HARMINCKETTEDIK FEJEZET

# ASP.NET-vezérlőelemek, -témák és -mesteroldalak

Az előző fejezetben az ASP.NET Page objektumok formázására és viselkedésére helyeztük a hangsúlyt. A következőkben az oldalak felhasználói felületeit felépítő *webes vezérlőelemek* részleteit vizsgáljuk meg. Az ASP.NET webes vezérlőelemek általános megvizsgálása után érthetővé válik, hogy hogyan használunk jó néhány felhasználóifelület-elemet, köztük az ellenőrző és adatközpontú vezérlőelemeket.

A fejezet második fele a *mesteroldalak* szerepét tekinti át, és bemutatja, hogyan lehet azok segítségével egyszerűen definiálni olyan közös felhasználóifelület-vázat, amelyet a webhelyünk oldalain többször is alkalmazhatunk. A célunk az, hogy bemutassuk, hogyan használjunk *témákat* az oldalakra a vezérlőelemek következetes megjelenéséhez. Az ASP.NET témamodulja kiszolgálóoldali alternatívát nyújt a hagyományos ügyféloldali stíluslapokkal szemben.

## A webes vezérlőelemek viselkedésének megértése

Az ASP.NET egyik fő erénye, hogy képes összeállítani az oldalak felhasználói felületét a `System.Web.UI.WebControls` névtérben definiált típusokkal. Ezek a vezérlőelemek (amelyek *server controls, web controls* vagy *Web Form controls* néven ismertek) rendkívül hasznosak, ugyanis automatikusan generálják a szükséges HTML-kódot a böngésző számára, és a webkiszolgáló számára feldolgozható eseménykészlettel szolgálnak. Továbbá mivel a `System.Web.UI.WebControls` névtérben megvannak az egyes ASP.NET-vezérlőelemeknek megfelelő osztályok, azok objektumorientáltan kezelhetők.

Amikor a Visual Studio 2008 Properties ablakának segítségével állítjuk be egy webes vezérlőelem tulajdonságait, a módosításainkat az `*.aspx` fájl adott elem deklarációjának nyitó tagjében rögzítjük név/érték párok sorozataként.

Így, ha hozzáadunk egy új TextBoxot az adott *.aspx fájl tervezőjéhez, és módosítjuk az ID, BorderStyle, BorderWidth, BackColor és Text tulajdonságokat, a nyitó <asp:TextBox> tag ennek megfelelően változik (a Text értéke a TextBox hatókörén belüli szöveg lesz):

```
<asp:TextBox ID="txtNameTextBox" runat="server"
     BackColor="#C0FFC0" BorderStyle="Dotted"BorderWidth="5px">
     Enter Your Name
</asp:TextBox>
```

Mivel a webes vezérlőelemek HTML-deklarációja végül a System.Web.UI.WebControls névtér egyik tagváltozója lesz (az előző fejezetben tárgyalt, dinamikus fordítási ciklus révén), képesek vagyunk ennek a típusnak a tagjaival kommunikálni a kiszolgálóoldali <script> blokkon keresztül vagy az oldal mögöttes kódfájljának a segítségével. Ha például egy adott Button típusnak implementáltuk a Click eseményét, a TextBox háttérszínét a következőképpen módosíthatjuk:

```
partial class _Default : System.Web.UI.Page
{
  protected void btnChangeTextBoxColor_Click(object sender,
                                             System.EventArgs e)
  {
    // Kódból módosítjuk a TextBox színét.
    this.txtNameTextBox.BackColor = Drawing.Color.DarkBlue;
  }
}
```

Az összes ASP.NET webes vezérlőelem végül is a System.Web.UI.WebControls.WebControl nevű közös ősosztályból, a WebControl pedig a System.Web.UI.Control osztályból származik (amely pedig a System.Object osztályból). Control és a WebControl számos olyan tulajdonságot definiálnak, amelyek közösek az összes kiszolgálóoldali vezérlőelem számára. Mielőtt megismernénk az öröklött működést, formalizáljuk a kiszolgálóoldali események kezelésének jelentését.

## Kiszolgálóoldali események kezelése

A web jelenlegi állapota mellett lehetetlen elkerülni a böngésző/webkiszolgáló interakció alapvető sajátosságait. Amikor ez a két entitás kommunikál, mindig egy mögöttes, állapotmentes, HTTP kérés/válasz ciklus jön létre. Amíg az ASP.NET kiszolgálójának vezérlőelemei sokat tesznek azért, hogy elkerülhes-

sük a nyers HTTP protokollt, ha a webet eseményvezérelt egyedként kezeljük, az nem más, mint a CLR „szemfényvesztő" produkciója, és nem egyezik meg a windowsos felhasználói felületek eseményvezérelt modelljével.

A `System.Windows.Forms`, a `System.Windows.Controls` és a `System.Web.UI.WebControls` névterek például ugyanolyan egyszerű elnevezésű típusokat definiálnak (`Button`, `TextBox`, `Label` stb.), mégsem ugyanazokat az eseménykészleteket biztosítják. A Web Form `Button` típus segítségével például nem lehet kiszolgálóoldali `MouseMove` eseményt kezelni akkor, amikor a felhasználó mozgatja a kurzort. Ez nyilvánvalóan hasznos. (Ki szeretne ugyanis minden alkalommal adatot visszaküldeni a kiszolgálónak, amikor megmozdítja az egerét a böngészőben?)

Végezetül, egy adott ASP.NET webes vezérlőelem korlátozott számú eseménykészletet biztosít, ezek mindegyike végül visszaküldést (postback) eredményez a webkiszolgáló felé. Az események ügyféloldali feldolgozásához nekünk kell elkészíteni azt az *ügyféloldali* JavaScript/VBScript kódot, amit a kérést intéző böngészőnek futtatnia kell. Mivel az ASP.NET elsősorban kiszolgálóoldali technológia, az ügyféloldali szkriptírást itt nem részletezzük.

**Megjegyzés** A Visual Studio 2008 esetében ugyanúgy kezelhetjük egy adott webes vezérlőelem eseményét, mint egy Windows Forms vezérlőelemét. Válasszuk ki a tervezőben a vezérlőelemet, majd kattintsunk a „villám" ikonra a Properties ablakban.

## Az AutoPostBack tulajdonság

Sok ASP.NET webes vezérlőelem támogatja az `AutoPostBack` nevű tulajdonságot (legjelentősebbek a jelölőnégyzet, a rádiógomb és a szövegdoboz-vezérlőelemek, valamint az absztrakt `ListControl` típusból származó elemek). Ennek a tulajdonságnak az alapértelmezett értéke hamis, így kikapcsolja a kiszolgálóoldali események automatikus feldolgozását (még akkor is, ha a mögöttes kódfájlban feliratkoztunk az eseményre). A legtöbb esetben erre a viselkedésre van szükség, hiszen az olyan felhasználóifelület-elemek, mint a jelölőnégyzetek, nem igényelnek visszaküldést (ugyanis az oldalobjektum a jóval természetesebb gomb kattintási eseménykezelőjével megtudhatja a vezérlőelem állapotát).

Ha azonban szeretnénk, hogy ezek a vezérlőelemek visszaszóljanak a kiszolgálóoldali eseménykezelőknek, akkor az AutoPostBack értékét egyszerűen állítsuk igazra. Ez a módszer nagyon hasznos lehet akkor, ha a vezérlőelem állapota alapján szeretnénk beállítani az oldal egy másik elemének az értékét. Ennek szemléltetéséhez tételezzük fel, hogy a weboldalunk egy txtAutoPostback nevű szövegdoboz, és egy lstTextBoxData nevű ListBox vezérlőelemet tartalmaz. Az ehhez tartozó kód az *.aspx fájlban a következő:

```
<form id="form1" runat="server">
  <asp:TextBox ID="txtAutoPostback" runat="server"></asp:TextBox>
  <br/>
  <asp:ListBox ID="lstTextBoxData" runat="server"></asp:ListBox>
</form>
```

Kezeljük a TextBox TextChanged eseményét, és a kiszolgálóoldali eseménykezelőben a ListBox értéke felveszi a TextBox jelenlegi értékét:

```
partial class _Default : System.Web.UI.Page
{
  protected void txtAutoPostback_TextChanged(object sender,
                                             System.EventArgs e)
  {
    lstTextBoxData.Items.Add(txtAutoPostback.Text);
  }
}
```

Ha futtatjuk az alkalmazást, azt tapasztaljuk, hogy ha beírunk valamit a szövegdobozba, nem történik semmi. Továbbá, ha írunk valamit a szövegdobozba, és a következő vezérlőelemre lépünk, ugyancsak nem történik semmi. Ez azért van, mert a szövegdoboz AutoPostBack tulajdonságának alapértelmezett értéke hamis. Viszont, ha az igaz értéket adjuk ennek a tulajdonságnak,

```
<asp:TextBox ID="txtAutoPostback"
     runat="server" AutoPostBack="true">
</asp:TextBox>
```

és továbblépünk a szövegdobozról (vagy megnyomjuk az Enter billentyűt), a ListBox automatikusan felveszi a szövegdoboz jelenlegi értékét. Biztosak lehetünk benne, hogy a más vezérlők értékeire épülő vezérlőelemek feltöltésén kívül általában nem szükséges az AutoPostBack tulajdonság állapotát módosítani (de még ebben az esetben is néha elég egy kliensszkriptet végrehajtani, ezzel teljesen kizárva a kiszolgálóval való interakció szükségességét).

# A System.Web.UI.Control típus

A `System.Web.UI.Control` alaposztály számos olyan tulajdonságot, metódust és eseményt definiál, amelyek engedélyezik a webes vezérlőelemek alapvető (általában nem grafikus felületi) funkcióival folytatott kommunikációt. A 32.1. táblázat a teljesség igénye nélkül, néhány jelentősebb tagot tartalmaz.

| Tag | Jelentés |
|---|---|
| `Controls` | Ennek a tulajdonságnak az értéke olyan `ControlCollection` objektum, amely tartalmazza az aktuális vezérlőelem gyermek-vezérlőelemeit. |
| `DataBind()` | Ez a metódus egy adatforrást köt a hívott kiszolgáló-vezérlőelemhez, illetve annak összes gyermek-vezérlőeleméhez. |
| `EnableTheming` | Ettől a tulajdonságtól függ, hogy a vezérlőelem támogatja-e a témákat. |
| `HasControls()` | Ez a metódus határozza meg, hogy a kiszolgáló-vezérlőelem tartalmaz-e gyermek-vezérlőelemeket. |
| `ID` | Ez a tulajdonság a kiszolgáló-vezérlőelemhez rendelt programozott azonosítót kérdezi le vagy állítja be. |
| `Page` | Ez a tulajdonság visszaadja a kiszolgáló-vezérlőelemet tartalmazó `Page` példány referenciáját. |
| `Parent` | Ez a tulajdonság visszaadja a kiszolgáló vezérlőelem szülő-vezérlőelemének referenciáját az oldal vezérlőelem-hierarchiájában. |
| `SkinID` | Ez a tulajdonság lekérdezi vagy beállítja a vezérlőelemre alkalmazott „arculatot". Az ASP.NET-ben így a vezérlőelemek megjelenését arculatok segítségével egyszerűen alakíthatjuk ki. |
| `Visible` | Ez a tulajdonság azt kérdezi le, vagy állítja be, hogy a kiszolgáló-vezérlőelem felhasználóifelület-elemeként megjelenik-e az oldalon. |

**32.1. táblázat:** *A System.Web.UI.Control néhány tagja*

## Vezérlőelemek felsorolása

Az első, amit a `System.Web.UI.Control` esetében megvizsgálunk, az a tény, hogy az összes webes vezérlőelem (köztük maga az oldal) egyedi vezérlőelem-gyűjteményt örököl, amely a `Controls` tulajdonságon keresztül érhető el. A Windows Forms-alkalmazásokban tapasztaltakhoz hasonlóan, a `Controls` tulajdonsággal férhetünk hozzá egy erősen típusos, `WebControl`-leszármazott típust tartalmazó gyűjteményhez. Mint a többi .NET-gyűjtemény esetében, ebben is futásidőben, dinamikusan adhatunk hozzá, szúrhatunk be és távolíthatunk el elemeket.

Bár technikailag lehetséges webes vezérlőelemeket közvetlenül hozzáadni `Page`-leszármazott típusokhoz, ennél egyszerűbb (és jóval hatékonyabb), ha Paneleket használunk. A Panel osztály olyan vezérlőelemtároló, amelyet a végfelhasználó vagy láthat, vagy nem láthat (a `Visible` és a `BorderStyle` tulajdonságainak értékétől függően).

