Bab 9

Webcontrol pada ASP.NET (1)

9.1 Hierarki Webcontrol pada ASP.NET

Halaman web didalam ASP.NET memiliki keterkaitan satu dengan yang lain. Secara struktur hierarki, adalah sebuah kontrol halaman yang memiliki `webcontrol` dan template control. Webcontrol berfungsi sebagai kelas dasar yang mendefinisikan metode, properti dan acara umum untuk semua kontrol dalam *namespace* `System.Web.UI.WebControls`. Suatu Hierarki `webcontrol` dapat dilihat pada gambar 9.1.

Kelas WebControl menyediakan sifat, metode, dan peristiwa yang umum untuk semua kontrol Web server. Kelas ini dapat mengontrol tampilan dan perilaku dari pengendalian Web server dengan menetapkan sifat yang telah didefinisikan dalam kelas ini. Misalnya, warna dan font warna latar belakang kontrol dikendalikan dengan menggunakan properti `BackColor` dan `ForeColor`. Pada kontrol yang dapat menampilkan perbatasan, pilihan properti dapat mengontrol perbatasan lebar, gaya

perbatasan, dan warna perbatasan dengan menetapkan properti `BorderWidth , BorderStyle , dan BorderColor`. Ukuran kontrol pada Web server dapat dispesifikasikan dengan menggunakan properti Tinggi dan Lebar.

Gambar 9.1 Hierarki Webcontrol

Perilaku kontrol dapat ditentukan dengan mengatur properti tertentu, sebagai contoh : mengaktifkan dan menonaktifkan kontrol dengan menyetel properti `Enabled`. Tempat kontrol dalam urutan tab dikontrol dengan menyetel properti `TabIndex`. Agar memudahkan pemakai untuk mengetahui arti sebuah obyek, biasanya dipakai *tooltiptext.* Hal ini dapat menentukan tooltip untuk mengontrol dengan menyetel properti `tooltip`.

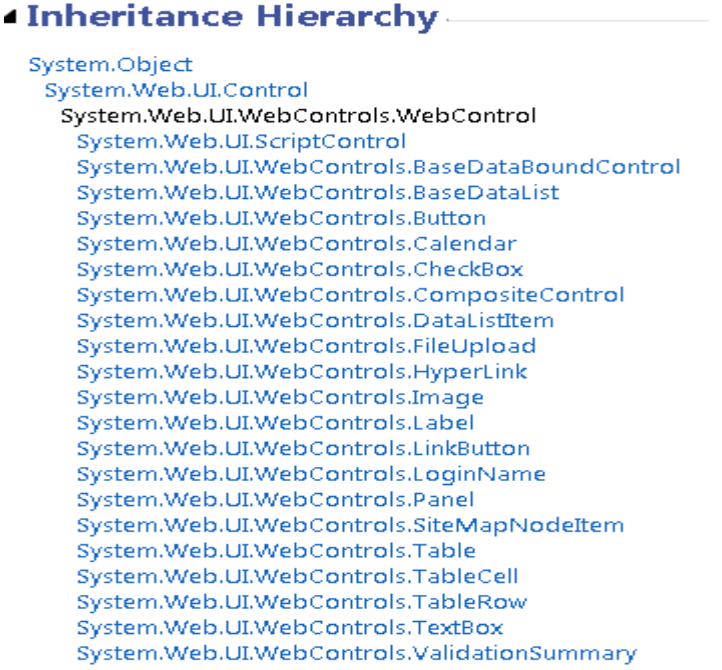
Tidak semua kontrol mendukung setiap properti yang didefinisikan di kelas ini. Untuk informasi spesifik tentang apakah sebuah properti didukung, hal ini dapat dilihat pada dokumentasi

untuk kontrol tertentu. Beberapa properti pada kelas ini dapat menghasilkan sebuah *render* yang berbeda bergantung pada browsernya, beberapa properti tidak berpengaruh sama sekali, namun pada browser yang lain, hal itu berpengaruh dan di *render*.

