:xml-
msdata">

```

```
<xs:element name="NewDataSet"
  msdata:IsDataSet="true"
  msdata:UseCurrentLocale="true">
  <xs:complexType>
    <xs:choice minOccurs="0"
      maxOccurs="unbounded">
      <xs:element name="Employees">
        <xs:complexType>
          <xs:sequence>
            <xs:element name="Name"
              type="xs:string" minOccurs="0"/>
            <xs:element name="Surname"
              type="xs:string" minOccurs="0"/>
            <xs:element name="Duty"
              type="xs:string" minOccurs="0"/>
            <xs:element name="Salary"
              type="xs:double" minOccurs="0"/>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
```

```
</xs:choice>  
</xs:complexType>  
</xs:element>  
</xs:schema>
```

XML dokumenta lasīšana

```
Dim Ds As New DataSet  
Ds.ReadXMLSchema(Server.MapPath("firm.xsd"))  
Ds.ReadXML(Server.MapPath("firm.xml"))  
GV.DataSource = Ds  
GV.DataBind()
```

Visbiežāk shēmu ielādē *pirms* XML dokumenta ielādes.

Objekts *DataSet* var patstāvīgi konstruēt shēmu ielādes procesā.

Automātiski iegūta shēma var atšķirties no programmētāja priekšstata.

XD shēma

Lai nepieciešams aprakstīt *tikai vienu* darbinieku.

<EMPLOYEE>

...

</EMPLOYEE>

Piezīme: šajā gadījumā saknes elementā *nenorāda* shēmu.

<xsd:schema

xmlns:xsd="http://www.w3.org/2001/XMLSchema" >

<xsd:element name="EMPLOYEE">

<xsd:complexType>

<xsd:sequence>

<xsd:element name="NAME"
type="xsd:string"/>

<xsd:element name="SURNAME"
type="xsd:string"/>

```

<xsd:element name="DUTY"
  type="xsd:string"/>
<xsd:element name="SALARY"
  type="xsd:float"/>
</xsd:sequence>
</xsd:complexType>
</xsd:element>
</xsd:schema>

```

Elementu **</xsd:complexType>** lieto, ja:

1. Elementa iekšā ir *citi elementi*.
2. Elementa tagā ir *atribūti*.

Elementu NAME, SURNAME, ..., SALARY secība ir *fiksēta*.

Elementa **</xsd:sequence>** vietā var izmantot **<xsd:all>** .

Tad elementu secība ir *patvaļīga* (SURNAME, NAME, ...).

Lai elementam **<STUDIOUS>** ir informācija par skolu un klasi, vai par universitāti un kursu.

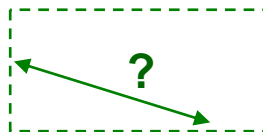
<STUDIOUS>

<NAME>Mihails**</NAME>**

<SURNAME>Dmitrijevs**</SURNAME>**

<SCHOOL>23**</SCHOOL>**

<CLASS>7**</CLASS>**



</STUDIOUS>

<UNIVERSITY>RTU**</UNIVERSITY>**

<COURSE>23**</COURSE>**

Elementi **<xsd:sequence>**, **<xsd:all>** un **<xsd:choice>** noteic grupas modeli (*model group*).

<xsd:complexType name="tSTUDIOUS">

<xsd:sequence>

<xsd:element name="NAME" type="xsd:string"/>

<xsd:element name="SURNAME"

type="xsd:string"/>

<xsd:choice>**<xsd:sequence>****<xsd:element** name="SCHOOL"
type="xsd:int"/>**<xsd:element** name="CLASS"
type="xsd:int"/>**</xsd:sequence>****<xsd:sequence>****<xsd:element** name="UNIVERSITY"
type="xsd:string"/>**<xsd:element** name="COURSE"
type="xsd:int"/>**</xsd:sequence>****</xsd:choice>****</xsd:sequence>****</xsd:complexType>****<xsd:element** name="STUDIOUS" type="tSTUDIOUS"/>

Lai nepieciešams aprakstīt firmu *ar vairākiem* darbiniekiem.