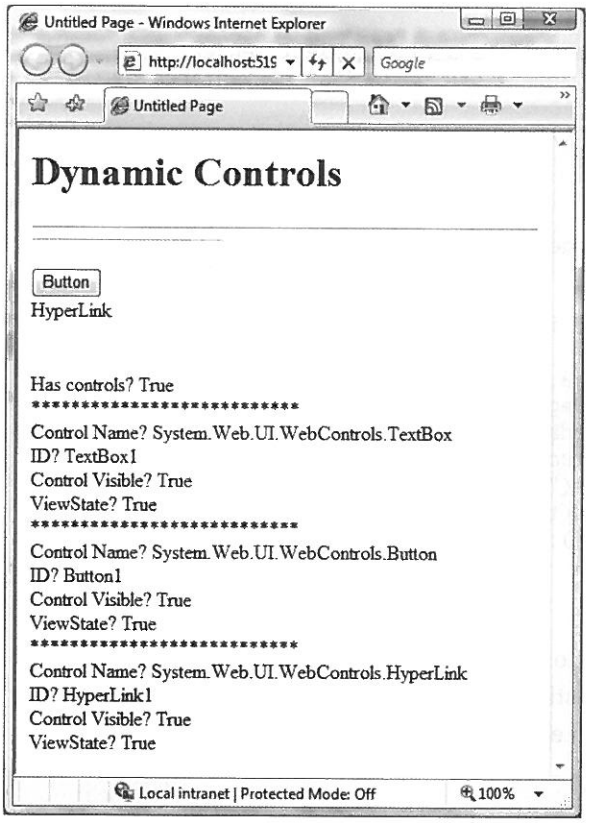
Ennek szemléltetésére, hozzunk létre egy új, DynamicCtrls nevű webhelyet. A Visual Studio 2008 oldaltervezőjében adjunk hozzá egy myPanel nevű panelt, amely egy tetszőleges elnevezésű `TextBox`, `Button` és `HyperLink` vezérlőket tartalmaz (ügyeljünk arra, hogy a tervezőben a `Panel` típus felhasználói felületére kell húznunk a belső elemeket). Ha ezt megtettük, az `*.aspx` fájl `<form>` eleme az alábbiak szerint módosul:

```
<asp:Panel ID="myPanel" runat="server" Height="50px" Width="125px">
  <asp:TextBox ID="TextBox1" runat="server"></asp:TextBox><br/>
  <asp:Button ID="Button1" runat="server" Text="Button"/><br/>
  <asp:HyperLink ID="HyperLink1" runat="server">HyperLink
  </asp:HyperLink>
</asp:Panel>
```

Ezután helyezzünk egy `lblControlInfo` nevű címkét a panel hatókörén kívülre, amelyben megjelenik a kimenet. Tételezzük fel, hogy a `Page_Load()` eseményben a panel összes vezérlőelemének listáját szeretnénk megkapni, az eredményt pedig hozzá szeretnénk rendelni a `lblControlInfo` nevű címkéhez:

```
public partial class _Default : System.Web.UI.Page
{
  private void ListControlsInPanel()
  {
    string theInfo = "";
    theInfo = string.Format("Has controls? {0} <br/>",
                            myPanel.HasControls());
    foreach (Control c in myPanel.Controls)
    {
```

```
        if (!object.ReferenceEquals(c.GetType(),
                    typeof(System.Web.UI.LiteralControl)))
        {
          theInfo += "**************************<br/>";
          theInfo += string.Format("Control Name? {0} <br/>",
                                   c.ToString());
          theInfo += string.Format("ID? {0} <br>", c.ID);
          theInfo += string.Format("Control Visible? {0} <br/>",
                                   c.Visible);
          theInfo += string.Format("ViewState? {0} <br/>",
                                   c.EnableViewState);
        }
      }
      lblControlInfo.Text = theInfo;
    }

    protected void Page_Load(object sender, System.EventArgs e)
    {
      ListControlsInPanel();
    }
}
```




**32.1. ábra:** *A panel vezérlőelemeinek felsorolása*

Végigiteráluk a panel `WebControl` elemein, és ellenőrizzük, hogy az aktuális típus `System.Web.UI.LiteralControl` típusú-e. Ezzel a típussal literális HTML-címkéket és -tartalmakat jelenítünk meg (mint a `<br>`, szövegliterálok stb.). Ha ezt az ellenőrzést nem végezzük el, meglepetésünkre a panel hatókörében mind a hét típus megjelenhet (a korábban ismertetett `*.aspx` deklaráció miatt). Ha a típus nem literális HTML-tartalom, jelenítsünk meg különféle statisztikákat a vezérlőelemről. A 32.1. ábrán láthatjuk az eredményt.

## Vezérlőelemek dinamikus hozzáadása (és törlése)

Mit kell tenni, ha futás közben szeretnénk módosítani a Panel tartalmát? Adjunk az aktuális oldalhoz egy `btnAddWidgets` nevű gombot, amellyel három új szövegdobozt adunk hozzá dinamikusan a panelhez, valamint egy másik, `btnRemovePanelItems` nevű gombot, amely törli az összes vezérlőelemet a panelből. Az alábbiakban látható mindkét gomb kattintási eseménykezelője:

```
protected void btnRemovePanelItems_Click(object sender,
                                         System.EventArgs e)
{
  myPanel.Controls.Clear();
  ListControlsInPanel();
}

protected void btnAddWidgets_Click(object sender,
                                   System.EventArgs e)
{
  for (int i = 0; i < 3; i++)
  {
    // Nevet rendel hozzá, így később
    // kapjuk meg a szövegértéket
    // a bemeneti űrlapadat segítségével.
    TextBox t = new TextBox();
    t.ID = string.Format("newTextBox{0}", i);
    myPanel.Controls.Add(t);
    ListControlsInPanel();
  }
}
```

Minden egyes szövegdobozhoz hozzárendeltünk egy egyedi azonosítót (pl. newTextBox1, newTextBox2 stb.) azért, hogy a `HttpRequest.Form` gyűjtemény segítségével, programozottan elérhessünk a bennük lévő szövegekhez.

Ahhoz, hogy megszerezzük az ezekben a dinamikusan generált szövegdobozokban levő értékeket, adjunk hozzá még egy gombot és címkét a felhasználói felülethez. A gomb kattintási eseménykezelőjében iteráljunk végig a `HttpRequest.NameValueCollection` típus elemein (amely a `HttpRequest.Form` révén érhető el), és a szöveges adatokat fűzzük hozzá a helyi hatókörű `System.String` típushoz. Ha teljesen feldolgoztuk a gyűjteményt, a sztringet rendeljük hozzá az új, `lblTextBoxText` nevű címke Text tulajdonságához:

```
protected void btnGetTextBoxValues_Click(object sender,
                                         System.EventArgs e)
{
  string textBoxValues = "";
  for (int i = 0; i < Request.Form.Count; i++)
  {
    textBoxValues += string.Format("<li>{0}</li><br/>",
                                   Request.Form[i]);
  }
  lblTextBoxText.Text = textBoxValues;
}
```

Ha lefuttatjuk az alkalmazást, meg tudjuk nézni az egyes szövegmezők tartalmát, köztük néhány meglehetősen hosszú (nem olvasható) sztringadatot. Ez a sztring (amelyet a következő fejezetben fogunk megvizsgálni) tartalmazza az oldal vezérlőelemeinek a *viewstate-jét*. Amint a kérés feldolgozódott, a szövegmezők eltűntek. Ez a HTTP állapotmentes természete miatt van. Ha a visszaküldések között módosítani szeretnénk ezeket a dinamikusan létrehozott szövegmezőket, el kell menteni az objektumokat az ASP.NET állapotkezelési módszereinek a segítségével (ezt ugyancsak a következő fejezetben ismertetjük).

**Forráskód** A DynamicCtrls kódfájlokat a forráskódkönyvtár 32. fejezetének alkönyvtára tartalmazza. A forráskódkönyvtárról lásd a Bevezetés xlv. oldalát.

# A System.Web.UI.WebControls.WebControl típus

A `Control` típus számos nem grafikus felülethez kapcsolódó viselkedéssel rendelkezik (vezérlőelem-gyűjtemény, automatikus visszaküldés támogatása stb.). A `WebControl` alaposztály viszont grafikus polimorf interfészt biztosít az összes webes vezérlőnek, ahogy azt a 32.2. táblázat mutatja.

| Tulajdonság | Jelentés |
|---|---|
| `BackColor` | Lekérdezi vagy beállítja a webes vezérlőelem háttérszínét. |
| `BorderColor` | Lekérdezi vagy beállítja a webes vezérlőelem szegélyszínét. |
| `BorderStyle` | Lekérdezi vagy beállítja a webes vezérlőelem szegélystílusát. |
| `BorderWidth` | Lekérdezi vagy beállítja a webes vezérlőelem szegélyének vonalvastagságát. |
| `Enabled` | Lekérdezi vagy beállítja, hogy a webes vezérlőelem engedélyezett-e. |
| `CssClass` | Lehetővé teszi, hogy egy CSS-stíluslapon definiált osztályt hozzárendeljünk egy webes vezérlőelemhez. |
| `Font` | Lekérdezi a webes vezérlőelem betűtípusadatait. |
| `ForeColor` | Lekérdezi vagy beállítja a webes vezérlőelem előtérszínét (általában a szöveg színét). |
| `Height`, `Width` | Lekérdezi vagy beállítja a webes vezérlőelem magasságát és szélességét. |
| `TabIndex` | Lekérdezi vagy beállítja a webes vezérlőelem tabulátorindexét. |
| `ToolTip` | Lekérdezi vagy beállítja a webes vezérlőelem eszköztippjét, amely akkor jelenik meg, amikor a kurzor a vezérlőelem felett van. |

**32.2. táblázat:** *A WebControl alaposztály néhány tulajdonsága*

A felsorolt tulajdonságok szinte mindegyike önmagáért beszél, így a következőkben néhány ASP.NET Web Form vezérlőelem működését vizsgáljuk meg.

# Az ASP.NET webes vezérlőelemeinek főbb kategóriái

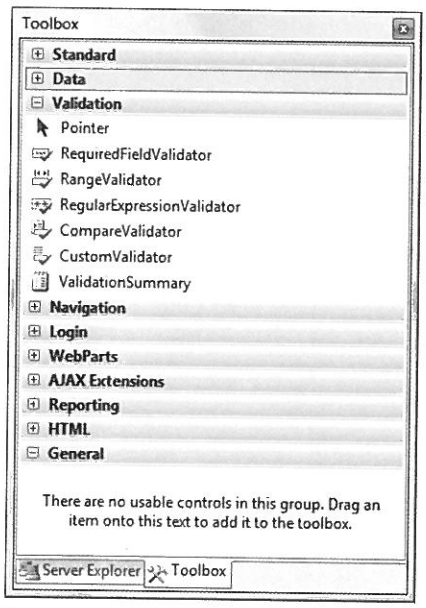
A `System.Web.UI.WebControls` típusokat több kategóriába sorolhatjuk:

- egyszerű vezérlőelemek,
- sokoldalú vezérlőelemek,
- adatközpontú vezérlőelemek,

- bemenet-ellenőrző vezérlőelemek (validátorok),
- webkijelzők,
- biztonsági vezérlőelemek.

Az *egyszerű vezérlőelemek* elnevezése arra utal, hogy olyan ASP.NET-es vezérlőelemekről van szó, amelyeket szabványos HTML-elemmé képezünk le (gombok, listák, hiperhivatkozások, képőrzők, táblázatok stb.). A következő vezérlőelem-gyűjteménybe a *sokoldalú vezérlőelemek* tartoznak, amelyeknek nincs közvetlen HTML-megfelelőjük (ilyen a `Calendar`, a `TreeView`, a `Menu`, a `Wizard` stb.). Az *adatközpontú vezérlőelemek* olyan céleszközök, amelyeket általában adatkapcsolat révén töltünk fel. Az ilyen típusú vezérlőelemek legjobb (és legkülönlegesebb) példája az ASP.NET GridView. A kategória többi tagjai az `Repeater` vezérlőelemek és a `DataList`.

Az *ellenőrző vezérlőelemek* olyan kiszolgálóoldali elemek, amelyek ügyfféloldali JavaScriptet bocsátanak ki automatikusan, űrlapmező-ellenőrzés céljából. Végezetül, az alaposztálykönyvtár számos biztonsági vezérlőelemet tartalmaz. Ezek a felhasználóifelület-elemek magukban foglalják a webhelyre való bejelentkezés részleteit, a jelszó-visszakeresési szolgáltatásokat és felhasználóiszerepkör-kezelést.



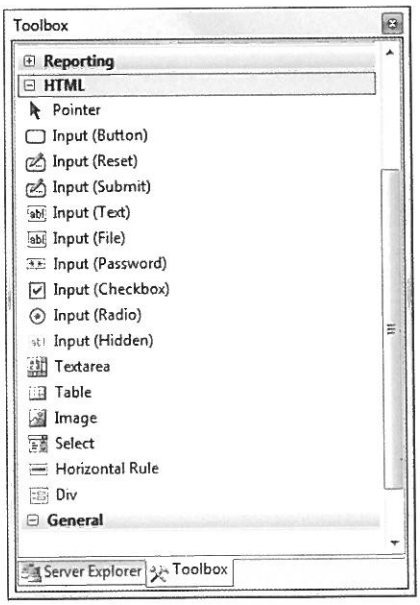

**32.2. ábra:** *Az ASP.NET webes vezérlőelemek*

Az ASP.NET teljes webes vezérlőelem-készletét a Visual Studio 2008 Toolbox használatakor nézhetjük meg. A 32.2. ábrán megfigyelhetjük, hogy a kapcsolódó vezérlőelemek speciális elnevezésű fülben vannak csoportosítva.

## Néhány szó a System.Web.UI.HtmlControls típusokról

Az ASP.NET két különböző webes vezérlőelem-eszközrendszert tartalmaz. Az ASP.NET webes vezérlőelemein kívül (amelyek a `System.Web.UI.WebControls` névtérben találhatók) az alaposztálykönyvtár a `System.Web.UI.HtmlControls` névtérben található vezérlőket is a rendelkezésünkre bocsátja.

A HTML-vezérlőelemek olyan típusok gyűjteménye, amelyek lehetővé teszik, hogy a hagyományos HTML-vezérlőelemeket webűrlapokon használjuk. A nyers HTML-tagektől eltérően azonban a HTML-vezérlőelemek olyan objektumorientált egyedek, amelyeket konfigurálhatunk úgy, hogy a kiszolgálón fussanak, így támogassák a kiszolgálóoldali eseménykezelést. Az ASP.NET webes vezérlőelemeivel ellentétben a HTML-vezérlőelemek meglehetősen egyszerűek, és a szabványos HTML-tageken túl csekély funkcionalitással szolgálnak (`HtmlButton`, `HtmlInputControl`, `HtmlTable` stb.). A Visual Studio 2008 egy speciális részt biztosít a Toolboxban a HTML-vezérlőelemek számára (lásd 32.3. ábrát).




**32.3. ábra:** *A HTML-vezérlőelemek*

A HTML-vezérlőelemek olyan nyilvános interfészt biztosítanak, amely a szabványos HTML-attribútumokat utánozza. Ahhoz például, hogy hozzáférjünk egy bemeneti tartomány adataihoz, használjuk a `Value` tulajdonságot a webes vezérlőelem-központú `Text` tulajdonság helyett. A HTML-vezérlőelemek nem olyan sokoldalúak, mint az ASP.NET webes vezérlőelemek. Ha szeretnénk többet megtudni ezekről a típusokról, lapozzuk fel a .NET Framework 3.5 SDK dokumentációját.