Properti `TagWriter` dari obyek `HttpBrowserCapabilities` object menentukan cara, bagaimana sebuah kontrol yang ada di Web Sever akan di *render.*

9.2 Kelas kontrol

Secara lengkap hierarki pewarisan dari sebuah obyek sampai kepada `webcontrol` menunjukkan bahwa `webcontrol` menerima pewarisan dari sebuah kontrol UI. Gambaran lengkap hierarki tsb terlihat pada gb 9.2



Gambar 9.2 Pewarisan WebControl

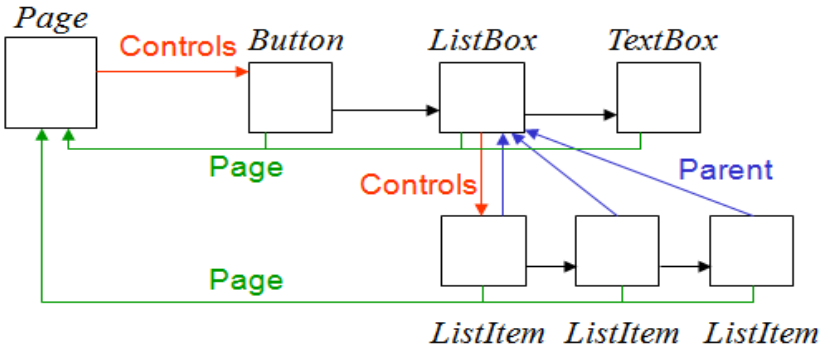
Secara gambaran umum kelas kontrol terbagi menjadi tiga bahasan yaitu; properti, metode dan events. Setiap kelas kontrol memiliki tiga komponen tsb. Secara umum kelas kontrol dilukiskan pada kode kelas dibawah ini

```
public class Control: ... {
       public virtual string ID { get; set; }
       public virtual ControlCollection Controls { get; }
       public virtual kontrol Parent { get; }
       public virtual Page Page { get; set; }
       public virtual bool Visible { get; set; }
       protected virtual StateBag ViewState { get; }
       public virtual bool EnableViewState { get; set; }
       ...
       public virtual bool HasControls();
       public virtual kontrol FindControl (string id);
       public virtual void DataBind();
       protected virtual void LoadViewState (object state);
       protected virtual object SaveViewState();
       protected virtual Render (HtmlTextWriter w);
       ...
       public event EventHandler Init;
       public event EventHandler Load;
       public event EventHandler DataBinding;
       public event EventHandler PreRender;
       public event EventHandler Unload;
       ...
}
```

Properti dari kode kelas diatas dimulai dari `ID` sampai `EnabledViewState`, metode dimulai dari `HasControls` sampai `Render`, dan events dimulai dari `Init` sampai `Unload`.

Sebuah kelas kontrol dapat memiliki kontrol lain yang lebih dari satu dan kelas kontrol memiliki hubungan dengan induknya. Gambaran ini dapat diperlihat pada gambar 9.3, sebuah page memiliki kontrol tombol, ListBox dan TextBox. ListBox memiliki kontrol ListItem, sehingga dapat pula disebut bahwa ListItem memiliki induk ListBox.

Kesemua kontrol itu merujuk pada sebuah page yang pada kelas kontrol ditunjukkan sebagai property: `public virtual Page Page { get; set; }`



Gambar 9.3 Hubungan antar kontrol

Didalam kelas kontrol terdapat properti ViewState yang berarti mendapatkan sebuah diktionari dari informasi state yang mengijinkan program untuk menyimpan dan mengambil kembali terhadap nilai viewstate pada kontrol server di beberapa permintaan untuk halaman yang sama.