```
<xsd:schema
```

```
  xmlns:xsd="http://www.w3.org/2001/XMLSchema" >
```

```
    <xsd:element name="FIRM" type="tFIRM"/>
```

```
    <xsd:complexType name="tFIRM">
```

```
      <xsd:sequence>
```

```
        <xsd:element
```

```
          name="EMPLOYEE" type="tEMPLOYEE"
```

```
          minOccurs="0" maxOccurs="unbounded" />
```

```
      </xsd:sequence>
```

```
    </xsd:complexType>
```

```
    <xsd:complexType name="tEMPLOYEE">
```

```
      <xsd:sequence>
```

```
        <xsd:element name="NAME" ...>
```

```
      </xsd:sequence>
```

```
    </xsd:complexType>
```

```
</xsd:schema>
```


Firma ar vairākiem darbiniekiem: *alternatīvais* risinājums.

Tipu deklarācijās *netiks* izmantoti vārdi.

```
<xsd:element name="FIRM">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="EMPLOYEE"
        minOccurs="1" maxOccurs="unbounded">
        <xsd:complexType>
          <xsd:sequence>
            <xsd:element name="NAME"
              type="xsd:string"/>
            . . .
            <xsd:element name="SALARY"
              type="xsd:float"/>
          </xsd:sequence>
        </xsd:complexType>
      </xsd:element>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
```

Shēmu var dalīt daļās un saglabāt *vairākos failos*.

```
<xsd:include schemaLocation="Employee.xsd"/>
<xsd:element name="FIRM">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="EMPLOYEE"
        type="tEMPLOYEE" minOccurs="1"
        maxOccurs="unbounded"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
```

Faila Employee.xsd fragments:

```
<xsd:complexType name="tEMPLOYEE">
  <xsd:sequence>
    <xsd:element name="NAME" type="xsd:string"/>
    ...
  </xsd:sequence>
</xsd:complexType>
```

Lai elementam **<EMPLOYEE>** ir *tikai atribūti*.

```
<EMPLOYEE Name="Mihails" Surname="Dmitrijevs"  
  Duty="Programmer" Salary="350.00"/>
```

```
<xsd:complexType name="tEMPLOYEE">  
  <xsd:attribute name="Name"  
    type="xsd:string" use="required"/>  
  <xsd:attribute name="Surname"  
    type="xsd:string" use="required"/>  
  <xsd:attribute name="Duty"  
    type="xsd:string" use="required"/>  
  <xsd:attribute name="Salary"  
    type="xsd:float" use="required"/>  
</xsd:complexType>
```

Pēc noklusēšanas atribūts *nav obligāts*.

To var norādīt patstāvīgi: `use="optional"`.

Lai elements **<EMPLOYEE>** ir *jauktais*.

```
<EMPLOYEE Salary="350.0">
  Mihails Dmitrijevs
  <DUTY>Programmer</DUTY>
</EMPLOYEE>
```

```
<xsd:complexType name="tEMPLOYEE" mixed="true" >
  <xsd:sequence>
    <xsd:element name="DUTY"
      type="xsd:string"/>
  </xsd:sequence>
  <xsd:attribute name="Salary" type="xsd:float"
    use="required"/>
</xsd:complexType>
```

Piezīme: elementu **<xsd:attribute>** var deklarēt tikai *pēc* **<xsd:sequence>** elementa.

Lai elementam **<EMPLOYEE>** ir *teksta saturs* un *atribūti*.

```
<EMPLOYEE Duty="Programmer" Salary="350.0">
  Mihails Dmitrijevs
</EMPLOYEE>
```

Elements **<EMPLOYEE>** *paplašinās* teksta rindiņu.

Paplašināšanu nodrošina *divu atribūtu pievienošana*.

```
<xsd:complexType name="tEMPLOYEE">
  <xsd:simpleContent>
    <xsd:extension base="xsd:string">
      <xsd:attribute name="Duty"
        type="xsd:string" use="required"/>
      <xsd:attribute name="Salary"
        type="xsd:float" use="required"/>
    </xsd:extension>
  </xsd:simpleContent>
</xsd:complexType>
<xsd:element name="EMPLOYEE" type="tEMPLOYEE"/>
```

Lai piemērā ar skolniekiem/studentiem norāda arī *dzimumu*.
Nepieciešams paplašināt jau *eksistējošo* tipu tSTUDIOUS.