**Megjegyzés** A HTML-vezérlőelemek hasznosak lehetnek, ha tisztán szétválnak a HTML felhasználói felület tervezői és a .NET fejlesztői szerepek. A HTML-fejlesztők a kívánt webszerkesztőt használhatják, miközben ismert tageket alkalmazva továbbítják a HTML-fájlokat a fejlesztőcsapatnak. A fejlesztők ekkor kiszolgálóoldali vezérlőelemekké konfigurálhatják ezeket a HTML-vezérlőelemeket (ha a jobb gombbal a HTML-vezérlőre kattintanak a Visual Studio 2008-ban). Így lehetőségük nyílik a kiszolgálóoldali események kezelésére, és hogy programozottan dolgozzanak a HTML-vezérlőkkel.

# Sokoldalú ASP.NET-webhely készítése

Mivel sok „egyszerű" vezérlőelem megjelenése hasonlít a Windows Forms párjáéra, az alapvető vezérlőelemeket nem részletezzük (gombok, címkék, szövegdobozok stb.). Ehelyett készítsünk egy olyan webhelyet, amely számos különlegesebb vezérlőelem, valamint az ASP.NET-mesterodal és a fejlett adatkötés használatát szemlélteti. A következő példa kifejezetten az alábbi módszereket mutatja be:

- mesteroldalak használata,
- a `Menu` vezérlőelem használata,
- a GridView vezérlőelem használata,
- a `Wizard` vezérlőelem használata.

Először hozzunk létre egy új ASP.NET webalkalmazás-projektet, AspNetCarsSite néven.

## Mesteroldalak használata

Sok webhely következetes megjelenést és működést biztosít több oldalon keresztül (szokásos menüalapú navigációs rendszer, szokásos fejléc- és lábléctartalom, vállalati logó stb.). Az ASP.NET 1.x alatt a tervezők széles körben alkalmazták a `UserControlokat` és egyedi webes vezérlőelemeket a több oldalon keresztül felhasznált webes tartalom definiálásához. Jóllehet a `UserControlok` és az egyedi webes vezérlőelemek még mindig nagyon hatékonyak az ASP.NET alatt, most már rendelkezésünkre állnak a *mesteroldalak*, amelyek a meglévő módszereket egészítik ki.

A mesteroldal valamivel több, mint egy ASP.NET-lap, ugyanis `*.master` fájlkiterjesztést kapott. A mesteroldalak önmagukban nem láthatók a böngészőkben (azaz az ASP.NET-futtatókörnyezet nem szolgál ki mesteroldalakra mutató kéréseket). A mesteroldalak ehelyett közös felhasználóifelület-keretet definiálnak, amelyet a webhely összes lapja (vagy a lapok egy részhalmaza) használ.

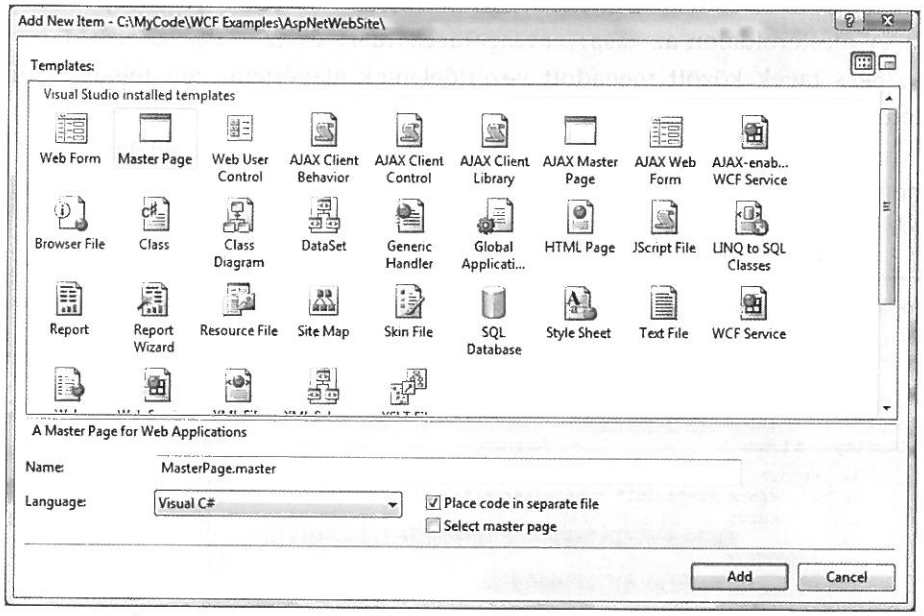
Emellett a `*.master` fájl különféle tartalomhelyőrző területeket definiál, amelyek olyan felhasználóifelület-tartományt alkotnak, amelyhez más `*.aspx` fájlok csatlakozhatnak. Azoknak az `*.aspx` fájloknak, amelyek egy mesteroldalhoz csatolják a tartalmukat, némileg eltérő a megjelenésük és a működésük, mint az eddig vizsgált `*.aspx` fájloknak. Az ilyen `*.aspx` fájlokat *tartalomlapnak* nevezzük. A tartalomlapok tehát olyan `*.aspx` fájlok, amelyek nem határoznak meg a HTML `<form>` elemét (ez a mesteroldal feladata).

Ám ami a végfelhasználót illeti, a kérés egy adott `*.aspx` fájlhoz fut be. A webkiszolgálón a kapcsolódó `*.master` fájl és az egyéb kapcsolódó `*.aspx` tartalomlapok egy lapban egyesülnek. A mesteroldalak és a tartalomlapok használatának szemléltetéséhez először adjunk a webhelyhez egy új mesteroldalt a Web Site > Add New Item menüpont segítségével (a 32.4. ábrán az eredményül kapott párbeszédablak látható).

A `MasterPage.master` fájl kezdeti markupja a következő:

```
<%@ Master Language="C#" AutoEventWireup="true"
  CodeFile="MasterPage.master.cs" Inherits="MasterPage" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
  <title>Untitled Page</title>
  <asp:ContentPlaceHolder id="head" runat="server">
```

```
    </asp:ContentPlaceHolder>
  </head>
  <body>
    <form id="form1" runat="server">
    <div>
      <asp:ContentPlaceHolder id="ContentPlaceHolder1" runat="server">
      </asp:ContentPlaceHolder>
    </div>
    </form>
  </body>
  </html>
```




**32.4. ábra:** *Új *.master fájl beszúrása*

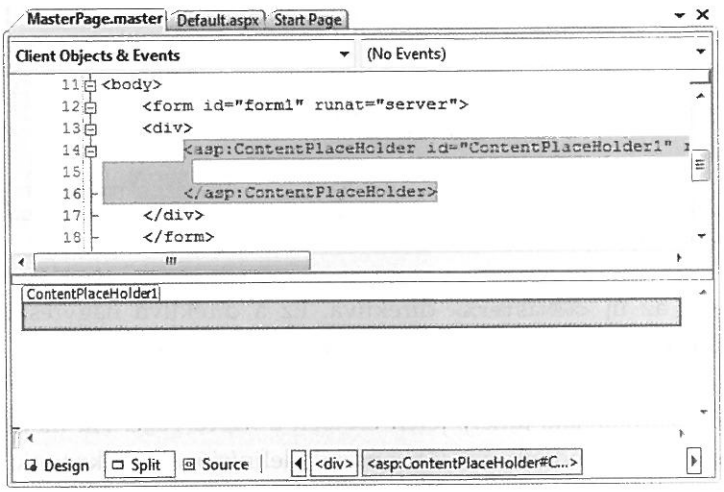
Az első érdekesség az új `<%@Master%>` direktíva. Ez a direktíva nagyrészt ugyanazokat az attribútumokat támogatja, mint az előző fejezetben bemutatott `<%@Page%>` direktíva. A `Page` típusokhoz hasonlóan a mesteroldalak egy adott ősosztályból származnak, amely jelen esetben a `MasterPage`. Ha megnyitjuk a kapcsolódó forráskódfájlt, az alábbi osztálydefinícióval találkozunk:

```
public partial class MasterPage : System.Web.UI.MasterPage
{
  protected void Page_Load(object sender, EventArgs e)
  {
  }
}
```

A mesteroldal markupjának másik érdekessége az `<asp:ContentPlaceHolder>` típus. A mesteroldalnak ez a tartománya azt a területet jelenti, amelyhez csatlakozhatnak a kapcsolódó `*.aspx` tartalomfájl felhasználóifelület-elemei. Ez a tartomány nem magán a mesteroldalon definiál tartalmat. Ha a `*.master` lap tervezői nézetére váltunk, láthatjuk minden egyes `<asp:ContentPlaceHolder>` elemhez tartozó megjelenést, ahogy a 32.5. ábra mutatja.

Ha egy `*.aspx` fájlban hivatkozunk erre a tartományra, akkor az `<asp:ContentPlaceHolder>` és az `</asp:ContentPlaceHolder>` tagek hatókörében a mesteroldalon található tartalom nem jelenik meg, hanem azt a `*.aspx` fájl felülírja. A mesteroldalon az `<asp:ContentPlaceHolder>` és az `</asp:ContentPlaceHolder>` tagek között megadott vezérlőelemek alapértelmezett felhasználói felületként működnek azokban az esetekben, amikor a webhely egy adott `*.aspx` fájlja nem hivatkozik erre a tartományra. Példaként tételezzük fel, hogy a webhely minden egyes `*.aspx` fájlja valóban hordoz egyedi tartalmat, ezért az `<asp:ContentPlaceHolder>` elemek üresek.

**Megjegyzés** A `*.master` lapok annyi tartalomhelyőrzőt definiálhatnak, amennyi csak szükséges. Emellett egy `*.master` lapba további `*.master` lapokat is beágyazhatunk.




**32.5. ábra:** *Egy *.master fájl <asp:ContentPlaceHolder> címkéi tervezés közben*

Ugyanazoknak a Visual Studio 2008 tervezőknek a segítségével hozhatunk létre közös felhasználói felületet a `*.master` fájlok számára, mint amelyekkel `*.aspx` fájlokat készítettünk. Ehhez a webhelyhez egy leíró címkét (amely egyszerű üdvözlőüzenetként fog szolgálni), egy `AdRotatort` (amely két kép

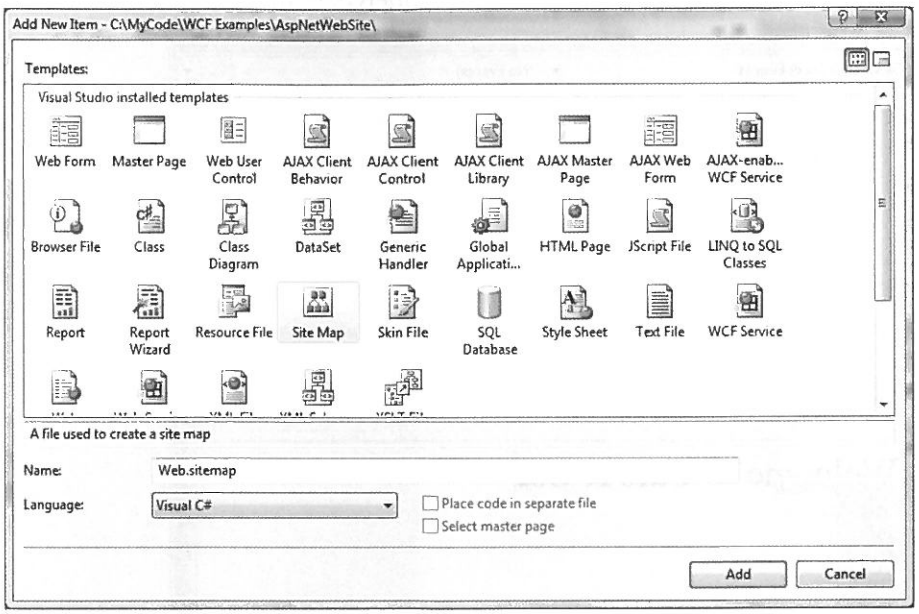
közül fog egyet véletlenszerűen megjeleníteni) és egy Menu vezérlőelemet (amelynek segítségével a webhely egyéb területeire navigálhat a felhasználó) fogunk hozzáadni. A 32.6. ábra a mesteroldal felhasználói felületének egyik lehetséges változatát mutatja be (a tartalomhelyőrző üres).


**32.6. ábra:** *A *.master fájl megosztott felhasználói felülete*

## A Menu vezérlőelem és a *.sitemap fájlok használata

Az ASP.NET számos olyan webes vezérlőelemet tartalmaz, amelyek a webhelyen belüli navigáció kezelésére szolgálnak: `SiteMapPath`, `TreeView` és `Menu`. Ezeket a webes vezérlőelemeket többféleképpen konfigurálhatjuk. Képesek például dinamikusan csomópontokat generálni egy külső XML-fájlból (vagy egy XML-alapú `*.sitemap` fájl alapján), programozottan forráskóddal vagy a markupban, a Visual Studio 2008 tervezői segítségével. A menürendszert dinamikusan fogjuk feltölteni egy `*.sitemap` fájl használatával. Ez azért előnyös, mert a webhely teljes struktúráját egy külső fájlban építhetjük fel, majd menet közben csatolhatjuk egy `Menu` (vagy `TreeView`) vezérlőhöz. Így, ha változik a webhely navigációs struktúrája, csak a `*.sitemap` fájlt kell módosíta-

nunk, majd az oldalt frissítenünk. Először szúrjunk be egy új `Web.sitemap` fájlt a projektbe a Web Site >Add New Item menüpont kiválasztása után megjelenő, a 32.7. ábrán látható párbeszédablakkal.