Contoh berikut menunjukkan menerapkan properti Text yang menyimpan dan mengambil nilai dari properti `viewstate`.

```
public String Text
{
 Get {object o = ViewState["Text"];
     return (o == null)? String.Empty : (string)o;}
 set {ViewState["Text"] = value;}}
```

Contoh yang lain adalah mengecek berapa kali sebuah tombol ditekan, dengan memanfaatkan `ViewState` yang terdapat pada kelas button:

```
public void ButtonClick (object Button, EventArgs e) {
int clicks = ViewState["nClicks"] == null ? 0 : (int)
ViewState["nClicks"];
ViewState["nClicks"] = ++clicks;}
```

Pertama kali ketika sebuah tombol ditekan, maka program akan membuat variable `clicks` dengan nilai yang bergantung pada pemeriksaan `ViewState` yang bernama `nClicks`, jika masih null, maka nilai variabel `clicks` bernilai 0, jika `ViewState["nClicks"]` sudah ada, maka nilainya akan diletakkan sebagai nilai awal dari variabel `clicks`. Kemudian nilai variabel clicks dinaikkan satu dengan perintah `++clicks`.

Keuntungan dari pemakain `ViewState` adalah:

- Programmer dapat menyimpan data dalam `ViewState`
- Suatu `ViewState` disimpan sebagai `hiddenfield` dari halaman HTML
- Tipe dari `ViewState` adalah `protected`, yang berguna untuk mendapatkan informasi yang sama walaupun terjadi beberapa akses .

9.3 Kelas Webcontrol

Berfungsi sebagai kelas dasar yang mendefinisikan metode, properti dan `events` untuk semua kontrol dalam `namespace System.Web.UI.WebControls`. Secara umum kelas `WebControl` memiliki properti yang akan diwariskan ke masing-masing kontrol yang menjadi anaknya.

```
public class WebControl: kontrol {
       public virtual Unit Width { get; set; }
       public virtual Unit Height { get; set; }
       public virtual FontInfo Font { get; set; }
       public virtual Color ForeColor { get; set; }
       public virtual Color BackColor { get; set; }
       public virtual Unit BorderWidth { get; set; }
       public virtual Color BorderColor { get; set; }
       public virtual BorderStyle BorderStyle { get; set; }
       public virtual bool Enabled { get; set; }
       public virtual short TabIndex { get; set; }
       public virtual string ToolTip { get; set; }
       ...}
```

Setiap kontrol yang berinduk pada WebControl pasti memiliki properti warisan seperti misalnya Width, Height, Font dan sebagainya.

9.3.1. Ukuran Obyek

Ukuran dari masing-masing properti mengikuti Strukur Unit yang didefiniskan dibawah ini (default adalah pixel):

```
public struct Unit {
       public Unit (double value, UnitType type);
       public double Value { get; }
       public UnitType Type { get; }
       ...
}
public enum UnitType {Cm, Em, Ex, Inch,
       Mm, Percentage, Pica, Pixel, Point
}
```

9.3.2. Warna Obyek

Nilai properti pada pewarnaan obyek, mengikuti struktur warna yang didefinisikan dalam `namespace: System.Drawing`, seperti terlihat dibawah ini:

```
public struct Color {
       public static Color Blue { get; }
       public static Color Red { get; }
       public static Color Yellow { get; }
       ...
       public static Color FromArgb (int R, int G, int B);
}
```

Contoh pengaturan properti memakai ukuran dan pewarna dapat dilihat pada kode dibawah ini :

```
<asp:TextBox ID="tb" Width="100" ... />
<asp:TextBox ID="tb" Width="10cm" ... />
<asp:TextBox ForeColor="Red" ... />
```

Penulisan diatas dapat pula dituliskan dalam kode skrip program:

```
tb.Width = 100;  // default: Pixel
tb.Width = new Unit(10, UnitType.Cm);
tb.ForeColor = Color.Red;
```