```
<STUDIOUS>
```

```
  <NAME>Mihails</NAME>
```

```
  . . .  
  <GENDER>M</GENDER>
```

```
</STUDIOUS>
```

```
<xsd:complexType name="tSTUDIOUS_GENDER">
```

```
  <xsd:complexContent>
```

```
    <xsd:extension base="tSTUDIOUS">
```

```
      <xsd:sequence>
```

```
        <xsd:element name="GENDER"  
          type="xsd:string"/>
```

```
      </xsd:sequence>
```

```
    </xsd:extension>
```

```
  </xsd:complexContent>
```

```
</xsd:complexType>
```

```
<xsd:element name="STUDIOUS"  
  type="tSTUDIOUS_GENDER"/>
```

Lai elementa vērtība var būt tikai *pozitīvais* skaitlis.

```
<AGE>1</AGE>
```

Shēmas fragments:

```
<xsd:element name="AGE"
  type="xsd:positiveInteger"/>
```

Lai elementa vērtība var būt skaitlis *noteiktajā diapazonā*.
Nepieciešams deklarēt *vienkāršo* tipu.

```
<xsd:simpleType name="tAge">
  <xsd:restriction
    base="xsd:nonNegativeInteger">
    <xsd:minInclusive value="0"/>
    <xsd:maxInclusive value="130"/>
  </xsd:restriction>
</xsd:simpleType>

<xsd:element name="AGE" type="tAge"/>
```

Elementārie datu tipi: **byte**, **short**, **int**, **long**, **float**, **double**.
Runa ir par *Java* valodas tipiem.

Reālo skaitļu deklarēšanai var arī izmantot tipu **decimal**.

Tipi **nonNegativeInteger** un **positiveInteger** ir tipa **integer** apakštipi.

Tipam **nonNegativeInteger** ir apakštipi bezzīmju veselo skaitļu deklarēšanai: **unsignedByte**, **unsignedShort**, **unsignedInt**, **unsignedLong**.

Skaitļi sešpadsmitnieku sistēmā: **hexBinary**.

Bula tips: **boolean** (**false**, **true**). **Neder**: **False**, **True**.

Laika tips: **time**. Formāts – hh:mm:ss (21:15:59).

Datuma tips: **date**. Formāts – yyyy:mm:dd (2011-11-24).

Lai cilvēka *vārda garums* ir diapazonā 1 . . 20.

<NAME>Uldis**</NAME>**

Shēmas fragments:

```
<xsd:simpleType name="tNAME">  
  <xsd:restriction base="xsd:string">  
    <xsd:minLength value="1"/>  
    <xsd:maxLength value="20"/>  
  </xsd:restriction>  
</xsd:simpleType>  
  
<xsd:element name="NAME" type="tNAME"/>
```

Elementi **minInclusive**, **maxInclusive**, **minLength**, **maxLength** un *vairāki citi* nodrošina ierobežojumus.

Šos elementus sauc par *fasetēm (facets)*.

Jaunā tipa veidošanas koncepcija ir *sašaurināšana*.

Lai darbiniekam ir *trīs veidu* diplomi: bakalaurs, inženieris, doktors.

<DIPLOMA>Master**</DIPLOMA>**

Shēmas fragments:

```
<xsd:simpleType name="tDIPLOMA">
  <xsd:restriction base="xsd:string">
    <xsd:enumeration value="Bachelor"/>
    <xsd:enumeration value="Master"/>
    <xsd:enumeration value="Doctor"/>
  </xsd:restriction>
</xsd:simpleType>
<xsd:element name="DIPLOMA" type="tDIPLOMA"/>
```

Lai darbiniekam ir *atribūts*: diploms.

Tiks izmantots tips tDIPLOMA.

<EMPLOYEE Diploma="Master"**></EMPLOYEE>**

```
<xsd:element name="EMPLOYEE">  
  <xsd:complexType>  
    <xsd:attribute name="Diploma"  
      type="tDIPLOMA"/>  
  </xsd:complexType>  
</xsd:element>
```

Lai telefona numurā ir *astoņi cipari*.