**32.7. ábra:** *Új Web.sitemap fájl beszúrása*

A kezdeti `Web.sitemap` fájl egy gyökérelemet és annak két alcsomópontját definiálja:

```
<?xml version="1.0" encoding="utf-8" ?>
<siteMap xmlns="http://schemas.microsoft.com/AspNet/SiteMap-File-1.0" >
  <siteMapNode url="" title="" description="">
    <siteMapNode url="" title="" description="" />
    <siteMapNode url="" title="" description="" />
  </siteMapNode>
</siteMap>
```

Ha ezt a szerkezetet összekapcsolnánk egy `Menu` vezérlőelemmel, egy fő menüelemet kapnánk két almenüvel. Ezért, ha alelemeket szeretnénk megadni, egyszerűen csak adjunk meg új `<siteMapNode>` elemeket egy meglévő `<siteMapNode>` hatókörén belül. Minden esetben az a célunk, hogy különféle `<siteMapNode>` elemek segítségével hozzuk létre a webhely teljes struktúráját egy `Web.sitemap` fájlban. Ezekkel az elemekkel cím- és URL-attribútumokat definiálhatunk.

Az URL-attribútum mutatja meg, hogy melyik `*.aspx` fájlra kell navigálni, amikor a felhasználó egy adott menüelemre (vagy a `TreeView` egy csomópontjára) kattint. A webhelyünk három oldalt tartalmaz, amelyek a következők:

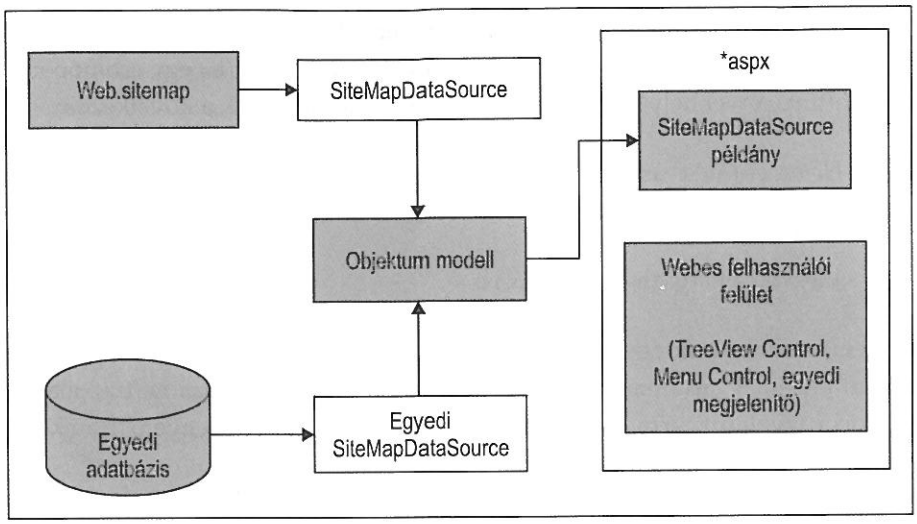
- *Home:* `Default.aspx`,
- *Build a Car:* `BuildCar.aspx`,
- *View Inventory:* `Inventory.aspx`.

A menürendszerünk egyetlen legfelsőbb szintű „Welcome" elemet és annak három alelemét tartalmazza. Ezért, módosítsuk a `Web.sitemap` fájlt az alábbi módon (ügyeljünk arra, hogy minden URL-érték egyedi legyen, ellenkező esetben futásidejű hiba lép fel):

```
<?xml version="1.0" encoding="utf-8" ?>
<siteMap xmlns="http://schemas.microsoft.com/AspNet/SiteMap-File-1.0" >
  <siteMapNode url="" title="Welcome!" description="">
    <siteMapNode url="~/Default.aspx" title="Home"
                 description="The Home Page" />
    <siteMapNode url="~/BuildCar.aspx" title="Build a car"
                 description="Create your dream car" />
    <siteMapNode url="~/Inventory.aspx" title="View Inventory"
                 description="See what is in stock" />
  </siteMapNode>
</siteMap>
```

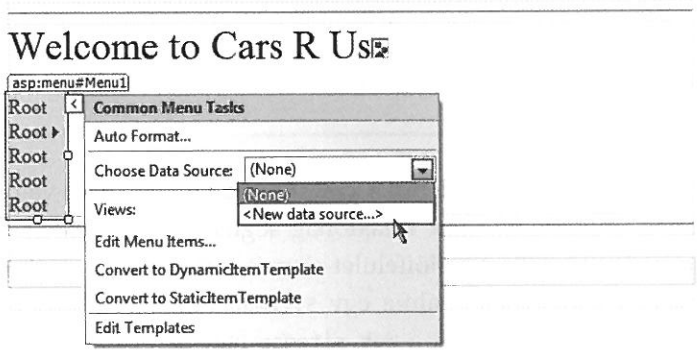
**Megjegyzés** Az oldalak előtt látható ~/ prefixum az URL-attribútumban a webhely gyökerét jelenti.

Ebben az esetben nem csatoljuk közvetlenül a `Web.sitemap` fájlt egy `Menu` vagy egy `TreeView` vezérlőelemhez egy adott tulajdonág segítségével. Ehelyett a `Web.sitemap` fájlt megjelenítő, felhasználóifelület-elemet tartalmazó `*.master` vagy `*.aspx` fájlnak kell magában foglalnia egy `SiteMapDataSource` komponenst. Ez a típus automatikusan betölti a `Web.sitemap` fájlt az objektummodelljébe, amikor lekérjük az oldalt. Ezután a `Menu` és a `TreeView` típus `DataSourceID` tulajdonsága a `SiteMapDataSource` példányára mutat. Az indirekció oka, hogy így olyan egyedi szolgáltatót hozhatunk létre, amely másik forrásból tölti be a webhely struktúráját (pl. egy adatbázistáblából, egy meglévő XML-fájlból stb.). A 32.8. ábra a `Web.sitemap`, a `SiteMapDataSource` és a különböző felhasználóifelület-elemek közötti kapcsolatot mutatja be.




**32.8. ábra:** *Az ASP.NET-webhelytérkép navigációs modellje*

Új `SiteMapDataSource` `*.master` fájlhoz adásához és a DataSourceID tulajdonság automatikus beállításához, használjuk a Visual Studio 2008 tervezőt. Aktiváljuk a `Menu` vezérlőelem `inline` szerkesztőjét, és válasszuk a New Data `Source` lehetőséget, ahogy azt a 32.9. ábra mutatja.




**32.9. ábra:** *Új SiteMapDataSource hozzáadása*

A megjelenő párbeszédablakban válasszuk a SiteMap ikont. Ezzel beállítjuk a Menu elem `DataSourceID` tulajdonságát, valamint hozzáadunk egy új `SiteMapDataSource` komponenst az oldalhoz. Csak ennyit kell tennünk, hogy a menü segítségével a webhely további oldalaira navigálhassunk.

Ha egyéb feldolgozást is szeretnénk végrehajtani, amikor a felhasználó egy adott menüelemet választ, kezeljük a `MenuItemClick` eseményt. Jelen példában erre nincsen szükség, de megadhatjuk a kiválasztott menüelemet a bemeneti `MenuEventArgs` paraméter segítségével.

## Kenyérmorzsa-vezérlőelem létrehozása a SiteMapPath típus segítségével

Mielőtt továbblépnénk, még adjunk hozzá egy `SiteMapPath` típust a `*.master` fájlhoz, a tartalomhelyőrző elem alá. Ez a vezérlő automatikusan beállítja a tartalmát a menürendszer aktuális kijelölésétől függően. Ez pedig hasznos vizuális figyelmeztetéssel szolgálhat a végfelhasználó számára (ezt a felhasználói felületi megoldást hivatalosan *kenyérmorzsának* nevezzük). A feladat befejezése után, amikor a Welcome > Build a Car menüelemet választjuk, a `SiteMapPath` ennek megfelelően automatikusan frissül.

## Az AdRotator használata

Az ASP.NET `AdRotator` vezérlőelemének az a szerepe, hogy véletlenszerűen kiválasztott képeket jelenítsen meg egy adott helyen a böngészőben. Amikor először helyezünk `AdRotatorta` tervezőbe, az üres helyőrzőként jelenik meg. Ez a vezérlőelem ugyanis addig nem képes működni, amíg az `AdvertisementFile` tulajdonság nem mutat a képeket leíró fájlra. Ebben a példában az adatforrás egy `Ads.xml` nevű, egyszerű XML-fájl.

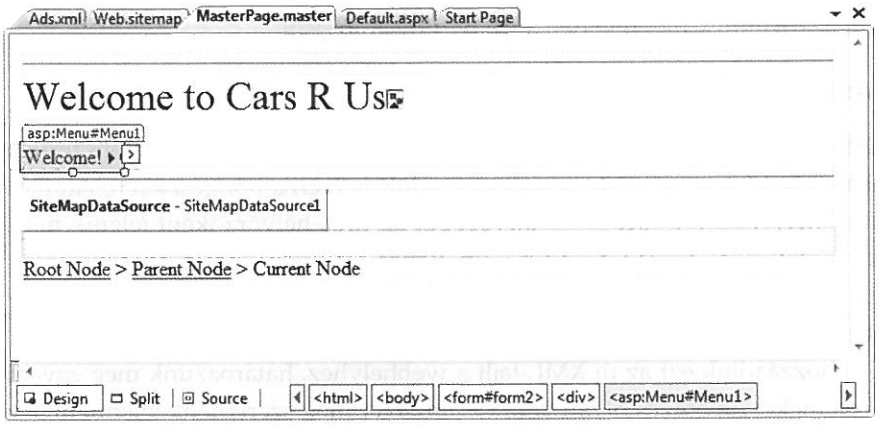
Ha hozzáadjuk ezt az új XML-fájlt a webhelyhez, határozzunk meg egyedi `<Ad>` elemeket az összes olyan képhez, amelyet meg szeretnénk jeleníteni. Az `<Ad>` elemek legalább a megjelenítendő képet (`ImageUrl`), a képre kattintást követően megjelenő URL-t (`TargetUrl`), az egérkurzor felett megjelenő szöveget (`AlternateText`) és az elem súlyát (`Impressions`) határozzák meg:

```
<Advertisements>
  <Ad>
    <ImageUrl>SlugBug.jpg</ImageUrl>
    <TargetUrl>http://www.Cars.com</TargetUrl>
    <AlternateText>Your new Car?</AlternateText>
    <Impressions>80</Impressions>
  </Ad>
  <Ad>
    <ImageUrl>car.gif</ImageUrl>
    <TargetUrl>http://www.CarSuperSite.com</TargetUrl>
    <AlternateText>Like this Car?</AlternateText>
    <Impressions>80</Impressions>
  </Ad>
</Advertisements>
```

Jelen esetben két képfájlt adtunk meg (`car.gif` és `slugbug.jpg`), ezért biztosítanunk kell, hogy ezek a fájlok a webhely gyökerében legyenek (a fájlok a könyvhöz tartozó letölthető forráskód mellett találhatóak.) Ahhoz, hogy a fájlokat az aktuális projekthez adjuk, válasszuk a Web Site > Add Existing Item menüpontot. Ekkor társíthatjuk az XML-fájlunkat az `AdRotator` vezérlőelemekkel az `AdvertisementFile` tulajdonság révén (a Properties ablakban):

```
<asp:AdRotator ID="myAdRotator" runat="server"
    AdvertisementFile="~/Ads.xml"/>
```

Később, amikor futás közben az alkalmazás visszaküld az oldalra, a két képfájl egyike véletlenszerűen meg fog jelenni. A 32.10. ábra a mesteroldalon látható felhasználói felület tervezésének utolsó lépéseit mutatja.




**32.10. ábra:** *A mesteroldal végső megjelenése*

## A Default.aspx tartalomlap definiálása

Miután létrehoztuk a mesteroldalt, elkezdhetjük az egyedi `*.aspx` fájlok tervezését, amelyek a felhasználói felület tartalmát a mesteroldal `<asp:ContentPlaceHolder>` tagjében egyesítik. Az új webhely létrehozásakor a Visual Studio 2008 automatikusan létrehozott egy kezdeti `*.aspx` fájlt, de az a mostani állapotában nem fésülhető össze a mesteroldallal.

Ennek az az oka, hogy a `*.master` fájl definiálja a végső HTML-oldal `<form>` részét. Ezért az `*.aspx` fájl meglévő `<form>` tartományát egy `<asp:content>` hatókörrel kell helyettesítenünk. Ahelyett, hogy manuálisan módosítanánk a kezdeti `*.aspx` fájl markupját, inkább töröljük a `Default.aspx` fájlt a projektből.

Ha szeretnénk automatikusan beszúrni egy új tartalomlapot, egyszerűen csak kattintsunk a jobb egérgombbal a `*.master` fájl tervezőfelületére, és válasszuk az Add Content Page menüpontot. Ezzel az alábbi markappal ellátott, új `*.aspx` fájlt generáltunk:

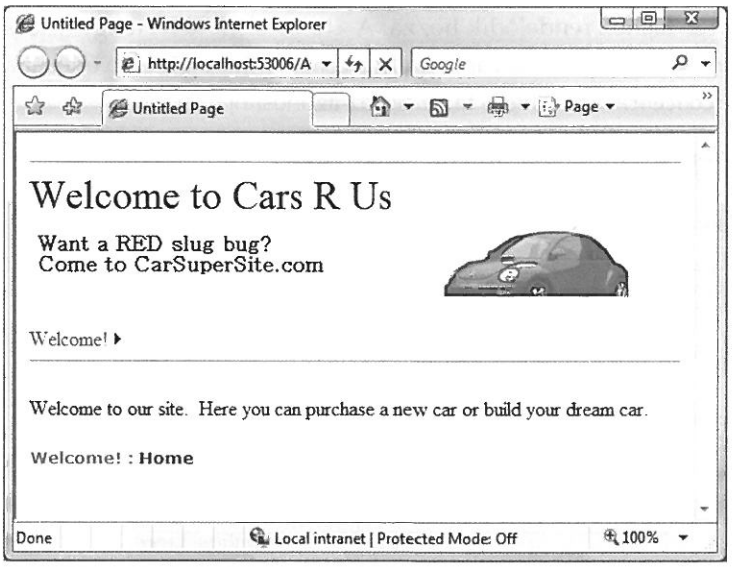
```
<%@ Page Language="C#" MasterPageFile="~/MasterPage.master"
        AutoEventWireup="true" CodeFile="Default.aspx.cs"
        Inherits="_Default" Title="Untitled Page" %>
<asp:Content ID="Content1"
     ContentPlaceHolderID="head" Runat="Server">
</asp:Content>
<asp:Content ID="Content2"
     ContentPlaceHolderID="ContentPlaceHolder1" Runat="Server">
</asp:Content>
```

Először is a `<%@Page%>` direktíva egy új MasterPageFile attribútummal egészült ki, amely a `*.master` fájlhoz rendelődik hozzá. A `<form>` elem helyett egy (jelenleg üres) `<asp:Content>` hatókörünk van, amely a `ContentPlaceHolderID` értékét a mesteroldal `<asp:ContentPlaceHolder>` tagjének az azonosítóját adja.