9.3.3. Jenis Huruf Obyek

Nilai properti pada jenis huruf pada suatu obyek, mengikuti struktur jenis huruf seperti dibawah ini:

```
public sealed class FontInfo {
       public string Name { get; set; }
       public FontUnit Size { get; set; }
       public bool Bold { get; set; }
       public bool Italic { get; set; }
       public bool Underline { get; set; }
       ...
}
public struct FontUnit {
       public FontUnit (Unit size);
       public FontUnit (FontSize size);
       public Unit Unit { get; }
       public FontSize Type { get; }
       ...
}
public enum FontSize { AsUnit, XSmall,
       Small, Medium, Large, XLarge, ... }
```

Contoh pengaturan properti memakai `font` dapat dilihat pada kode dibawah ini :

```
<asp:Button ID="b1" Font-Name="Arial"
       Font-Size="Large" Font-Bold="true" .../>
<asp:Button ID="b2" Font-Name="Times"
       Font-Size="12px" Font-Italic="true" ... />
```

Penulisan diatas dapat pula dituliskan dalam kode skrip program:

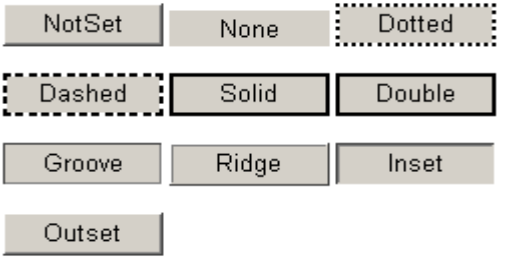
```
b1.Font.Name = "Arial";
b1.Font.Size = new FontUnit(FontSize.Large);
b1.Font.Bold = true;
b2.Font.Name = "Times";
b2.Font.Size = new FontUnit(12);
b2.Font.Italic = true;
```

9.3.4. Jenis Batas Obyek

Nilai properti pada jenis batas pada suatu obyek, mengikuti enum `BorderStyle` seperti dibawah ini:

```
public enum BorderStyle {
       NotSet, None, Dotted, Dashed,Solid, Double, Groove,
       Ridge, Inset, OutSet}
```

Contoh hasil secara visual dari pilihan `BorderStyle` diperlihatkan pada gambar 9.4



Gambar 9.4 Visualisasi BorderStyle

9.3.5. Pengaktifan Obyek

Nilai properti `enabled` pada suatu obyek, merupakan sebuah keadaan sebuah obyek, jika `enabled` bernilai `True`, maka obyek tsb dapat dikenai sebuah aksi, jika bernilai `False`, maka obyek tersebut dalam keadaan tidak bias dikenai aksi oleh pemakai.

Contoh obyek button dibawah ini:

```
<asp:Button Enabled="false" ... />
```

Kode tsb menampilkan kontrol button, tapi menonaktifkan kontrol tsb, sehingga pemakai tidak dapat memberikan aksi, baik menekan tombol atau menekan klik tombol tsb, hasil secara visualisasi diperlihatkan pada gambar 9.5

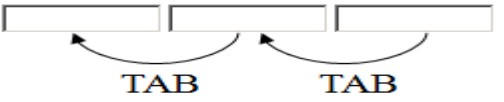
Gambar 9.5 Visualisasi Enabled

9.3.6. Urutan Obyek

Nilai properti `tabindex` pada suatu obyek, merupakan urutan dari sekumpulan obyek dalam `webform`. Hal ini sangat berpengaruh pada peralihan posisi kursor dari sebuah kontrol ke kontrol lain, misalnya perpindahan dengan memakai tombol tab. Contoh sederhana pemakaian `tabindex`:

```
<asp:TextBox TabIndex="3" ... />
<asp:TextBox TabIndex="2" ... />
<asp:TextBox TabIndex="1" ... />
```

Hasilnya adalah tiga buah texbox dengan urutan textbox paling kanan sampai ke kiri, seperti diperlihatkan pada gambar 9.6



Gambar 9.6 Visualisasi TabIndex

9.4 Kelas Button

Merupakan kontrol berupa tombol pada Windows, yang bereaksi terhadap event Klik pada `ButtonBase`. Pemakaian kontrol `Button` dipakai untuk membuat tombol pada halaman Web yang memungkinkan pengguna untuk mengirim halaman ke server. Kontrol memicu sebuah events di kode server yang dapat diprogram untuk menangani dan menanggapi `postback` tersebut. Hal ini juga dapat meningkatkan sebuah events pada skrip klien bahwa terdapat sebuah proses yang dapat menangani sebelum halaman tersebut diposting.