```
<TEL>12345678</TEL>
```

```
<xsd:simpleType name="tTEL">  
  <xsd:restriction base="xsd:string">  
    <xsd:pattern value="[0-9]{8}"/>  
  </xsd:restriction>  
</xsd:simpleType>  
  
<xsd:element name="TEL" type="tTEL"/>
```

Lai piemērā ar diplomiem izmanto *saīsinājumus*: B, M, D.

<DIPLOMA>B</DIPLOMA>

Iekļaut shēmā *komentārus* par saīsinājumu *jēgu*.

```

<xsd:simpleType name="tDIPLOMA">
  <xsd:restriction base="xsd:string">
    <xsd:enumeration value="B">
      <xsd:annotation>
        <xsd:documentation>
          Bachelor
        </xsd:documentation>
      </xsd:annotation>
    </xsd:enumeration>
  </xsd:restriction>
</xsd:simpleType>
  
```

Piezīme: **<xsd:documentation>** vietā var izmantot **<xsd:appinfo>**. Tad tas *nebūs* “klasiskais” komentārs.

Lai ir *veselu skaitļu* saraksts, kurā ir *četri* elementi.

<MARKS>9 10 9 9 **</MARKS>**

```
<xsd:simpleType name="tMarks">
  <xsd:restriction>
    <xsd:simpleType>
      <xsd:list
        itemType="xsd:positiveInteger"/>
    </xsd:simpleType>
    <xsd:length value="4"/>
  </xsd:restriction>
</xsd:simpleType>

<xsd:element name="MARKS" type="tMarks"/>
```

Lai krāsu var norādīt kā *teksta rindiņu* vai kā *skaitli*.

<COLOR>White**</COLOR>** vai **<COLOR>**1**</COLOR>**

```
<xsd:simpleType name="tColors">
```

```
  <xsd:union>
```

```
    <xsd:simpleType>
```

```
      <xsd:restriction
```

```
        base="xsd:nonNegativeInteger">
```

```
          <xsd:minInclusive value="0"/>
```

```
          <xsd:maxInclusive value="1"/>
```

```
        </xsd:restriction>
```

```
    </xsd:simpleType>
```

```
    <xsd:simpleType>
```

```
      <xsd:restriction base="xsd:string">
```

```
        <xsd:enumeration value="Black"/>
```

```
        <xsd:enumeration value="White" />
```

```
      </xsd:restriction>
```

```
    </xsd:simpleType>
```

```
  </xsd:union>
```

```
</xsd:simpleType>
```

```
<xsd:element name="COLOR" type="tColors"/>
```

Lai ir nepieciešams noformēt *loģiski saistītus* atribūtus kā grupu.

```
<EMPLOYEE Surname="Strods" Salary="400"/>
```

```
<xsd:attributeGroup name="tEmplAttr">
```

```
  <xsd:attribute name="Surname"
    type="xsd:string"/>
```

```
  <xsd:attribute name="Salary"
    type="xsd:float"/>
```

```
</xsd:attributeGroup>
```

```
<xsd:complexType name="tEMPLOYEE">
```

```
  <xsd:attributeGroup ref="tEmplAttr"/>
```

```
</xsd:complexType>
```

```
<xsd:element name="EMPLOYEE" type="tEMPLOYEE"/>
```

Lai `Surname` un `Salary` noformēti kā elementi un veido *grupu*. Savukārt, ir papildus atribūts: `Duty`.

```
<EMPLOYEE Duty="Programmer">
  <SURNAME>Strods</SURNAME>
  <SALARY>400</SALARY>
</EMPLOYEE>
```

```
<xsd:group name="tEMPL">
  <xsd:sequence>
    <xsd:element name="SURNAME" type="xsd:string"/>
    <xsd:element name="SALARY" type="xsd:float"/>
  <xsd:sequence>
</xsd:group>

<xsd:complexType name="tEMPLOYEE">
  <xsd:group ref="tEMPL"/>
  <xsd:attribute name="Duty" type="xsd:string"/>
</xsd:complexType>

<xsd:element name="EMPLOYEE" type="tEMPLOYEE"/>
```