**32.11. ábra:** *Az első tartalomlap létrehozása*

Ezek miatt a társítások miatt tudja a tartalomlap, hova csatolja a tartalmát, amíg a mesteroldal tartalma csak olvasható formában jelenik meg a tartalomlapon. Nincs szükség arra, hogy bonyolult felhasználói felületet készítsünk a `Default.aspx` számára, így jelen példánkban csak valamilyen literális szöveget adjunk hozzá, amely némi alapadattal szolgál a webhelyről, a 32.11. ábrán látható módon (a tartalomlap jobb felső részén egy hivatkozás található a kapcsolódó mesteroldalra).

Ha lefuttatjuk az alkalmazást, látható, hogy a `*.master` és a `Default.aspx` fájlok felhasználóifelület-tartalmát egyetlen HTML-folyamba egyesítettük. Ahogy a 32.12. ábra mutatja, a végfelhasználó nem tudja, hogy a mesteroldal egyáltalán létezik. Ha frissítjük az oldalt (az F5-ös billentyű megnyomásával), az `AdRotator` véletlenszerűen megjeleníti a két kép egyikét.




**32.12. ábra:** *Futás közben a mesteroldal és a tartalomlapok együtt jelennek meg*

**Megjegyzés** Egy oldal mesteroldalát programozottan egy `Page` osztályból származtatott típus `PreInit` eseményén belül rendelhetjük hozzá az oldalhoz a Master tulajdonság segítségével.

## Az Inventory tartalomlap tervezése

Ahhoz, hogy az `Inventory.aspx` tartalomlapot beszúrjuk az aktuális projektbe, nyissuk meg az IDE `*.master` lapját, válasszuk a Web Site > Add Content Page lehetőséget (ha nem `*.master` fájl az aktív elem a tervezőben, ez a me-

nüpont nem jelenik meg), és nevezzük át a fájlt `Inventory.aspx`-re. Az Inventory tartalomlap szerepe az, hogy egy GridView vezérlőelemben megjelenítse az AutoLot adatbázis Inventory táblájának a tartalmát.

Habár ez a vezérlőelem sok tekintetben úgy viselkedik, mint elődje, az ASP.NET *1.x* DataGrid, a GridView belső támogatással rendelkezik az ASP.NET legújabb adatkötési modulját illetően. Az új modell alatt már lehetőség van kapcsolatsztring-adatokat, illetve `Select`, `Insert`, `Update` és `Delete` SQL-utasításokat (vagy esetleg tárolt eljárásokat) a *markup*ban megadni. Ezért ahelyett, hogy manuálisan programoznánk az összes szükséges ADO.NET-forráskódot, használjuk az új `SqlDataSource` típust. A vizuális tervezők alkalmazásával deklaratív módon hozhatjuk létre a szükséges markupot, majd rendelhetjük hozzá a GridView `DataSourceID` tulajdonságát a `SqlDataSource` komponenshez.

**Megjegyzés** Neve ellenére az `SqlDataSource` konfigurálható úgy, hogy képes legyen bármely ADO.NET-adatszolgáltatóval (ODBC, Oracle stb.) kommunikálni, amelyet a Microsoft .NET platform tartalmaz; nem korlátozódik a Microsoft SQL Serverre. A kívánt DBMS-t a `Provider` tulajdonság segítségével állíthatjuk be.

Néhány kattintással beállíthatjuk, hogy a GridView a mögöttes adattároló rekordjait automatikusan lekérdezze, módosítsa és törölje. Ez a kódolásmentes szemlélet jócskán lecsökkenti a sablonkódok mennyiségét, ez az egyszerűség azonban a testreszabhatóság csökkenésével is jár, és talán nem a legmegfelelőbb egy vállalati szintű alkalmazás számára. Ez a modell kis forgalmú oldalak, webhelyprototípusok vagy kisebb, házon belüli alkalmazások számára lehet jó megoldás.

A GridView (és az új adatkötési modul) deklaratív használatának szemléltetéséhez módosítsuk az `Inventory.aspx` tartalomlapot egy leíró címkével. Majd nyissuk meg a Server Explorert (a View menün keresztül), és bizonyosodjunk meg róla, hogy létrehoztuk az adatkapcsolatot az ADO.NET vizsgálata közben létrehozott AutoLot adatbázishoz (az adatkapcsolatok hozzáadását a 22. fejezetben tekinthetjük át). Fogjuk meg az Inventory ikont, és húzzuk az `Inventory.aspx` fájl tartalomlapjára. Ha ezzel megvagyunk, az IDE a következő lépéseket hajtja végre:

1. A `web.config` fájlt egy új `<connectionStrings>` elemmel egészíti ki.
2. Az `SqlDataSource` számára konfigurálja a szükséges `Select`, `Insert`, `Update` és `Delete` utasításokat.
3. A GridView `DataSourceID` tulajdonságának beállítja az új `SqlDataSource` azonosítóját.

**Megjegyzés** Másik megoldásként beállíthatjuk, hogy a GridView az inline szerkesztőt használja. A legördülő Choose Data Source listában válasszuk a New Data Source lehetőséget. Ezzel egy varázslót indítunk el, amely néhány lépésen keresztül összekapcsolja az összetevőt a kívánt adatforrással.

Ha megvizsgáljuk a GridView vezérlőelem nyitó tagjét, látható, hogy a `DataSourceID` tulajdonság a definíciónak megfelelően az `SqlDataSource` azonosítóját vette fel:

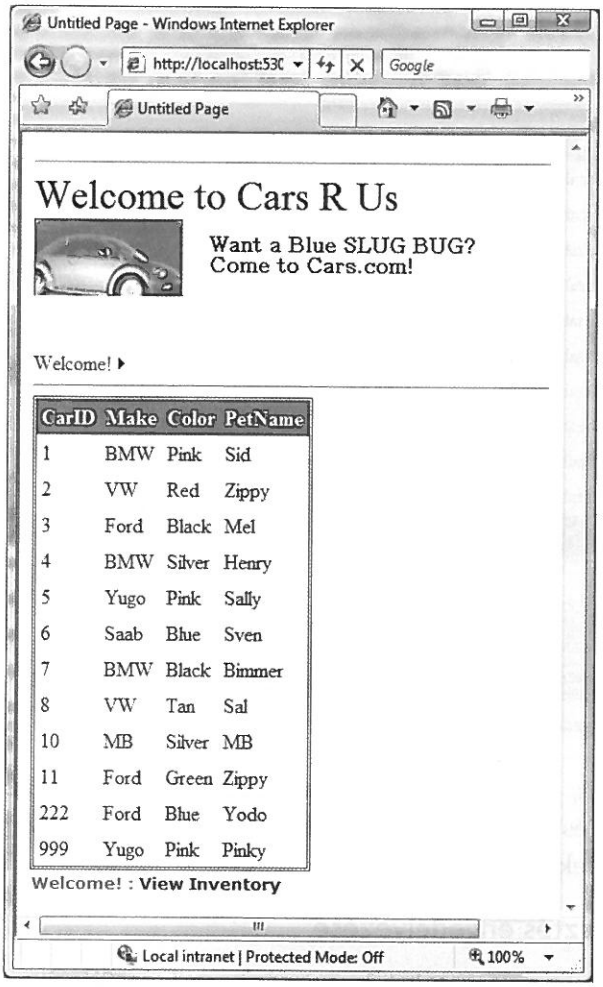
```
<asp:GridView ID="GridView1" runat="server"
    AutoGenerateColumns="False"
    CellPadding="4" DataKeyNames="CarID"
    DataSourceID="SqlDataSource1"
    ForeColor="#333333" GridLines="None">
    ...
</asp:GridView>
```

A lényeg az `SqlDataSource` típus leírásában található. A markupban a következő részben láthatjuk, hogy a típus felvette a szükséges SQL-utasításokat (egyszerű paraméterezett lekérdezésekként) az AutoLot adatbázis Inventory táblájával folytatott interakcióhoz. Emellett, az összetevő automatikusan kiolvassa a `web.config` `<connectionString>` értékét a `ConnectionString` tulajdonság `$` szintaxisa révén:

```
<asp:SqlDataSource ID="SqlDataSource1" runat="server"
    ConnectionString=
    "<%$ ConnectionStrings:CarsConnectionString1 %>"
  DeleteCommand="DELETE FROM [Inventory] WHERE [CarID] = @CarID"
  InsertCommand="INSERT INTO [Inventory] ([CarID], [Make], [Color],
                [PetName]) VALUES (@CarID, @Make, @Color, @PetName)"
  ProviderName=
     "<%$ ConnectionStrings:CarsConnectionString1.ProviderName %>"
  SelectCommand="SELECT [CarID], [Make], [Color], [PetName]
                FROM [Inventory]"
  UpdateCommand="UPDATE [Inventory] SET [Make] = @Make,
                [Color] = @Color, [PetName] = @PetName
                WHERE [CarID] = @CarID">
  <DeleteParameters>
    <asp:Parameter Name="CarID" Type="Int32" />
  </DeleteParameters>
  <UpdateParameters>
    <asp:Parameter Name="Make" Type="String" />
    <asp:Parameter Name="Color" Type="String" />
    <asp:Parameter Name="PetName" Type="String" />
    <asp:Parameter Name="CarID" Type="Int32" />
  </UpdateParameters>
```

```
    <InsertParameters>
      <asp:Parameter Name="CarID" Type="Int32" />
      <asp:Parameter Name="Make" Type="String" />
      <asp:Parameter Name="Color" Type="String" />
      <asp:Parameter Name="PetName" Type="String" />
    </InsertParameters>
  </asp:SqlDataSource>
```

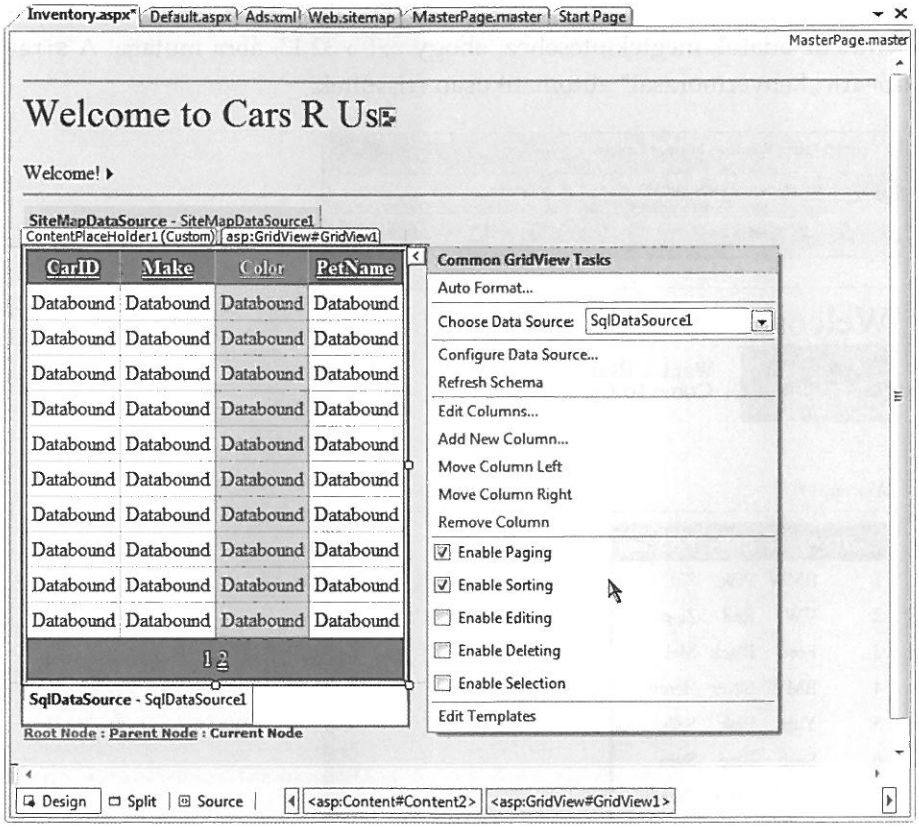
Futtassuk le az alkalmazást, közben kattintsunk a View Inventory menüelemre az adatok megtekintéséhez, ahogy azt a 32.13. ábra mutatja. A `SiteMapPath` „kenyérmorzsái" automatikusan frissültek.




**32.13. ábra:** *Az SqlDataSource „kódolásmentes" modellje*

## Rendezés és lapozás engedélyezése

A GridView vezérlőelem egyszerűen konfigurálható rendezésre (oszlopnév-hivatkozásokkal) és lapozásra (numerikus vagy következő/előző hivatkozásokkal). Ehhez használjuk az inline szerkesztőt, és ellenőrizzük a megfelelő beállításokat, ahogy azt a 32.14. ábra mutatja.