Suatu ASP.NET mencakup beberapa jenis kontrol tombol, masing-masing yang muncul berbeda pada halaman Web. Jenis kontrol tombol antara lain kontrol `Button`, yang membuat sebagai tombol push, sedangkan kontrol `LinkButton`, yang membuat sebagai `link`, dan kontrol `ImageButton`, yang membuat sebagai foto;

dan kontrol `imagemap`, yang memungkinkan pembuatan grafik yang memiliki `hotspot` bahwa pengguna bisa meng-klik.

Secara default, semua kontrol tombol adalah sebuah submit pada halaman saat diklik, selain itu dapat juga menggunakan kontrol `HtmlButton` dan `HtmlInputButton` untuk membuat tombol pada halaman yang diprogram dalam kode server.

Sebuah tombol memiliki nama perintah yang berhubungan dengan tombol. Hal ini memungkinkan untuk membuat beberapa kontrol tombol pada halaman web dan secara pemrograman menentukan kontrol `Button` diklik untuk diproses pada metode yang sama. Properti `CommandArgument` dapat dipakai untuk memberikan informasi kepada pemakai tentang arti tombok tsb. Sebuah `event handler` dapat dilibatkan secara pemrograman untuk mengendalikan tindakan yang dilakukan ketika tombol Command diklik.

Secara umum kelas button menerima warisan dari webcontrol, sehingga kode secara umum ditunjukkan di bawah ini:

```
public class Button: WebControl {
       //--- properties
       public string Text { get; set; }
       public string CommandName { get; set; }
       public string CommandArgument { get; set; }
       public bool CausesValidation { get; set; }
       //--- events
       public event EventHandler Click;
       public event CommandEventHandler Command;
}
```

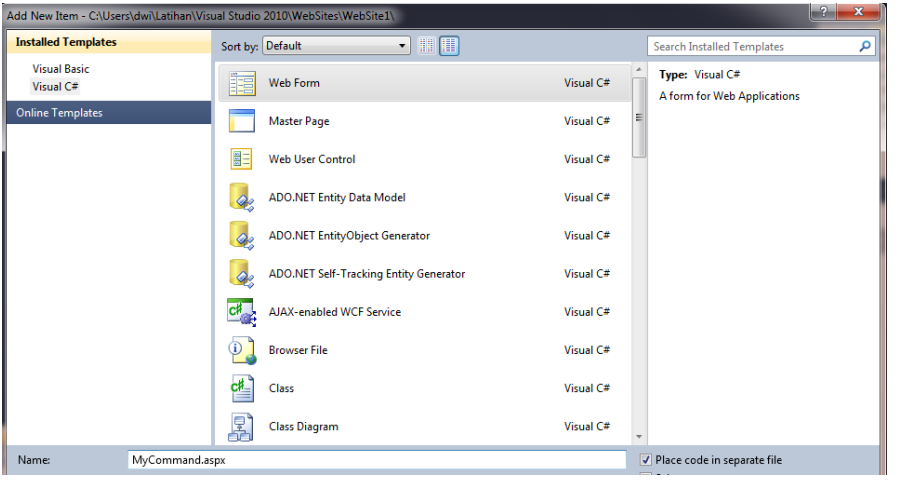
Contoh pemakain kelas button adalah `<asp:Button Text="click me" OnClick="DoClick" Runat="server" />`
Di web server, program yang menerima events dari klik berupa sebuah metode yang menerima delegasi dari *EventHandler*

```
public void DoClick (object sender, EventArgs e) {...}
```

Sekumpulan kelas button dapat diciptakan dengan sebuah metode yang menerima *eventHandler*. Hal ini disebut sebagai ***Command* Event**.