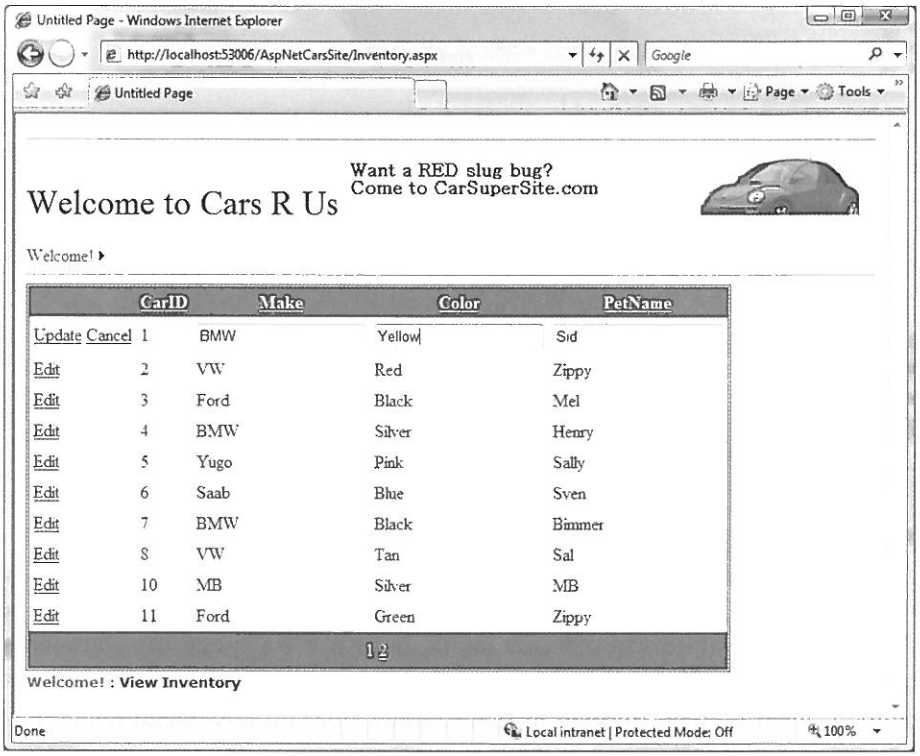

**32.14. ábra:** *Rendezés és lapozás engedélyezése*

Ha újra lefuttatjuk az oldalt, lehetőségünk nyílik az adatok rendezésére, az oszlopok nevére kattintva, és lapozására a lapozóhivatkozások segítségével (ha az Inventory tábla elég rekordot tartalmaz).

## Helyben történő szerkesztés engedélyezése

Az utolsó lépés a GridView vezérlőelem helyben történő szerkesztésének az engedélyezése. Mivel az `SqlDataSource` már tartalmazza a szükséges törlési és módosítási logikát, elegendő, ha a GridView Enable Deleting és az Enable

Editing jelölőnégyzeteit bejelöljük (a 32.14. ábra alapján). Amikor visszatérünk az `Inventory.aspx` lapra, lehetőségünk lesz rekordokat szerkeszteni és törölni, ahogy azt a 32.15. ábra mutatja, valamint módosítani az AutoLot adatbázis mögöttes Inventory tábláját.



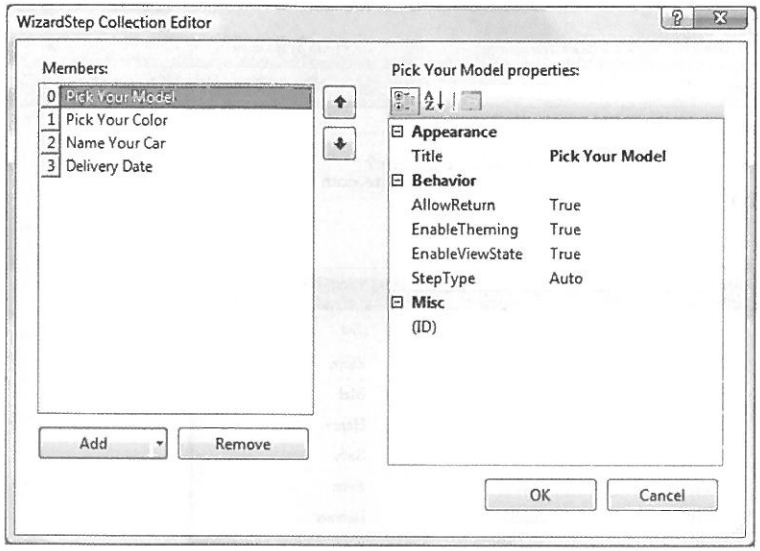

**32.15. ábra:** *Szerkesztés és törlés*

**Megjegyzés** A GridView helybeni szerkesztésének engedélyezéséhez egy elsődleges kulcsot kell az adatbázistáblához rendelni. Ha nem tudjuk engedélyezni a fenti opciókat, előfordulhat, hogy elfelejtjük a `CarID` tulajdonságot az AutoLot adatbázis Inventory táblájának elsődleges kulcsaként beállítani.

## A Build-a-Car tartalomlap tervezése

A példa utolsó lépése a `BuildCar.aspx` tartalomlap megtervezése. Szúrjuk be ezt a fájlt az aktuális projektbe (a Web Site > Add Content Page menüponton keresztül). Ez az új oldal az ASP.NET Wizard webes vezérlőeleme révén egyszerűen végigkíséri a felhasználót egy sor kapcsolódó lépésen. A szóban forgó lépések jelen esetben egy beszerzendő autó összeszerelési folyamatát szemléltetik.

Helyezzünk egy leíró címkét és egy Wizard vezérlőelemet a tartalomtartományba. Ezután aktiváljuk az inline szerkesztőt a Wizard vezérlőelemre, és kattintsunk az Add/Remove WizardSteps hivatkozásra. Végezzük el mind a négy lépést úgy, ahogy a 32.16. ábra mutatja.




**32.16. ábra:** *A varázsló konfigurálása*

A lépések meghatározása után azt látjuk, hogy a Wizard egy üres tartalomtartományt definiál, amelyre ráhúzhatjuk az aktuálisan kijelölt lépés vezérlőelemeit. Jelen esetben minden lépéshez adjuk hozzá a következő felhasználóifelület-elemet (ügyeljünk rá, hogy minden elemhez csökkenő azonosító értéket adjunk a Properties ablak használatával):

- *modell kiválasztása:* TextBox,
- *szín kiválasztása:* ListBox,
- *autó elnevezése:* TextBox,
- *szállítás dátuma:* Calendar.

A ListBox vezérlőelem a Wizard egyetlen olyan felhasználóifelület-eleme, amely további lépéseket igényel. Jelöljük ki ezt az elemet a tervezőben (bizonyosodjunk meg róla, hogy előtte kijelöltük a Pick Your Color hivatkozást), és töltsük fel színkészlettel a Properties ablak `Items` tulajdonsága révén. Ezután az alábbi markupot láthatjuk a Wizard definíció hatókörében:

```
<asp:ListBox ID="ListBoxColors" runat="server" Width="237px">
  <asp:ListItem>Purple</asp:ListItem>
  <asp:ListItem>Green</asp:ListItem>
  <asp:ListItem>Red</asp:ListItem>
  <asp:ListItem>Yellow</asp:ListItem>
  <asp:ListItem>Pea Soup Green</asp:ListItem>
  <asp:ListItem>Black</asp:ListItem>
  <asp:ListItem>Lime Green</asp:ListItem>
</asp:ListBox>
```

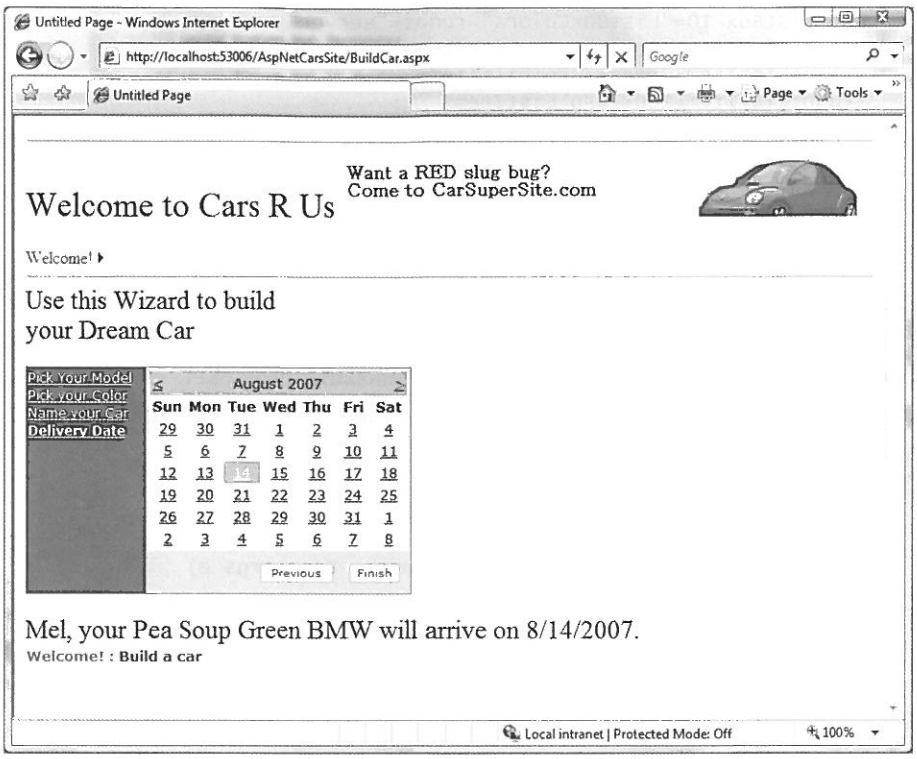
Az összes lépést definiáltuk, így kezeljük a `FinishButtonClick` eseményt az automatikusan generált Finish gombra. A kiszolgálóoldali eseménykezelőben kérdezzük le a vezérlőelemekben kiválasztott értékeket, és készítsünk egy leírósztringet, amelyet az `lblOrder` nevű újabb címke típus `Text` tulajdonságához rendelünk hozzá:

```
public partial class Default2 : System.Web.UI.Page
{
  protected void Page_Load(object sender, EventArgs e)
  {
  }

  protected void carWizard_FinishButtonClick(object sender,
            System.Web.UI.WebControls.WizardNavigationEventArgs e)
  {
    // Kérdezzük le az összes értéket.
    string order = string.Format("{0}, your {1} {2} will arrive
                  on {3}.", txtCarPetName.Text,
                  ListBoxColors.SelectedValue, txtCarModel.Text,
                  carCalendar.SelectedDate.ToShortDateString());

    // Adjuk hozzá a címkéhez.
    lblOrder.Text = order;
  }
}
```

Az AspNetCarsSite webalkalmazás elkészült. A 32.17. ábra a Wizard működését mutatja be.

Ezzel befejeztük az ASP.NET különféle, felhasználóifelület-központú webes vezérlőelemeinek vizsgálatát. A következőkben az ellenőrző vezérlőelemeket vizsgáljuk meg.

**Forráskód** Az AspNetCarsSite kódfájlokat a forráskódkönyvtár 32. fejezetének alkönyvtára tartalmazza. A forráskódkönyvtárról lásd a Bevezetés xlv. oldalát.




**32.17. ábra:** *A Wizard céleszköz működés közben*

# Az ellenőrző vezérlőelemek szerepe

A következő Web Form vezérlőelem-csoport gyűjtőneve: *ellenőrző vezérlőelemek*. Az eddig vizsgált Web Form-vezérlőelemektől eltérően, az ellenőrzőket nem a megjelenést szolgáló HTML-tagek beszúrására használjuk, hanem kliensoldali JavaScript (és esetleges kapcsolódó szerveroldali kód) űrlapellenőrzésre. Az kliensoldali űrlapellenőrzés meglehetősen hasznosnak bizonyul, hiszen ezzel különböző korlátokat helyezhetünk el még a webkiszolgálóhoz való visszaküldés előtt, költséges üzenetváltásokat kerülve el ezzel. A 32.3. táblázat az ASP.NET ellenőrző vezérlőelemeinek áttekintését tartalmazza.

| Vezérlőelem | Jelentés |
|---|---|
| `CompareValidator` | Ellenőrzi, hogy egy bemeneti vezérlőelem értéke egyenlő-e egy másik bemeneti vezérlőelem vagy egy konstans értékével. |

| Vezérlőelem | Jelentés |
|---|---|
| `CustomValidator` | Egyedi ellenőrző függvények készítését teszi lehetővé vezérlőelemek ellenőrzésére. |
| `RangeValidator` | Meghatározza, hogy egy adott érték része-e egy előre megadott tartománynak. |
| `RegularExpressionValidator` | Ellenőrzi, hogy a hozzárendelt bemeneti vezérlőelem értéke kielégíti-e a megadott reguláris kifejezést. |
| `RequiredFieldValidator` | Biztosítja, hogy egy adott bemeneti vezérlőelem tartalmazzon értéket (azaz ne legyen üres). |
| `ValidationSummary` | Egy lap ellenőrzési hibáit összesíti listázva, felsorolással vagy egyszerű bekezdésformátumban. A hibák inline módon és/vagy előugró üzenetdobozban jeleníthetők meg. |

**32.3. táblázat:** *Az ASP.NET ellenőrző vezérlőelemei*

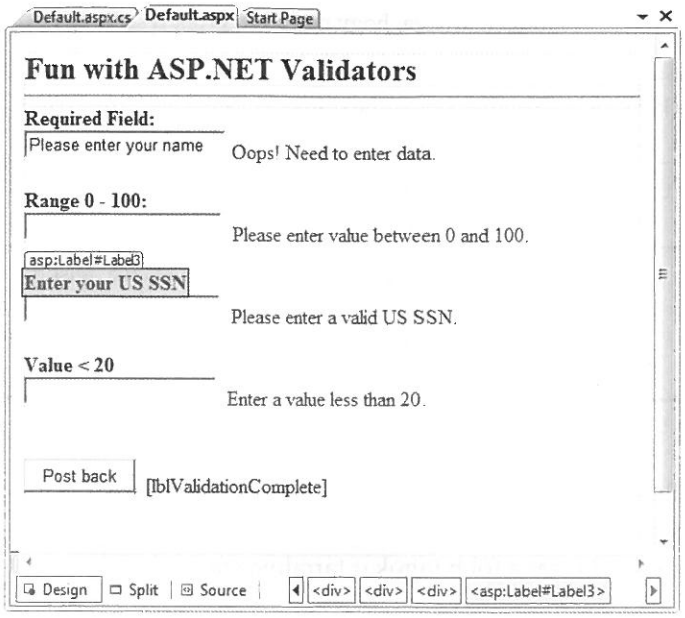
Végül is az összes ellenőrző vezérlőelem a `System.Web.UI.WebControls.BaseValidator` nevű, közös ősosztályból származik, így közös tulajdonságkészlettel rendelkeznek. A 32.4. táblázat a főbb tagokat tartalmazza.

| Tag | Jelentés |
|---|---|
| `ControlToValidate` | Lekérdezi vagy beállítja az ellenőrizendő bemeneti vezérlőelemet. |
| `Display` | Lekérdezi vagy beállítja az ellenőrző vezérlőelem hibaüzenetének megjelenítési viselkedését. |
| `EnableClientScript` | Lekérdezi vagy beállítja, hogy engedélyeztük-e az ügyféloldali ellenőrzést. |
| `ErrorMessage` | Lekérdezi vagy beállítja a hibaüzenet szövegének értékét. |
| `ForeColor` | Lekérdezi vagy beállítja a sikertelen ellenőrzéskor megjelenő üzenet színének értékét. |

**32.4. táblázat:** *Az ASP.NET ellenőrző vezérlőelemeinek közös tulajdonságai*

Az ellenőrző vezérlőelemek használatának bemutatásához, hozzunk létre egy új, ValidationCtrls nevű webhelyprojektet. Először helyezzünk el négy (megfelelően elnevezett) szövegdobozt (négy megegyező nevű címkével) a lapon.

Ezután helyezzünk el egy `RequiredFieldValidator`t, egy `RangeValidator`t, egy `RegularExpressionValidator` és egy `CompareValidator` a megfelelő mezők mellé. Végül adjunk hozzá egy gombot és egy címkét (lásd 32.18. ábra).




**32.18. ábra:** *Különböző ellenőrző vezérlőelemek*

Ha már van egy felhasználói felületünk, nézzük végig az egyes tagok konfigurálásának folyamatát.

## A RequiredFieldValidator vezérlőelem

A RequiredFieldValidator konfigurálása egyértelmű. Állítsuk be az `ErrorMessage` és a `ControlToValidate` tulajdonságokat a Visual Studio 2008 Properties ablakának használatával. Az eredmény, hogy a `txtRequiredField` szövegdoboz nem üres:

```
<asp:RequiredFieldValidator ID="RequiredFieldValidator1"
     runat="server" ControlToValidate="txtRequiredField"
     ErrorMessage="Oops! Need to enter data.">
</asp:RequiredFieldValidator>
```

A RequiredFieldValidator támogatja az `InitialValue` tulajdonságot. Ezzel a tulajdonsággal biztosíthatjuk, hogy a felhasználó által beírt értékek különbözzenek a kapcsolódó szövegdoboz kezdőértékétől. Konfigurálhatunk például egy szövegdobozt, amely a „Kérem adja meg a nevét" értéket tartalmazza arra az esetre, amikor a felhasználó először küld adatot az oldalra. Ha nem állítottuk be a RequiredFieldValidator `InitialValue` tulajdonságát, a futtatókörnyezet azt feltételezi, hogy a „Kérem adja meg a nevét" sztring érvényes. Így annak biztosítására, hogy a kért szövegdoboz csak akkor legyen érvényes, ha a felhasználó a „Kérem adja meg a nevét" szövegtől eltérő adatot visz be, a következőképpen állítsuk be a vezérlőelemet:

```
<asp:RequiredFieldValidator ID="RequiredFieldValidator1"
     runat="server" ControlToValidate="txtRequiredField"
     ErrorMessage="Oops! Need to enter data."
     InitialValue="Please enter your name">
</asp:RequiredFieldValidator>
```

## A RegularExpressionValidator vezérlőelem

A RegularExpressionValidatort akkor használjuk, ha mintailletszést szeretnénk alkalmazni az adott bemeneti mezőbe bevitt karakterekre. Ha azt szeretnénk, hogy az adott szövegdoboz csak érvényes amerikai társadalombiztosítási azonosítót tartalmazzon, az alábbiak szerint állíthatjuk be a vezérlőelemet:

```
<asp:RegularExpressionValidator ID="RegularExpressionValidator1"
     runat="server" ControlToValidate="txtRegExp"
     ErrorMessage="Please enter a valid US SSN."
     ValidationExpression="\d{3}-\d{2}-\d{4}">
</asp:RegularExpressionValidator>
```

Látható, hogyan definiálja a RegularExpressionValidator a `ValidationExpression` tulajdonságot. A reguláris kifejezéseket egy adott sztringmintával való összevetésre használjuk. Jelen esetben a `"\d{3}-\d{2}-\d{4}"` kifejezés egy xxx-xx-xxxx formátumú, szabványos amerikai társadalombiztosítási azonosítót takar (ahol az x bármilyen szám lehet).

Ez a reguláris kifejezés meglehetősen egyértelmű, ám tételezzük fel, hogy egy érvényes japán telefonszámot szeretnénk tesztelni. A pontos kifejezés jóval bonyolultabb lett: `"(0\d{1,4}-|\(0\d{1,4}\)?)?\d{1,4}-\d{4}"`. Amikor a Properties ablakban kijelöljük a ValidationExpression tulajdonságot, egy előre megadott reguláriskifejezés-készletből válogathatunk, a „..." feliratú gombra kattintva.

**Megjegyzés** A .NET platform két olyan névteret tartalmaz (`System.Text.RegularExpressions` és `System.Web.RegularExpressions`), amely az ilyen minták programozott kezelésével foglalkozik.

## A RangeValidator vezérlőelem

A `MinimumValue` és a `MaximumValue` tulajdonságokon kívül a RangeValidator vezérlőelem egy `Type` nevű tulajdonsággal is rendelkezik. Mivel a felhasználók által megadott bemeneteket szeretnénk ellenőrizni az egész számok halmazán, az Integer típust kell meghatároznunk (nem ez az alapértelmezett):

```
<asp:RangeValidator ID="RangeValidator1"
     runat="server" ControlToValidate="txtRange"
     ErrorMessage="Please enter value between 0 and 100."
     MaximumValue="100" MinimumValue="0" Type="Integer">
</asp:RangeValidator>
```

A RangeValidator vezérlőelemmel azt is tesztelhetjük, hogy egy adott érték valuta, dátum, lebegőpontos szám vagy sztringadat-e (alapértelmezett beállítás).

## A CompareValidator vezérlőelem

Végezetül figyeljük meg, hogy a CompareValidator egy `Operator` tulajdonságot támogat:

```
<asp:CompareValidator ID="CompareValidator1" runat="server"
     ControlToValidate="txtComparison"
     ErrorMessage="Enter a value less than 20." Operator="LessThan"
     ValueToCompare="20">
</asp:CompareValidator>
```

Minthogy ennek az ellenőrző vezérlőelemnek az a szerepe, hogy a szövegdobozban lévő értéket egy kétoperandusú operátor segítségével egy másikkal összehasonlítsa, nem meglepő, hogy az `Operator` tulajdonság értéke `LessThan`, `GreaterThan`, `Equal` és `NotEqual`. A `ValueToCompare` az összehasonlítás alapjául szolgáló érték megadására használatos.

**Megjegyzés** A CompareValidator vezérlőelemet úgy is konfigurálhatjuk, hogy egy másik Web Form vezérlőelem értékét hasonlítsa össze (inkább, mint egy bedrótozott értéket) a `ControlToValidate` tulajdonság révén.

Az oldal kódjának befejezéseként, kezeljük a gomb `Click` eseményét, és tudassuk a felhasználóval, hogy az ellenőrzési folyamat sikeresen lezárult:

```
public partial class _Default : System.Web.UI.Page
{
  protected void Page_Load(object sender, EventArgs e)
  {
  }

  protected void btnPostback_Click(object sender, EventArgs e)
  {
    lblValidationComplete.Text = "You passed validation!";
  }
}
```

Ezt követően navigáljunk erre az oldalra a kívánt böngészővel. Egyelőre semmi figyelemre méltó változást nem látunk. Ha azonban fiktív adatok bevitele után megpróbálunk rákattintani a Submit gombra, hirtelen megjelenik a hibaüzenet. Ha érvényes adatot viszünk be, a hibaüzenetek eltűnnek, és megtörténik a visszaküldés.

Ha megnézzük a böngésző által megjelenített HTML-t, azt látjuk, hogy az ellenőrző vezérlőelemek olyan ügyféloldali JavaScript függvényt generálnak, amely egy automatikusan letöltendő JavaScript függvénykönyvtárat használ (ami a `WebUIValidation.js` fájlban található). Ha az ellenőrzés sikeresen befejeződött, az űrlapadatok visszajutnak a kiszolgálóhoz, ahol az ASP.NET-futtatókörnyezet *ugyanazokat* az ellenőrzéseket fogja végrehajtani a webkiszolgálón (csak azért, hogy kizárja a hamisítás lehetőségét a felküldés során).

Ha olyan böngészőből érkezik a HTTP-kérés, amely nem támogatja az ügyféloldali JavaScript-kódokat, az összes ellenőrzés a kiszolgálón fog megtörténni. Ekkor úgy programozhatjuk az ellenőrző vezérlőelemeket, hogy figyelmen kívül hagyjuk a böngészőt, és a visszaküldött HTML-oldal a webkiszolgálóhoz irányítja vissza a hibafeldolgozást.

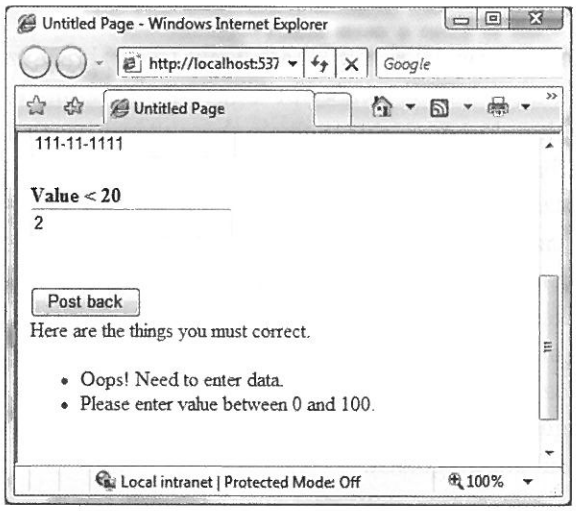
## Ellenőrzés összegzésének létrehozása

A következő ellenőrzéssel kapcsolatos témánk a ValidationSummary használata. Jelenleg az összes ellenőrző vezérlőelemünk pontosan ott jeleníti meg a hibaüzenetét, ahol azt a tervezéskor elhelyeztük. Sokszor ez éppen megfelelő. Valószínűleg azonban nem szeretnénk véletlenszerűen felugró, vörös szövegeket látni egy olyan összetett űrlapon, amelyen számos bemeneti elem található. A ValidationSummary típus segítségével utasíthatjuk az ellenőrzőket, hogy az oldal meghatározott pontjain jelenítsék meg a hibaüzeneteiket.

Először helyezzünk el egy ValidationSummary típust az `*.aspx` fájlba. Opcionálisan beállíthatjuk a típus `HeaderText` és `DisplayMode` tulajdonságait, ez utóbbi egy felsorolásban jeleníti meg az összes hibaüzenetet az alapértelmezés szerint.

```
<asp:ValidationSummary id="ValidationSummary1"
     runat="server" Width="353px"
     HeaderText="Here are the things you must correct.">
</asp:ValidationSummary>
```

Ezután az oldalon lévő összes ellenőrző vezérlőelem `Display` tulajdonságának (pl. `RequiredFieldValidator`, `RangeValidator`) adjunk `None` értéket. Ezzel biztosíthatjuk, hogy egy adott sikertelen ellenőrzés hibaüzenete ne jelenjen meg kétszer (egyik üzenet az összegző panelen és egy másik az ellenőrző vezérlőelem helyén). A 32.19. ábra az összegző ablaktáblát mutatja be működés közben.



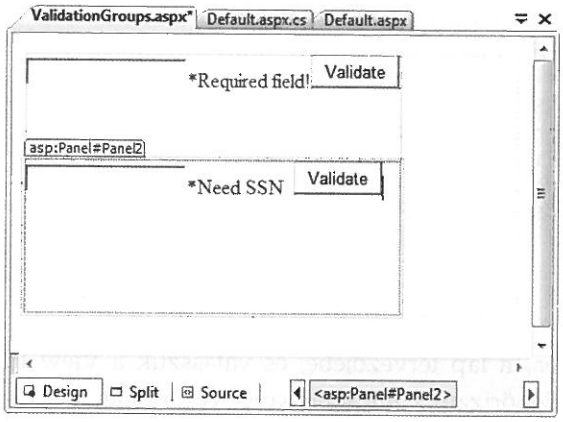

**32.19. ábra:** *Ellenőrzés összegzése*

Végül, de nem utolsósorban, ha azt szeretnénk, hogy a hibaüzenetek ügyféloldali MessageBox révén jelenjenek meg, a ShowMessageBox tulajdonság értékét állítsuk igazra, a ShowSummary tulajdonságét pedig hamisra.

## Ellenőrzési csoportok definiálása

Az ellenőrző vezérlőelemek számára *csoportokat* is definiálhatunk. Ez akkor lehet hasznos, amikor az oldal bizonyos tartományai együttműködésben állnak. Lehet például egy olyan vezérlőelem-csoportunk egy panelben, amely azt teszi lehetővé, hogy a felhasználó megadja a postacímét, egy másik panel pedig hitelkártya-információkat gyűjtő felhasználóifelület-elemeket tartalmaz. A csoportok használatával úgy konfigurálhatjuk az egyes vezérlőelem-csoportokat, hogy azok ellenőrzése egymástól független legyen.