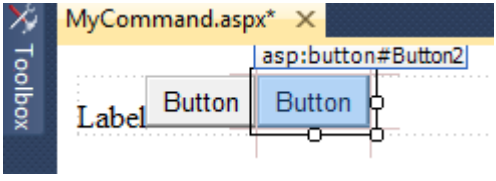
Contoh sebuah ***Command*** **Event** adalah dua tombol dengan satu fungsi yang sama. Langkah-langkah untuk membuat contoh program tsb adalah:

1. Buka VS2010 dan buat situs baru (atau buka situs yang telah ada pada bab sebelumnya)
2. Tambahkan file kode program baru dengan menekan CTL-N, pilih template webform beri nama dengan MyCommand, seperti pada gambar 9.7 dibawah ini



Gambar 9.7 Jendela Pembuatan MyCommand.aspx

3. Pada mode desain, buat label dan dua tombol, sehingga tampil seperti rancanngan pada gambar 9.8



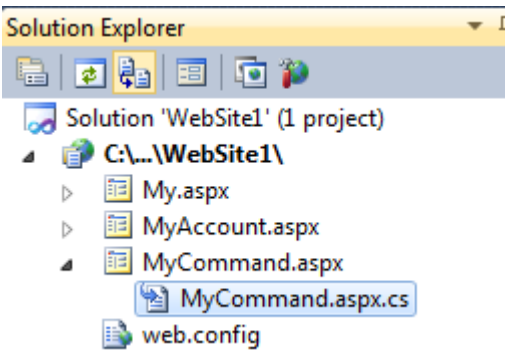
Gambar 9.8 Rancangan MyCommand.aspx

4. Ubah properti sehingga program selengkapnya menjadi skrip dibawah ini

```
<form Runat="server">
<asp:Label ID="label" Text="100.00" Runat="server" />
<br><br>
<asp:Button Text="+ 10%"
CommandName="add" CommandArgument="0.1"
 OnCommand="DoCommand" Runat="server" />
<asp:Button Text="- 5%"
 CommandName="sub" CommandArgument="0.05"
 OnCommand="DoCommand" Runat="server" />
</form>
```

   Terlihat pada contoh diatas terdapat dua button dengan memanggil satu metode yaitu `DoCommnad`. Didalam metode tsb, harus dicek terlebih dahulu berdasarkan `CommandName` untuk masing-masing aksi yang akan dilakukan

5. Setelah kode program MyCommand.aspx sudah selesai, kemudian masuk kepada file MyCommand.aspx.cs: buka file tsb, dari jendela solution explorer, seperti gambar 9.9 dibawah ini:

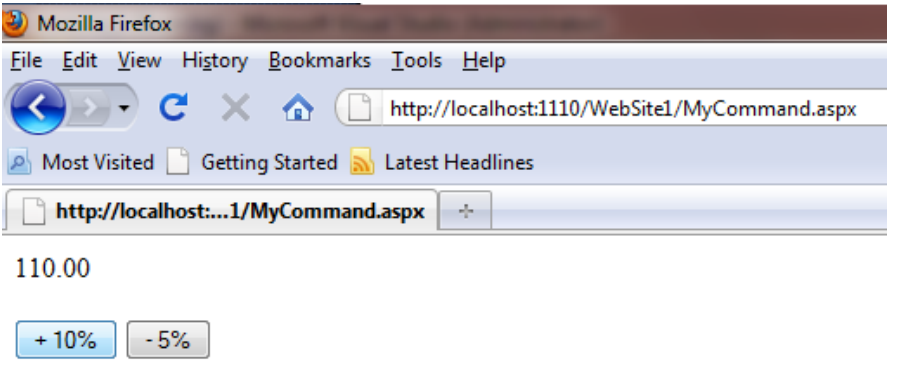


Gambar 9.9 Membuka File MyCommand.aspx.cs

6. Ketika program dibawah ini yang berisi aksi untuk penanganan pada penekanan dua tombol

```
public void DoCommand (object sender, CommandEventArgs e)
{double total = Convert.ToDouble(label.Text);
if (e.CommandName == "add")
 total += total * Convert.ToDouble(e.CommandArgument);
else if (e.CommandName == "sub")
 total -= total * Convert.ToDouble(e.CommandArgument);
label.Text = total.ToString("f2");
}
```

7. Jalankan progran tsb, dan hasil dari contoh program diatas, diperlihatkan pada gambar 9.10



Gambar 9.10 Visualisasi Command Event

Tombol pertama ditekan, maka nilai awal akan bertambah 10%, sebaliknya tombol yang kedua ditekan, nilai akan berkurang 5%, dua tombol dengan satu metode yaitu `DoCommand`, dimana terdapat seleksi dari `commandName` yang membedakan dua tombol tsb

9.4 Kelas TextBox

Merupakan kelas yang sederajat dengan kelas button, berguna sebagai tampilan kotak teks kontrol untuk input pengguna. Kontrol server TextBox adalah kontrol input yang

memungkinkan pengguna memasukkan teks. Secara default, `textmode` diatur ke `TextBoxMode.SingleLine`, yang menampilkan garis kotak teks-tunggal. Pilihan model ini dapat diubah untuk menampilkan kotak teks `multiline` atau sebagai kotak teks yang tersembunyi sebagai masukan pengguna dengan mengubah nilai dari properti `textmode` untuk `TextBoxMode.MultiLine` atau `TextBoxMode.Password`. Teks yang ditampilkan dalam kontrol TextBox ditetapkan atau ditentukan dengan menggunakan Teks properti.

Kontrol TextBox berisi beberapa properti yang memungkinkan untuk mengontrol tampilan. Ukuran lebar layar dari kotak teks, dalam karakter, ditentukan oleh properti `column`. Jika kontrol TextBox adalah kotak teks `multiline`, jumlah baris ini akan menampilkan data yang ditentukan oleh properti `row`. Untuk menampilkan teks yang tertata secara *wrap*, dapat diatur properti *wrap* menjadi true.

Textbox dapat pula ditentukan bagaimana data yang dimasukkan pada kontrol TextBox dengan menetapkan beberapa properti. Untuk mencegah teks yang ditampilkan pada kontrol tidak diubah, properti readonly diset menjadi true. Batasan input pengguna ke sejumlah karakter tertentu, dapat diatur properti MaxLength.

Secara umum kelas TextBox menerima warisan dari webcontrol, sehingga kode secara umum ditunjukkan di bawah ini:

```
public class TextBox: WebControl {
       //--- properties
       public virtual string Text { get; set; }
       public virtual TextBoxMode TextMode { get; set; }
       public virtual int MaxLength { get; set; }
       public virtual int Columns {get; set; }
       public virtual int Rows { get; set; }
       public virtual bool Wrap { get; set; }
       public virtual bool ReadOnly { get; set; }
       public virtual bool AutoPostBack { get; set; }
       //--- events
       public event EventHandler TextChanged;
}
```

Nilai `TextMode` merupakan sebuah enumerasi dari tiga nilai yaitu:

```
public enum TextBoxMode {
       MultiLine, Password, SingleLine
}
```

Contoh pemakaian kelas textbox adalah:

```
<asp:TextBox Text="sample" Runat="server" />
<asp:TextBox TextMode="Password" MaxLength="10" Runat="server"
/>
<asp:TextBox TextMode="MultiLine"
       Rows="2" Columns="15" Wrap="true" Runat="server" />
line 1
line 2
line 3
</asp:TextBox>
```

Hasil dari pemakaian kelas textbox diperlihatkan pada gambar 9.7

Gambar 9.7 Visualisasi Pemakaian Kelas TextBox