Szúrjunk be egy új lapot `ValidationGroups.aspx` névvel az aktuális projektünkbe, amely két panelt definiál. Az első panel egy olyan szövegdobozt igényel, amely tetszőleges felhasználói adatot tartalmaz (egy RequiredFieldValidator vezérlőelemen keresztül), míg a második egy amerikai társadalombiztosítási azonosító értéket (RegularExpressionValidator vezérlőelemen keresztül) vár. A 32.20. ábra egy lehetséges felhasználói felületet ábrázol.




**32.20. ábra:** *Ezek a Panel objektumok egymástól függetlenül konfigurálják a bemeneti tartományaikat*

Annak biztosítására, hogy az ellenőrző vezérlőelemek egymástól függetlenül működjenek, az ellenőrző és az ellenőrzött vezérlőelemeket rendeljük hozzá egy egyedi elnevezésű csoporthoz a `ValidationGroup` tulajdonság segítségével. Az alábbiakban megadunk néhány lehetséges megoldást (az itt használt kattintási események kezelői lényegében a kódfájl üres rutinjai, és csak arra használjuk őket, hogy engedélyezzék a visszaküldést a webkiszolgálóra):

```
<form id="form1" runat="server">

  <asp:Panel ID="Panel1" runat="server" Height="83px" Width="296px">
    <asp:TextBox ID="txtRequiredData" runat="server"
         ValidationGroup="FirstGroup">
    </asp:TextBox>
    <asp:RequiredFieldValidator ID="RequiredFieldValidator1"
         runat="server" ErrorMessage="*Required field!"
         ControlToValidate="txtRequiredData"
         ValidationGroup="FirstGroup">
    </asp:RequiredFieldValidator>
    <asp:Button ID="bntValidateRequired" runat="server"
         OnClick="bntValidateRequired_Click"
         Text="Validate" ValidationGroup="FirstGroup" />
  </asp:Panel>

  <asp:Panel ID="Panel2" runat="server" Height="119px"
       Width="295px">
    <asp:TextBox ID="txtSSN" runat="server"
         ValidationGroup="SecondGroup">
    </asp:TextBox>
    <asp:RegularExpressionValidator ID="RegularExpressionValidator1"
         runat="server" ControlToValidate="txtSSN"
         ErrorMessage="*Need SSN"
         ValidationExpression="\d{3}-\d{2}-\d{4}"
         ValidationGroup="SecondGroup">
    </asp:RegularExpressionValidator> 
    <asp:Button ID="btnValidateSSN" runat="server"
         OnClick="btnValidateSSN_Click" Text="Validate"
         ValidationGroup="SecondGroup" />
  </asp:Panel>

</form>
```

Kattintsunk a jobb egérgombbal a lap tervezőjébe, és válasszuk a View In Browser menüpontot, hogy ellenőrizzük, mindkét panel vezérlőelemei kölcsönösen, egymást kizáróan működik.

**Forráskód** A ValidatorCtrls kódfájlokat a forráskódkönyvtár 32. fejezetének alkönyvtára tartalmazza. A forráskódkönyvtárról lásd a Bevezetés xlv. oldalát.

# Témák használata

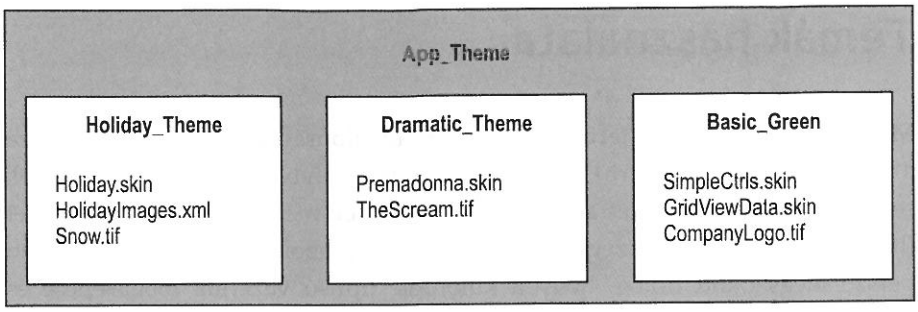
Minden vezérlőelemhez tartozik egy olyan tulajdonságkészlet (ezek nagy része a `System.Web.UI.WebControls.WebControl` alaposztályból származik), amelynek segítségével kialakíthatjuk a felhasználói felületek megjelenését és működését (háttérszín, betűméret, szegélystílus stb.). Egy többoldalas webhely esetében persze megszokott dolog, hogy a különféle típusú vezérlők megjelenése és működése megegyezik. Az összes szövegdoboz például egy adott betűtípust támogat, az összes gomb elemhez tartozik egy egyedi kép, és az összes naptár vezérlőelemnek világoskék a szegélye.

Nyilván nagyon sok munkát igényelne (és sok hibalehetőséget is rejtene magában), ha a webhely *összes* lapjának vezérlőire ugyanazokat a tulajdonságbeállításokat alkalmaznánk. Még ha képesek is lennénk manuálisan frissíteni az összes oldal összes felhasználóifelület-elemének tulajdonságát, nagyon fáradságos lenne az összes szövegdoboz háttérszínét szükségszerűen újra megváltoztatni. Kell lennie egy jobb megoldásnak a teljes webhelyre kiterjedő felhasználói felületi beállításokra.

Az egyik lehetőség a közös felhasználói felületi megjelenés és működés alkalmazásának leegyszerűsítésére *stíluslapok* definiálása. Ha van webes fejlesztői tapasztalatunk, tudjuk, hogy a stíluslapok böngészőkre alkalmazott, közös felhasználóifelület-központú beállításkészleteket definiálnak. Az ASP.NET webes vezérlőelemeihez a `CssStyle` tulajdonsággal rendelhetünk hozzá stílusokat.

Az ASP.NET azonban egy másik technológiát is biztosít számunkra a közös felhasználói felületek definiálásához, mégpedig a *témákat*. A stíluslapoktól eltérően a témákat a webkiszolgálóra alkalmazzuk (és nem a böngészőre), programozottan vagy deklaratív módon. Emiatt a témák a webhely összes kiszolgálóoldali erőforrásához hozzáférhetnek. Továbbá a témákat ugyanúgy definiáljuk, ahogyan azt bármelyik `*.aspx` fájlban megtalálhatjuk (a stíluslapok szintaxisa valamivel tömörebb).

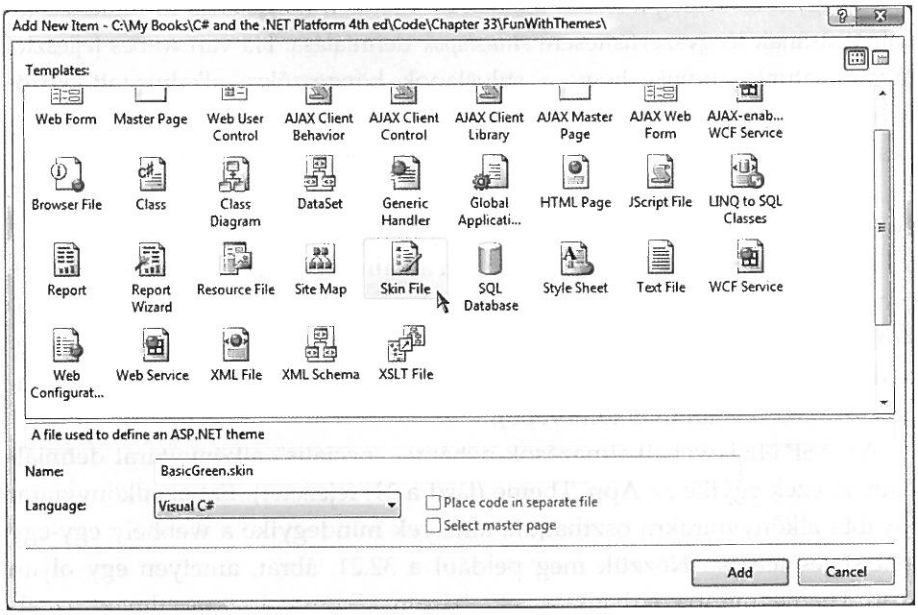
Az ASP.NET-webalkalmazások néhány „speciális" alkönyvtárat definiálhatnak, ezek egyike az App_Theme (lásd a 31. fejezetet). Ezt az alkönyvtárat további alkönyvtárakra oszthatjuk, amelyek mindegyike a webhely egy-egy lehetséges témája. Nézzük meg például a 32.21. ábrát, amelyen egy olyan App_Theme mappa látható, amely három alkönyvtárat tartalmaz, az alkönyvtárakban pedig az egy-egy témát felépítő fájlok találhatók.




**32.21. ábra:** *Egyetlen App_Theme mappa több témát is definiálhat*

## A *.skin fájlok

Az egyetlen fájl, amelyet minden témaalkönyvtár biztosan tartalmaz, a `*.skin` fájl. Ezek a fájlok határozzák meg különféle webes vezérlőelemek megjelenését és működését. Ennek szemléltetésére hozzunk létre egy új, FunWithThemes nevű webhelyet. Ezután szúrjunk be egy új, BasicGreen.skin nevű `*.skin` fájlt (a Web Site >Add New Item menüpont segítségével), a 32.22. ábrán látható módon.




**32.22. ábra:** *A *.skin fájlok beszúrása*

A Visual Studio 2008 figyelmeztet, hogy hagyjuk jóvá a fájl hozzáadását egy App_Theme mappához (ez pontosan az, mint amit szeretnénk). Ekkor az App_Theme mappa valóban tartalmaz egy BasicGreen nevű almappát, és benne az új BasicGreen.skin fájlt.

A `*.skin` fájlokban nyílik lehetőség a különféle vezérlőelemek megjelenésének és működésének meghatározására az ASP.NET-vezérlőelem deklarációs szintaxisa révén. Az IDE sajnos nem támogatja a `*.skin` fájlok tervezését. Úgy csökkenthetjük a gépelési időt, ha beszúrunk egy ideiglenes `*.aspx` fájlt a programunkba (pl. `temp.aspx`), amelyben elkészíthetjük a vezérlőelemek felhasználói felületét a Visual Studio laptervezője segítségével.

Az eredményül kapott markupot bemásolhatjuk a `*.skin` fájlba. Ekkor azonban el *kell* távolítanunk az összes webes vezérlőelem azonosítóattribútumát. Erre azért van szükség, mert nem egy adott, például gomb megjelenését és működését szeretnénk megadni, hanem az *összes* gombét.

A BasicGreen.skin alapértelmezett megjelenést és működést definiál a `Button`, a `TextBox` és a `Calendar` típusok számára:

```
<asp:Button runat="server" BackColor="#80FF80"/>
<asp:TextBox runat="server" BackColor="#80FF80"/>
<asp:Calendar runat="server" BackColor="#80FF80"/>
```

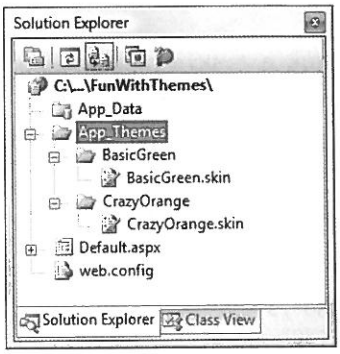
Minden vezérlőelem tartalmazza még a `runat="server"` attribútumot (ez kötelező), és egyikhez sem rendeltünk hozzá azonosítóattribútumot.

Definiáljunk egy másik, CrazyOrange nevű témát. A Solution Explorerben kattintsunk a jobb gombbal az App_Theme mappára, és adjunk hozzá egy új, CrazyOrange nevű témát. Ezzel új alkönyvtárat hozunk létre a webhely App_Theme mappájában. Ezután kattintsunk a jobb gombbal a CrazyOrange mappára a Solution Explorerben, és válasszuk az Add New Item lehetőséget. Adjunk hozzá egy új `*.skin` fájlt a megjelenő párbeszédablakban. Módosítsuk a CrazyOrange.skin fájlt úgy, hogy egyedi felhasználói felületi megjelenést és működést definiálunk a fent említett vezérlőelemekre. Például:

```
<asp:Button runat="server" BackColor="#FF8000"/>
<asp:TextBox runat="server" BackColor="#FF8000"/>
<asp:Calendar BackColor="White" BorderColor="Black"
    BorderStyle="Solid" CellSpacing="1"
    Font-Names="Verdana" Font-Size="9pt" ForeColor="Black"
    Height="250px" NextPrevFormat="ShortMonth"
    Width="330px" runat="server">
  <SelectedDayStyle BackColor="#333399" ForeColor="White" />
  <OtherMonthDayStyle ForeColor="#999999" />
  <TodayDayStyle BackColor="#999999" ForeColor="White" />
```

```
    <DayStyle BackColor="#CCCCCC" />
    <NextPrevStyle Font-Bold="True" Font-Size="8pt"
                   ForeColor="White" />
    <DayHeaderStyle Font-Bold="True" Font-Size="8pt"
                    ForeColor="#333333" Height="8pt" />
    <TitleStyle BackColor="#333399" BorderStyle="Solid"
                Font-Bold="True" Font-Size="12pt"
                ForeColor="White" Height="12pt" />
</asp:Calendar>
```

A Solution Explorer ekkor a 32.23. ábrán látható képet mutatja.




**32.23. ábra:** *Több téma egyetlen webhelyen*

A következő logikus kérdés az, hogy hogyan alkalmazzuk a témákat a lapokra? Ennek többféle módja van.

**Megjegyzés** Ezek a témapéldák meglehetősen egyszerűek, ám nyugodtan továbbalakíthatók.

## Témák alkalmazása a teljes webhelyre

Ha szeretnénk meggyőződni arról, hogy a webhely összes lapjára ugyanazt a témát alkalmaztuk, a legegyszerűbb, ha módosítjuk a `web.config` fájlt. Nyissuk meg az aktuális `web.config` fájlt, és keressük meg a `<pages>` elemet a `<system.web>` elem hatókörében. Ha hozzáadunk egy témaattribútumot a `<pages>` elemhez, a webhely összes lapjához a kijelölt témát rendeljük hozzá (amely természetesen az App_Theme egyik alkönyvtárának a neve). A módosítás a következő: