

Windows Formsszal ellentében – a WPF kezdeti kiadása nem biztosít lejtőt-gomb-vezetőlemelet, amely lehetővé teszi a felhasználó számára, hogy a fel- és lefelé nyílik segítséggel állítsan be numerikus értéket. A Repeatbutton funkcióval összhangban viszonylag könnyedén elkezthetünk egy lejtőt-vezetőt-onalitásnak köszönhetően amelyet a Windows.Miwindow : system.Windows.Window típusú osztályának privatate int currvalue = 0;

```

    {
        public partial class Mainwindow : system.Windows.Window
    }

    class StackPanel : system.Windows.UIElement
    {
        private int currvalue = 0;
        private void OnValueChange(object sender, system.EventArgs e)
        {
            if (currvalue < 0)
                currval
            else
                currval
        }
    }

    class Label : system.Windows.UIElement
    {
        private string value;
        private void OnValueChange(object sender, system.EventArgs e)
        {
            if (value < 0)
                currval
            else
                currval
        }
    }

    class Button : system.Windows.UIElement
    {
        private void OnClick(object sender, system.EventArgs e)
        {
            if (currval < 0)
                currval
            else
                currval
        }
    }

    class Window : system.Windows.Window
    {
        private void OnValueChange(object sender, system.EventArgs e)
        {
            if (currval < 0)
                currval
            else
                currval
        }
    }
}

```

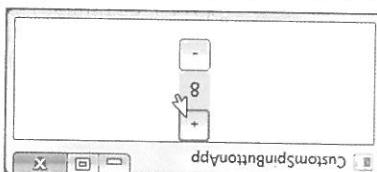
Ez a kód a Windows Formsszal ellentében a WPF kezdeti kiadása nem biztosít lejtőt-gomb-vezetőt, amely lehetővé teszi a felhasználó számára, hogy a fel- és lefelé nyílik segítséggel állítsan be numerikus értéket. A Repeatbutton funkcióval összhangban viszonylag könnyedén elkezthetünk egy lejtőt-vezetőt-onalitásnak köszönhetően amelyet a Windows.Miwindow : system.Windows.Window típusú osztályának privatate int currvalue = 0;

Ahoogy korábban említettük, a Checkbox "az-egy", ToggleButton, "az-egy" ButtontnBase, ami elég kilönösnek tűnhet, mivel egy gomb felhasználói felülete megfelelhetősen különbözik egy jelekönnyezetétől. Azonban, ha a checkbox típus

A Jelölőnagyzetek és a rádiógombok használata

Források A CustomSpinButtonApp kodfájljukat a forrásokonnyvatar 29. fejezetbenek alkotnyi-térre tárta meg. A forrásokonnyvatarról lásd a bevezetés kiv. oldalat.

29.6. adra: Leptetogomb kezeltise, kezdőponktként a RepeatButton használataval



Ahozgáyan latthatjuk, amikor a felhasználó rakkattint valamelyik RepeatButton az egységi lepettőgombunkat mutatókodes közben.

```

public Mainwindow()
{
    InitializeComponent();
    lblCurrentComponent = curvalue;
}

private void RepeatAddValueButton_Click(object sender, EventArgs e)
{
    if (hözéadedgyetazaktuálisértekhez, és megmutatja a címkeben.)
    {
        hözéadedgyetazaktuálisértekhez = true;
        lblCurrentComponent = curvalue;
        curvalue++;
    }
}

private void RepeatMoveValueButton_Click(object sender, EventArgs e)
{
    if (kivonegyettazaktuálisértekben.)
    {
        kivonegyettazaktuálisértekben = false;
        lblCurrentComponent = curvalue;
        curvvalue--;
    }
}
}

```

29. Fejezet: Programozás WPF-vezetőfelületekkel

Ha tesszémenk ezt a XAML-t, azt tapasztalnunk, hogy csak egyet választhatunk a hat lehetőség közül, ami valószínűleg nem áll számban, mivel ket különböző csoport van (rádió és szín lehetőségek).

```
<!-- RadioButton típusok zeneváltaszashoz -->
<StackPanel>
    <Label FontSize="15" Content="Select your Color Choice"/>
    <RadioButton Name="Red"/>
    <RadioButton Name="Green"/>
    <RadioButton Name="Blue"/>
    <RadioButton Name="Yellow"/>
    <Label FontSize="15" Content="Select your Music Media"/>
    <RadioButton Name="CD Player"/>
    <RadioButton Name="MP3 Player"/>
    <RadioButton Name="8 Track"/>
    <Label FontSize="15" Content="Select your Favorite Artist-->
<StackPanel>
```

A RadioButton típusok zeneváltaszashoz. A checkbox típusú ellentétben azonban rendelkezik azzal a termesztes környezetben, amely biztosítja, hogy az ügyanabban a trálosban (StackPanel), minden mászóknak a rádiók mellett a többi funkcióval szinkronizálva legyenek elérhetők.

29.7. ábra: Egyszerű Checkbox típusok

<input checked="" type="checkbox"/>	Contact me over the phone
<input type="checkbox"/>	Send me more information

```
<StackPanel>
    <CheckBox Name="checkboxNameContact" Content="Contact me over the phone"/>
    <CheckBox Name="checkboxInfo" Content="Send me more information"/>
    <!-- CheckBox típusok -->
<StackPanel>
```

Látható kimenetet eredményezik:

Tekintünk atta környezetet (checkbox) deklarációkat, amelyek a 29.7. ábrán hogy elválasszuk egy vezérlélem megjelenését annak funkciójával. Néset és mutatódeseit (jussónak eszünkbe, hogy a WPF fü mozgatásával az button különböző virtuális tagjait, hogy letelepítsse a jelenlegi pozíciót megjelenítően kidérül, hogy a checkbox típus egyszerűen felülírja a ToggleSwitchet, valamint követi a WPF-tartalommodellt. A hasonlatosagoknak köszönhetően mindenekkel, reagál az egér - es billentyűzetről, valamint környezet-purra (mint a button típusra) rakkattintáshatunk, reagál az egér - es billentyűzetről, valamint környezetek és a radiókombók használata

szégebe az előző Radiobutton típusokat:

vetkező két GroupBox deklaráció, amelyek közönökéle módosítani fogják egyetemes felületeket, gombot stb.), hogy fejleskünk működésünket. Vizsgálunk meg a kódolásban, hogy a Headert tulajdonos objektumot (egyszerű stringet, szövegötök, hozzárendelhetünk barátság alapvetően System.Object típusral használata. Mivel a Headert tulajdonos módosítja ehhez a GroupBox vezetőjélem portként viselkednek. Az általános módszer így mindenekkel szemben használható. Amikor radiogombok vagy jelölőnégyzetek gyűjteményét tervezzük, megszokott dolgozni vizualis részletekkel körülvenni így a GroupBox vezetőjélem portként viselkedőkkel. Az általános módszer így mindenekkel szemben használható. Amikor radiogombok vagy jelölőnégyzetek gyűjteményét tervezzük, meg-

A kapcsolódó elemek összeállítása GroupBox

Megjegyzés: Alapértelmezés szerint a GroupName értékkel nem rendelkező tarolóban lévő összes Radiobutton egyetlen fizikai csoporthoz kötődik. Ezért, az előző példában a színezés pontjukat a GroupName tulajdonoson belül elhagyott GroupName értékkel is, a Zene csoport jelenlétének köszönhetően.

Ezáltal egy másik függeléknél beállíthatjuk az egyes Logikai csoporthoz tartozókat, még akkor is, ha azok ugyanabban a fizikai tarolóban vannak.

```
<!-- A Zené csoport (tetszőlegesen valasztható ebben a példában,-->
<StackPanel>
    <!-- A Zené csoport -->
    <Label FontSize="15" Content="Select your Music Media"/>
    <RadioButton GroupName="Music" >CD Player</RadioButton>
    <RadioButton GroupName="Music" >MP3 Player</RadioButton>
    <RadioButton GroupName="Music" >8-Track</RadioButton>
    <Label FontSize="15" Content="Select your Color Choice"/>
    <RadioButton GroupName="Color" >Red</RadioButton>
    <RadioButton GroupName="Color" >Green</RadioButton>
    <RadioButton GroupName="Color" >Blue</RadioButton>
</StackPanel>
```

Ha egyetlen tarolt szeremeket több Radiobutton típusával, amelyek közönökéle felhasználhatók csoporthoz, ez a GroupName tulajdonoság beállítása val teljessílik a Radiobutton típus nyitó elemeit:

Logikai csoporthozok letrehozása

```

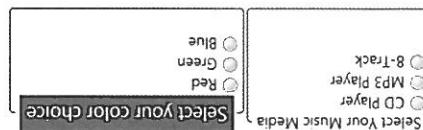
<RadioButtton GroupName = "Music" >MP3 Player</RadioButtton>
<RadioButtton GroupName = "Music" >CD Player</RadioButtton>
<StackPanel>
<Expander Header = "Select your Music Media" BorderBrush = "Black">
<StackPanel>

```

alapvetően <groupbox> tipusú <expander> tipusú modosul):
 (tel, le, bárha vagy jobbra). Tekintésük meg az alábbi XML-leírás (amely tulajdonság segítségevel meghatározunk az elemek megjelenítésének irányát tök. Ez az elem, az Expander típus levele tesz, hogy az Expandable-rection menyét, amelyek egy kapcsoló segítségevel elérhetők, illetve megmutathatók. Ez azzal kapcsolatos, hogy az Expander típus levele minden elemmel, amely csoporthoz olyan felhasználói felület-élémek gyűjtése állhat. A szokásos Groupbox típus mellé a WPF rendelkezik olyan új felhasználói fe-

Kapcsolódó elemek bövítő tipusokba csortoztatása

29.8. ábra: RadioButtton típusokat körtező GroupBox típusok



A képenet a 29.8. ábrán látható.

```

<StackPanel>
<Expander Header = "Select your color choice">
<GroupBox BorderBrush = "Black">
<GroupBox Header = "GroupBox Header">
<Label Background = "Blue" Foreground = "White">
  FontSize = "15" Content = "Select your color choice"/>
</Label>
<RadioButtton Red/>
<RadioButtton Green/>
<RadioButtton Blue/>
<RadioButtton Yellow/>
<RadioButtton 8-Track/>
</GroupBox>
</StackPanel>
<StackPanel>
<RadioButtton GroupName = "Music" >MP3 Player</RadioButtton>
<RadioButtton GroupName = "Music" >CD Player</RadioButtton>
<RadioButtton GroupName = "Music" >8-Track</RadioButtton>
</StackPanel>
<StackPanel>
<Expander Header = "Select your Music Media" BorderBrush = "Black">
<StackPanel>

```

Ahogyan azt remélhetünk, a WPF biztosít olyan típusokat, amelyek valászt-mindiggyike az Itemscorntrol absztrakt osztályból származik. A Legfontosabb elémek csoportját taralmazzák –ilyen a Listbox és a ComboBox, ezek használata elérhető.

A Listbox és a Combobox típusok használata

Források A ChecRadioGroup-xamit foglalja a forrásokkal kapcsolatos 29. fejezetek alkonyvátra tartalmazza. A forrásokkal kapcsolatos rész a Bevezetés xli. oldalát.

29. 10. ábra: Kitérjesszettet Expanderek



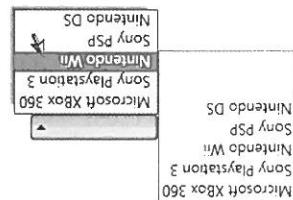
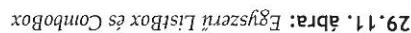
A 29.10. ábra mutatja az egyes expandereket kinyitott állapotban.

29.9. *abra*: Osszeeskott Expanderek



A 29.9. ábra mutatja az egyes Expandereket összeesküvőt állapotban.

29. fejezet: Programozás WPF-vezérlőelemekekkel



Nem megelőző módon a 29. Jl. ábrán látható rendszerrel szemben.

Megjegyzés A ComboBox típusokat felsorolhatunk („ListboxItem”). Elérhetők a többesítésű („Combo-Boxkészítmény”) és a többesítés nélküli („ListboxItem”) típusok.

sababb, hogy ez a szülőösszatyú detinája az items névű tulajdonságot, amely a részelmeket tárlo, erősít a típusos itemcöllectiion objektumot adja vissza. Az itemcöllectiion típus a system-object típusokkal dolgozik, ezért gyakorlatilag bármilyen táratlanmazsát. Ha maradék résven szeretnék egyszerű szöveges adatokkal feltölteni egy itemscontroll-leszámazott típuszt, akkor ezt lista-típusztm» típusok készleteinek használatával tehetjük meg. Példaként nezzük a következő XAML-kódot:

A Listbox és a Combobox típusok használata

Az egyik dolgozó, amely különösenek tűnhet a Listbox XML-leírásában, hogy a «listboxitem» típusokat használhatunk az add() metódus meghívása során. Ennek roszinigtipusokat használhatunk az add() metódus meghívása során. Ennek ro-

```

public partial class Mainwindow : System.Windows.Window
{
    public Mainwindow()
    {
        InitializeComponent();
        MinimizeWindow();
        FillListbox();
        InitializeComponents();
    }

    private void FillListbox()
    {
        // Elmek hozzáadása a Listamézőhöz.
        ListView.ListViewItemCollection items = listView1.Items;
        items.Add("Microsoft Xbox 360");
        items.Add("Sony PlayStation 3");
        items.Add("Nintendo Wii");
        items.Add("Sony PSP");
        items.Add("Nintendo DS");
        items.Add("Microsoft Xbox");
        items.Add("Sony PlayStation");
        items.Add("Nintendo Wii");
        items.Add("Sony PSP");
        items.Add("Nintendo DS");
    }

    private void InitializeComponents()
    {
        // Létrehozás a Listamézőt.
        listView1 = new ListView();
        listView1.Location = new Point(100, 100);
        listView1.Size = new Size(300, 200);
        listView1.View = View.List;
        listView1.Items.Add("Microsoft Xbox 360");
        listView1.Items.Add("Sony PlayStation 3");
        listView1.Items.Add("Nintendo Wii");
        listView1.Items.Add("Sony PSP");
        listView1.Items.Add("Nintendo DS");
        listView1.Items.Add("Microsoft Xbox");
        listView1.Items.Add("Sony PlayStation");
        listView1.Items.Add("Nintendo Wii");
        listView1.Items.Add("Sony PSP");
        listView1.Items.Add("Nintendo DS");
    }

    private void MinimizeWindow()
    {
        // A fő ablak kicsitítése.
        this.WindowState = WindowState.Minimized;
    }
}

```

És a kapcsolódó kódjában az alábbiak szerint tolthetők fel:

```
<Window x:Class="ListControls.MainWindow" xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml" xmlns:mc="http://schemas.microsoft.com/marketing/clickOnce/resources/microsoft.com/windows呈現"
        mc:Ignorable="targetNSVersion" TargetNSVersion="1.0">
    <Grid>
        <Grid.ColumnDefinitions>
            <ColumnDefinition Width="100px" />
            <ColumnDefinition Width="100px" />
        </Grid.ColumnDefinitions>
        <StackPanel Orientation="Vertical" Grid.Column="0">
            <TextBlock Text="StackPanel 1" />
            <TextBlock Text="StackPanel 1" />
            <TextBlock Text="StackPanel 1" />
            <TextBlock Text="StackPanel 1" />
        </StackPanel>
        <StackPanel Orientation="Vertical" Grid.Column="1">
            <TextBlock Text="StackPanel 2" />
            <TextBlock Text="StackPanel 2" />
            <TextBlock Text="StackPanel 2" />
            <TextBlock Text="StackPanel 2" />
        </StackPanel>
    </Grid>

```

Listá-vezérloelemek feltöltésére programozott módon

```

<StackPanel Orientation="Horizontal">
</StackPanel>
    <Label FontSize="20" HorizontalAlignment="Center" VerticalAlignment="Center">
        <Label>Hello</Label>
    </Label>
<StackPanel Orientation="Horizontal">
</StackPanel>
<!-- Every Listbox contains a horizontal stack panel -->
<StackPanel>
    <Listbox Name="Listbox1" Orientation="Horizontal">
        <Listbox>
            <Label>Hello</Label>
        </Listbox>
    </Listbox>
</StackPanel>

```

Mivel minden a listbox, minden a combobox származtatásátának láncaiban megtalálható a contentcontrol alapozottan, ezért azok egyetérülhetnek a combobox típusú, amely két dimenziós grafikus objektumokat és egy leíró címkeket tartalmazó <stackpanel> tagjai között. Tékinthető attól, hogy a combobox nyolultabb adatokat is tartalmazhatnak.

Tetszőleges tartalom hozzáadása

Ellenben, ha szeretnék, programozott módon felülnéhaentek egy itemscontrol-t. Ezszármazott típusú részen típusos listboxitem objektumok segítségével; a jelenlegi példában azonban ezzel nem nyerünk semmit, hanem valójában csak konsztuktőrrel a content tulajdonásig békálhatunk.

```

<StackPanel>
    <Listbox Name="Listbox1" xmlns:Corlib="System.Collections.Generic">
        <Corlib:String>Nintendo DS</Corlib:String>
        <Corlib:String>Sony PSP</Corlib:String>
        <Corlib:String>Microsoft Xbox 360</Corlib:String>
        <Corlib:String>Sony PlayStation 3</Corlib:String>
        <Corlib:String>Microsoft Xbox</Corlib:String>
        <Corlib:String>Sony PlayStation Portable</Corlib:String>
        <Corlib:String>Microsoft Zune</Corlib:String>
    </Listbox>
</StackPanel>

```

A határvonal a rendszer a toastring() metódust hívja minden egyes listbox-tib. A játék (további részletekért lásd a 28. fejezetet):

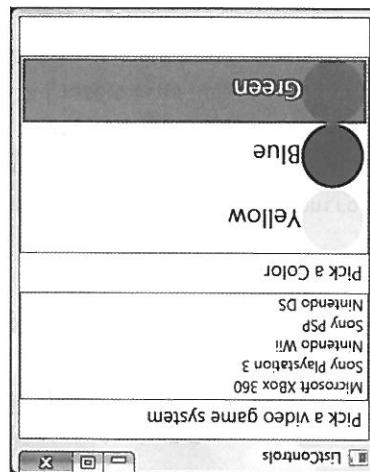
meg kellene határozunk egy új XML-nevteret, hogy beholzzuk az mscorl- nálataval tölteneink fel a listbox (vagy combobox) típusú XAML-ben, akkor boxitem> típuson, így a végrehedmény azonos. Ha system. string névter hozzá- rendelkezünk a közvetlen referenciajukkal.

vagy arrázata az, hogy a XAML használatakor a <listboxitem> típusok kenyelmesebbek abból a szempontból, hogy azokat a http://schemas.microsoft.com/winfx/2006/xaml/presentation XML-nevterben definiáltuk, ezért

A lista box vagy a combobox típus félétolese után a következők nyilvánvaló ker- des az, hogy futásidőben hogyan lehet eldönthetni, melyik elemet választotta ki a felhasználó. Minthá lenti foglalkozik, erre harrom módszer létezik. Ha a kiírás- tottal elem numerikus indexének a kidartható eredkélt bemutatásra, akkor hasz- nálhatók a SelectIndex tulajdonosaiból (amely nulla alapú); a -1 érték azt mutatja, hogy nincs kijelölés). Ha a listaban a kiírásokat objektumot szere- nek megszerezni, arra a SelectItem tulajdonaság való. Végül, a Selected- Value lehetővé teszi, hogy a kiírászott objektum ertekeit megkapjuk (több-

Az aktuális kiállás meghatározása

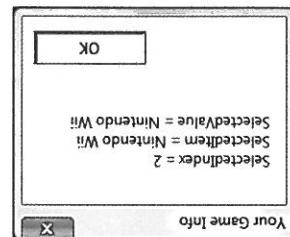
29. 12. ábra: ItecsControl-leszámított típusok tartalmazhatnak bármilyen jeleget tartalmat



A 29.12. ábra mutatja az aktuális lista típusai kimenetét.

Az onban, mi a helyzet a kiválasztott szín megszerzésével?

29.13. ábra: Megtalaáltuk a kiválasztott színgét



Ha a „Nintendo Wii”-t választanánk, akkor a 29.13. ábrán látható szöveg dobjozt kaphatunk.

```
{
    string data = string.Empty;
    protected void btnGetGameSystem_Click(object sender,
                                         RoutedEventArgs args)
    {
        string data = string.Format("SelectedIndex = {0}\n",
                                    SelectedIndex);
        data += string.Format("SelectedValue = {1}\n",
                             SelectedValue);
        data += string.Format("SelectedColor = {2}\n",
                             SelectedColor);
        MessageBox.Show(data, "Your Game Info");
    }
}
```

A bingengetőrendszer a kattintási esemény kezelője megszerzi a List-objektumokat azokat egészítve, és megjeleníti azokat egy úzenet dobjozzában:

```
<!-- Gombok a kiválasztott elem megszerzéséhez -->
<Button Name = "btnGetColor" Click = "btnGetColor_Click">
</Button>
<Button Name = "btnGetGameSystem" Click = "btnGetGameSystem_Click">
</Button>
<Button Name = "btnGetVideoGameSystem" Click = "btnGetVideoGameSystem_Click">
</Button>
Get Video Game System
Get Color
Get Game System
```

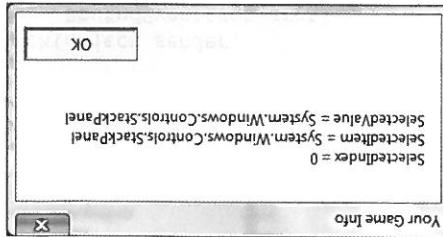
Nem hangszik bonyolultnak, igaz? Most, hogy tesztjelek az egyes tulajdonságok viselkedését, tettezzük fel, hogy definíáltunk két új button típusát az aktuális ablak számára, amelyek egyaránt kezelik a kattintási eseményt:

```

private void buttonColorClick(object sender,
                           EventArgs args)
{
    // A content értekk megszerezésé a stackpanel elemein
    // a kiválasztott label típusban.
    // A kontext逸 erkek megszerezésé a stackpanel elemein
    // stackpanel selekciestack =
    string color = stackpanel1.SelectedStackPanel.Children[0].Content.ToString();
    string[] stackcolor = color.Split('#');
    string[] stackcolor2 = stackcolor[1].Split(',');
    int r = int.Parse(stackcolor2[0]);
    int g = int.Parse(stackcolor2[1]);
    int b = int.Parse(stackcolor2[2]);
    string hexcolor = "#"+r.ToString("X2") + g.ToString("X2") + b.ToString("X2");
    foreach (var item in stackpanel1.Items)
    {
        if (item.GetType() == typeof(Button))
        {
            Button button = (Button)item;
            button.BackColor = ColorTranslator.FromHtml(hexcolor);
        }
    }
}

```

29.14. ábra: Megtalálunk a kiválasztott... StackPanel?



Az aktuális kválasztás meghatározása a békagyározott tartalom esetében

Förrakoskod A fizikai kontináltság körüljárókat a förrakoskodknivalyrat 29. fejezetnek alkonyváltara tar-talmazza. A förrakoskodknivalyratot lásd a Bevezetés xli. oldalán.

Noha ez a megeközelítés egy kicsit tisztább, mint az előző kiserelőtink, vanakk más modok is, ahol adatszablonok segítségevel erhejlik el egy összetett vezérlelőnek eretkezt. Ezhez meg kell ellenírnunk a WPF adatkötési motor multikódeseit, amelyet a fejezet végén vizsgálunk meg.

```

protected void btnGetColor_Clicked(object sender,
                                  RoutedEventArgs args)
{
    string data = string.Empty;
    data += "string format(" + args.ToString() + ",\n";
    data += "    StackPanel1.Tag);\n";
    data += "    MessageBox.Show(data, \"Your Color Info\" );\n";
}

```

Hizen megközölhetés használatával a kódunk jelenlegesen letisztult, mivel programozott módon kiszéddhetőük a Tag tulajdonosághoz rendelt erőket, a következők szerint:

```
<LSBROX Name = "ListColors">
<StackPanel Orientation = "Horizontal" Tag = "Yellow">
<StackPanel Orientation = "Horizontal" Tag = "Blue">
<StackPanel Orientation = "Horizontal" Tag = "Green">
<StackPanel Orientation = "Horizontal" Tag = "Grey">
</StackPanel>
</StackPanel>
</StackPanel>
</StackPanel>
</LSBROX>
```

Noha ez a megoldás megtesszi, mégis elég kényes, mivel minden pozitívának a stacskanál találhatóban (a második gyermek a labda), és számos kasz-tolási mivéletekkel végezhetjük. Márlik alternatíva minden stacskanál tag tulajdonosa gának beállítása, amelyet a Frameworkrel emellett a fejlesztőkben.

A Listbox es a QMBOBQ tipusor nászhatába

```
<label FontSize="15">Is this word spelled correctly?</label>
<stackpanel>
```

A WPF textb ox több soros szövegeket tömögat, és engedélyezí a helyesírás-ellenőrzést:
használhatunk a Label, a TextBox és a Button típusokat (folyékony meg, hogy a
az alábbiak szerint módosítunk az aktuális által XAML-definícióját, hogy
helytelenül írt szavakat vonatkozó javaslatok listáját.
egyel hozzáérhetünk a helyesírástellenőrző-motorhoz, ezáltal megkaphatjuk a
alathizza. Bármi meg jobb, hogy az alapú szolgálat programozási modellek
höz hasonlóan – helytelenül írt szavakat az alkalmazás vörös hullámok vonallal
tulajdonsgárt kijelöl, hogy – a Microsoft Office-
segé rövén elérheti a bevitl adatok helyesírást, ha a spellcheck. Isenből eddig
WPF textb ox több sorak egyszerű fellemzésre, hogy beépített képes-

```
<TextBox Name="txtData" Text="Hello!" BorderBrush="Blue" Width="100"/>
```

A multibán használt más *Textbox* típusokhoz hasonlóan a *WPF Textbox* típusát beállíthatjuk, hogy egy sor mi szövegeket tartalmazzon (ez az alapértelmezett beállítás), vagy több sort, ha az *AcceptText* tulajdonsság értéke true. A *Textbox* beállításai közül, hogy milyen szövegeket tartalmazzon (ez az alapértelmezett beállítás), amelyet a *Text* tulajdonsság segítségével beállíthatunk esetleg olyan sztringtípus, adatokat minden karakteradatot kent kezeljük, ezet a „tartalom” mindejel az adatokat tartalmazó *Text* tulajdonsság segítségével beállíthatunk esetleg olyan sztringtípus,

A Textbox típus használata

A WPF több olyan felhasználófelület-elemet biztosít, amelyek lehetővé tezik a szöveggalapú bevitelre és a szövegegyüttesekre. A két leggyakrabban típusa a szövegesállapú bevitelre és a jelszavak kezelésére. A többi részben bemutatott Textbox és a Passwordbox, amelyeket most a Textcontrols névű, új Visual Studio 2008 WPF alkalmazás használataval vizsgálnunk meg.

A több soros szövegbeviteli mezők használata

A kód megeléhetősen egyszerű. Csak megkeressük a kurzor jelelnélküli helyet a szövegdobozban a caretindext tulajdonoság használatával, hogy kiírjunk egypteket nem null), akkor véglépedtünk a javaslatok listáján a stíluséről elnevezett suggestions tulajdonság révén. Végül megjelenítük a lehetőségeket egy egyszírű messagebox.show() keresés használatával. A 29.15. ábra lehetőségek tesztjeit futás mutat, amikor a kurzor a helytelentüli rit "automatically" szónál.

```

    }
}

MessageBox.Show(spellingsHints, "Try these instead");

// Mutassuk meg a javaslatokat!
{
    spellingsHints += string.Format("{0}\n", s);
}

foreach (string s in error.Suggestions)
{
    // Készítsük el a helyesírási javaslatok stringjét.
    if (error != null)
        txtData.GetSpellingsError(txtData.CaretIndex);
    SpellingsError error =
        // helyen.
        // Probálunk helyesírási hibát keresni a kurzor aktuális
        // string spellingsHints = string.Empty;
    {
        protected void btnOK_Click(object sender, RoutedEventArgs args)
    }
}

```

Ilyen funkcióval tisztázza a TextBox-ba, akkor a rendszer megjelöl a helyesírási hibákat. Az szavakat a TextBox-ba, amikor helytelentüli trinak be szavakat a TextBox-asellenőrzőnek befejezéséhez a következőképpen módosít-e egyszírű helyesírással mindenrőlük szerinti kattintásai eseményeketől. Az szövegdobozban a caretindext tulajdonosának befejezéséhez a következőképpen módosít-suk a gomb kattintásai eseményeketől:

```

</Window>
</StackPanel>
<Button Name="btnOK" Content="Get Spellings" Click="btnOK_Click"/>
<TextBox>
    Name="txtData" FontSize="12"
    BorderBrush="Blue" Height="100" Width="100"
    </TextBox>
<TextBlock SpellCheck.IsEnabled="True" AcceptsReturn="True">
    A többösörs szövegbeviteli mezők használatá

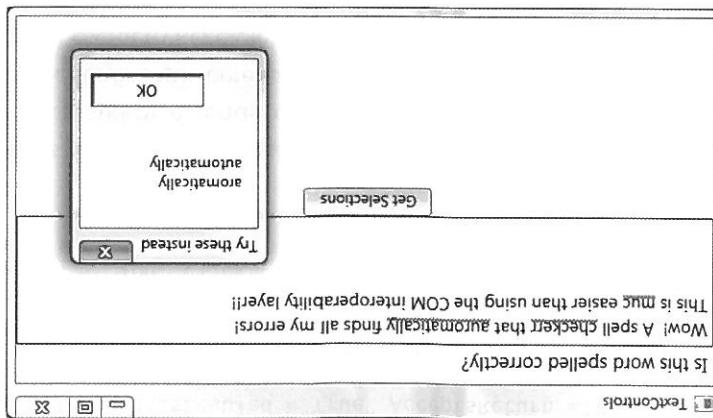
```

```
<StackPanel>
    <Label FontSize="15">Is this word spelled correctly?</Label>
    <TextBox SpellCheck.IsEnabled="True" AcceptsReturn="True"
        Name="txtFavoriteColor" FontSize="14" BorderBrush="Blue" Height="100" />
    <TextBlock FontSize="14" Text="This word spelled correctly?">
        <StackPanel Orientation="Horizontal" Margin="10">
            <TextBlock FontSize="12" Text="Label" />
            <TextBlock FontSize="12" Text="Text Box" />
            <TextBlock FontSize="12" Text="Text Block" />
        
```

A passwordbox típus – nem megfelelően – egy olyan biztonságos helyet kínál, aholova érzékeny szöveges adatokat röhthünk be. Alapértelmezés szerint a jelszavak karaktereit típusni, ez azonban megalátottattható a passwordchar tulajdonság használatával. A végelessen által bérített megszerezéshez egy szereűen ellenorizzük a Password tulajdonását. Modositunk a jelenlegi helyes-szövegben a karaktereket. A végelessen által bérített megszerezéshez egy-ütválas-ellenirázó alkalmazásunkat a helyes jelszó kerésevel, hogy lathassuk a helyes-ellenirázó (password checker) működését. Először modositunk a már megfelelő *stac-kpanel* tárolót egy beágyazott (*stac-kpanel*) típusúval, amely vizsgázza a megfelelő (*button*) típus mellett a passwordbox mezőt:

A Passwordbox típus használata

29. 15. ábra: Egy egységi hellyestris-ellenőrzőj



29. fejzett: Programozás WPF-vezetőlemekekkel

It találunk olyan típusokat, amelyek lehetővé teszik *folyoszöveg-dokumentumok* (flow document) készítését, és ezek segítségevel programozott módon tölthetők ki a weboldalra. A kódokban a *ASP.NET* részleteket, például a *Table*, *Form* és *Text* kontrollak használatát tanulhatjuk meg. A könyv minden részben a *ASP.NET* felhasználásával foglalkozik, így minden részben a *ASP.NET* részleteit is megtanulhatjuk.

A TextBox és a PasswordBox mellett né felejtedik, hogy ha egy olyan alkalmazás kezítnek, amely rendelkezik olyan szövegmmezővel, amely bármilyen típusú tartalomat (grafikus renderelekt, szövegét stb.) magabán foglalhat, a WPF biztosítja számunkra a RichTextBox típusát. Emellett, ha kell a teljesítéshez, a menyügyi különbsen szövegintenzív alkalmazások esetében, a WPF biztosít megoldásokat a dokumentumprezentaciós API-t, amelyet elsősorban a System.Windows.Documents Dokumentus nevűter kepvisele.

```
public partial class Mainwindow : System.Windows.Window
{
    protected void button1_Click(object sender, RoutedEventArgs args)
    {
        if (Checkpassword())
        {
            // Ugyanaz a helyestrás-ellenőrző logika, mint korábban...
            else
                MessageBox.Show("Security error!!");
        }
    }

    private bool Checkpassword()
    {
        if (pwdtext.Password == "Chucky")
            return true;
        else
            return false;
    }
}
```

Mindösszesikkeres volt, íme a reléveans módosítások:
Győződjünk meg arról, hogy a jáváslatok csak akkor jelenhetessenek meg, ha a megfelelő a checkpassword() segédfüggvényt, amely teszeli a kódolt sztringet.

```
<!-- a gomb mindenig az ablak középen van -->
<window x:Class="MyWPFApp.MainWindow"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:xaml="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:fun="http://schemas.microsoft.com/winfx/2006/xaml/fun"
        Title="Fun with Panels!" Height="285" Width="325">
    <button Name="btnOK" Height="100" Width="80">OK</button>
</window>
```

Egy valós WP-Falkamazás mindenekppen taratmaz jó néhány felhasználójára. Aholgy az előző fejezetben elmelekezettünk rá, hogy amikor taratmálat hoztunk el olyan ablakban, több *pantethips* áll rendelkezésünkre. Katt a hozzátóló ablakkal, több *pantethips* áll rendelkezésünkre. Ahhoz, hogy biztosítunk, a WPF-vezérlőelemek megtartásak pozíciójáról. Ablakot vagy akár az ablak egy részét (osztott ablak esetében) átmerehetni az ablaktól vagy akár az elvártak szerint viselkednek, amikor a végfelhasználó átmereheti az ablaktól vagy akár az elvártak szerint viselkedne. Akárunk lenni abból, hogy az elvártak szerint viselkedne, biztosak, hogy a felhasználói felület vezérlőit az új helyükre, billapatosavokat stb.), amelyeket jól kell elrendezni a táróhoz ablakkba. Emlétek, hogy a felhasználói felület vezérlőit szétküldheti, illetélelementet (beviteli vezérlőelemeket, grafikus taratmákat, menürendszereket, alkalmazásokat stb.), amelyeket a felhasználó felületekkel szemben használhat. Ezáltal a felhasználó a felületekkel szemben használhatja a felhasználó felületeit.

A tartalom rendezés kezelése panellek használatával

Források A Textcon-toros Kodályokat a forrásokonnyvár 29. fejezetének alkonyvtára tartalmazza. A forrásokonnyvatrol lásd a Bevezetés xli., oldalat.

29. fejezet: Programozás WPF-vezérlőelemekkel

Bonyolult felhasználói felület készítéséhez osszekerthetők a panelbeleírásoknak, ahol a tervezéskor kerültek; attendeződjönek bárra-jobbra, föl-le; stb. A panelek használatával megadhatók, hogyán viselkedjenek a vezérlők, egyike azt szabályozza, hogy az egyes részletek hogyan helyezkednek el. A rendszem Windows Controls névvel több paneltípuszt biztosít, amelyek mindenhol általánosan használhatók. Ez a paneltípusokat ezáltal nagyjából rögzítik a tartalom elrendezéseit egy adott panelellen belül. A 29.3. táblázat bemutatja néhány gyakran használt WPF Panelbeleírás elem szerepéit.

A WPF alapvető paneltípusai

Nyilvánvaló, hogy nem sok hasznára van egy olyan ablaknak, amely csak egy elemet tartalmazhat. Ha arra van szüksége, hogy egy ablak több elemet tartalmazza, akkor azokat tözsdeleges számla panelbe kell rendezniuk. A paneltervezés során a részleteknek a vezérlőkkel szemben általában a panel típusa meghatározott. A paneltípusokat többféle módon lehet megvalósítani, a leggyakrabban a Content tulajdonságban rendelt objektumként használjuk.

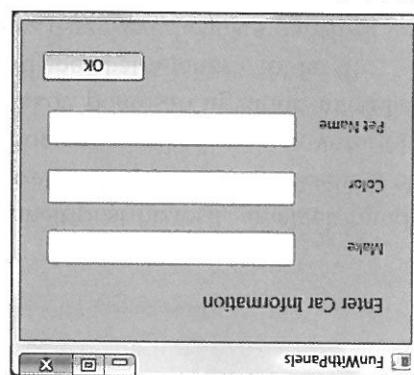
```
<!-- Hiba! A Content tulajdonságát implicit módon többször
     által történt be! -->
<!-- Nyugtával! A Window két közvetlen gyermekelme! -->
<Label Name="lblInstructions" Width="328" Height="27" FontSize="15">Enter
<Car Information>/Label>
<Button Name="btnOK" Height="30" Width="80">OK</Button>
```

Ara is emlékezzünk, hogy ha több elemet próbálnunk elhelyezni közvetlenül egy `(window)` típus hatókörben belül, akkor címkezési és/vagy fordítási idejű hibát kapunk. Ezekenek a hibáknak az az oka, hogy egy ablak (vagy ami azt lététi, a Contentcontrol barátjának leszármazottja) csak egyetlen objektumot rendelhet a Content tulajdonságához:

Vitathatlanul a leggyorsabb panel a Canvas. A Canvas az a panel, amelyben lebegőként elrendezett objektumokat mutat. A Canvas panel lehetővé teszi a mazás alapértelmezett elrendezést utánra.

Tartalom elhelyezése a Canvas paneleken belül

29.16. ábra: A felhasználói felület megjelölése elrendezése



A következő részben bemutatjuk ezeknek a gyakran használt paneltípusoknak a használatát, elkezdtük a 29.16. ábrán látható grafikus felületet külön-állapotot elérni.

29.3. táblázat: Alapfűzfö WPF panelbeli szervezőelemei

Canvas	"Klasszikus" mód a tartalom elhelyezésére. Az elemek pontosan ügyanomot maradnak, ahová a tervezéskor tettük őket.
Grid	Cellák sorozataba rendező a tartalmat, táblázatos rácsozásban.
StackPanel	Függőlegesen vagy vízszintesen egymás után helyező a tartalmat, ahogy azt az orientáció meghatározza.
WrapPanel	Balról jobbra pozicionálja a tartalmat, új sorba fordítva azt a többöt széleire.
Table	Erre a célra van kialakítva a táblázat, az osztályokat sorba rendezve.
DockPanel	Zárója a tartalmat a panel megegyező oldalaira (Top, Bottom, Left vagy Right).
Grid	Cellák sorozataba rendező a tartalmat, táblázatos rácsozásban.
StackPanel	Függőlegesen vagy vízszintesen egymás után helyező a tartalmat.
WrapPanel	Balról jobbra pozicionálja a tartalmat, új sorba fordítva azt a többöt széleire.
Table	Erre a célra van kialakítva a táblázat, az osztályokat sorba rendezve.
Canvas	"Klasszikus" mód a tartalom elhelyezésére. Az elemek pontosan ügyanomot maradnak, ahová a tervezéskor tettük őket.

29. fejezet: Programozás WPF-vezérlőelemekkel

Vagyunk ezzel, hogy az egysések repeatbutoron típusok eggyéi eseményeket lővel hozzájunk a kattintási eseményt. Ezzel a következő C#-logikát írhatunk le, hogy novellük vagy csökkenztük a `<Label>` címkeben megjelentetett értéket (nagyodtan adjunk hozzá tövábbi logikát, ha szerelemek rögzítési a maximális minimális értékét):

```
public partial class Mainwindow : System.Windows.Window
{
    private int curvalue = 0;
```

```

<Window x:Class="CustomSpinsInButtonsApp.MainWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        Title="Custom Spins In Buttons App" Height="300" Width="300">
    <StackPanel>
        <!-- A "F1" gomb -->
        <RepeatButton Height="25" Width="25">
            <Name>repeatAAdvvalUebutton</Name>
            <Click>repeatAAdvvalUebutton_Click</Click>
            <Delay>200</Delay>
            <Interval>1</Interval>
            <Content>+</Content>
        </RepeatButton>
        <!-- A "Le" gomb -->
        <RepeatButton Height="25" Width="25">
            <Name>repeatBAdvvalUebutton</Name>
            <Click>repeatBAdvvalUebutton_Click</Click>
            <Delay>200</Delay>
            <Interval>1</Interval>
            <Content>-</Content>
        </RepeatButton>
    </StackPanel>

```

Windows formsszal ellentetben - a WPF kezdeti kialadasa nem biztosít leírásra-gombok-vezetőt, amely mindenre lehetséges a részletekkel ellátott információk elérésére. A felhasználó számára, hogy a fel- és lefelé nyilak segítségével állíthat be numerikus értékeket. A RepeatButton funkcióval használhatók a kosztosabb viszonylag könnyedén elkezdtethetők léptető- és gomb-vezetőit XAML-ben.

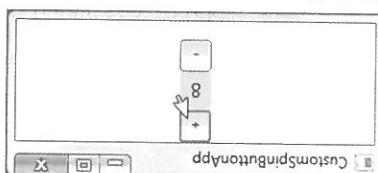
Ennek bemutatásához hozzunk létre új Visual Studio WPF-alakmazas projektet CustomSpinButtonApp néven. Cseréljük le a kezdeti <grid> definíciót olyan <stackpanel> típusúval, amely az alábbi marakkupot tartalmazza:

Ahogy korábban említettük, a CheckBox „az-egy” ToggleButton, „az-egy” ButtonBase, ami elég különösenek trinhez, mivel egy gomb felhasználói felülete megfelelősen különözik egy jelölőnégyzetetől. Azonban, ha a checkbox típusú fejzete

A jelölőnégyzetelek és a radiogombok használata

Forráskód A CustomSpinbuttonApp kodfájljukat a forrásoknávtár 29. fejzetenek alkonyva-tara tartalmazza. A forrásoknávtárrol lásd a Bevezetés részét.

29.6. ábra: Léptetőgomb készítése, kezdőponktuknál a RepeatButton használataival



Ahogyan láthatjuk, amikor a felhasználó rákattint valamelyik RepeatButton gombra, annak megfelelően noveltük vagy csökkentük a privat currvalue értékét, és beállítjuk a label típus Content tulajdonát. A 29.6. ábra mutatja az egyedi léptetőgombunkat működés közben.

```

public Mainwindow()
{
    InitializeComponent();
}

protected void repeatAddValueButton_Click(object sender,
                                         RoutedEventArgs e)
{
    if (currentValue++ > max)
        currentValue = min;
    else if (currentValue < min)
        currentValue = max;
    else
        label.Content = currentValue;
}

protected void repeatRemoveValueButton_Click(object sender,
                                             RoutedEventArgs e)
{
    if (currentValue-- < min)
        currentValue = max;
    else if (currentValue > max)
        currentValue = min;
    else
        label.Content = currentValue;
}

private void repeatButton_Click(object sender, RoutedEventArgs e)
{
    if (sender == plus)
        currentValue++;
    else if (sender == minus)
        currentValue--;
    label.Content = currentValue;
}
}

```

29. fejzet: Programozás WPF-vázéről elemekkel

Hátha tesztelethetők ez a XML-t, azt tapasztalnánk, hogy csak egypt vallászthatunk a hat lehetőségek közül, ami valószerűleg nem áll személyesekben, mivel ket különbszöző csoport van (rádió és szín lehetőségek).

```
<StackPanel>
    <!-- RadioButtons for selecting music media -->
    <Label FontSize="15" Content="Select your Music Media-->
        <RadioButton CD Player/>
        <RadioButton MP3 Player/>
        <RadioButton 8-Track/>
    </Label>
    <!-- RadioButtons for selecting font size -->
    <Label FontSize="15" Content="Select your Font Size-->
        <RadioButton Large/>
        <RadioButton Medium/>
        <RadioButton Small/>
    </Label>
    <!-- RadioButtons for selecting color choice -->
    <Label FontSize="15" Content="Select your Color Choice-->
        <RadioButton Red/>
        <RadioButton Green/>
        <RadioButton Blue/>
    </Label>
    <!-- StackPanel for stack panel -->
</StackPanel>
```

A Radiobutton „az-egy” masik TogglyButton tipus. A checkbox tipussal ellenezve azonban rendelkezik azzal a termesztes kepesseggel, amely biztosita, hogy az gyanaabban a taroloban (stackpanel), grid stb). Levő osszes RadioButtont tipus kollozionosen kizárolagos, es nem igényel tövábbi mutatásra.

29.7. abra: Egy szemre checkbox tipusok

- Send me more information
 Contact me over the phone

```
<StackPanel>
    <StackPanel>
        <!-- Checkbox tipusok -->
        <checkbox Name="checkbox1" checked="checked" /> Phone
        <checkbox Name="checkbox2" /> Send me more information
        <checkbox Name="checkbox3" /> Contact me over the phone
    </StackPanel>
</StackPanel>
```

pusra (mint a button típusra) rakkattintáshatunk, reagál az éger- és billentyűzet-bevitelre, valamint kovethet a WPF-tartalommodellt. A hasonlatos sárgóknamak közszövetségen kiderül, hogy a checkbox típus egyszerűen felülről a Togglere-buttont különöző virtuális tagjait, hogy leterhezesse a jelenlegi szövegben megjelentetését. A többi funkcióval hasonlóan a checkbox típus megjelenítése a következő (checkbox) deklarációkat, amelyek a 29.7. ábrán holgy elválasszuk egy vezetőelem meglehetések annak funkcionálisától).

Amikor rádiogombok vagy jelenlegégyzetelek gyűjteményét tervezzük, megszokott dolgozni vizuális részletekkel körülvenni őket annak ellenére, hogy csökkenheti a használatát. Mivel a header tulajdonság alapvetően szintegy, színes és átlátható, az oldalakon mindenki megérzi, hogy csökkenheti a használatát. Ezért a vezetőknek az oldalakon mindenki megérzi, hogy csökkenheti a használatát. Amikor rádiogombok vagy jelenlegégyzetelek gyűjteményét tervezzük, meg-

A kapcsolódó elemelek összeállítása GroupBox tipusokba

Megjegyzés Alapértelmezés szerint a GroupName értékkel nem rendelkező tarolóban levő összes Radiobutton egyetlen fizikai csoportként működik. Ezért, az elöző példában a színkészleteknek közösnek kell lenniük.

Ezáltal egy második függeléknél beállíthatjuk az egyes logikai csoportosításokat, még akkor is, ha azok ugyanabban a fizikai tarolóban vannak.

```
<!-- A Zenek csoport (tetszőlegesen valasztható ebben a példában,  
lásd a megjegyzést lent) -->  
<Label FontSize = "15" Content = "Select your Music Media"/>  
<RadioButtons GroupName = "Music" >-</RadioButtons>  
<RadioButtons GroupName = "CD Player" >-</RadioButtons>  
<RadioButtons GroupName = "MP3 Player" >-</RadioButtons>  
<Label FontSize = "15" Content = "Select your Color Choice"/>  
<RadioButtons GroupName = "Color" >-</RadioButtons>  
<RadioButtons GroupName = "Green" >-</RadioButtons>  
<RadioButtons GroupName = "Blue" >-</RadioButtons>
```

Ha egyetlen taroló szerelménk több Radiobutton típusával, amelyek különálló fizikai csoportosításokat viselkednek, ez a GroupName tulajdonság beállítása val teljességi következményt hozza:

Logikai csoportosítások létrehozása

```

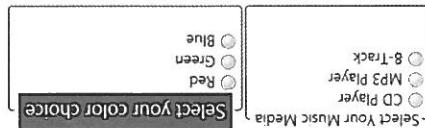
<RadioButton GroupName = "Music" MP3 Player/>/RadioButton>
<RadioButton GroupName = "Music" CD Player/>/RadioButton>
<StackPanel>
<Expander Header = "Select your Music Media" BorderBrush = "Black">
<StackPanel>

```

alapvetően <GroupBox> típusról <Expander> típusra modosul):
 (fél, le, bár) vagy jobbra). Tekintésük meg az alábbi XML-leírást (amely
 tulajdonsság segítségevel meghatározzuk az elemek megjelenítésének irányát
 tök. Ez az elem, az Expander típus lehetővé teszi, hogy az expandibőrrektíon
 menyét, amelyek egy kapszola segítségevel elréthetők, illetve megmutatható-
 külön elemmel, amely csoporthoz olyan felhasználóiról különböző színű eleme-
 két tartalmaz. A szokásos GroupBox típus mellett a WPF rendelkezik olyan új felhasználói fe-

Kapcsolódó elemek közötti csatlakozás

29.8. ábra: RadioButton típusoskatt keretéző GroupBox típusok



A kimenet a 29.8. ábrán látható.

```

<StackPanel>
<Expander Header = "Select your color choice">
<StackPanel>
<RadioButton GroupName = "Blue" Content = "Select your color choice"/>
<RadioButton GroupName = "Red" Content = "Select your color choice"/>
<RadioButton GroupName = "Green" Content = "Select your color choice"/>
</StackPanel>
</Expander>
<GroupBox BorderBrush = "Black">
<Label Background = "Blue" Foreground = "White">
  FontSize = "15" Content = "Select your color choice"/>
</Label>
<GroupBox Header = "Select Your Music Media" BorderBrush = "Black">
<StackPanel>
<RadioButton GroupName = "Music" MP3 Player/>/RadioButton>
<RadioButton GroupName = "Music" CD Player/>/RadioButton>
<StackPanel>

```

Ahogyan azt remélhetünk, a WPF biztosít olyan típusokat, amelyek valászt-ható elemek csoportját tartalmazzák –ilyen a Listbox és a ComboBox, ezek mindenkorban az Itemscntrorl absztrakt osztályból származik. A legfontos-

használata

A Listbox és a ComboBox típusok

Forráskód A CheckRadioGroup.xaml fájlja a forráskódoknnyváról lásd a Bevezetés részét XV. oldalán.

29.10. ábra: Kiterjesztett Expanderek



A 29.10. ábra mutatja az egyes Expanderek kinyitott állapotban.

29.9. ábra: Összecsukott Expanderek



A 29.9. ábra mutatja az egyes Expanderek összecsukott állapotban.

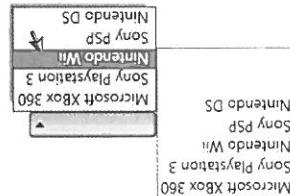
```

</StackPanel>
</Expander>
</StackPanel>
<RadioButton>Blue</RadioButton>
<RadioButton>Green</RadioButton>
<RadioButton>Red</RadioButton>
<StackPanel>
<Expander Header=>
<Label Background="Blue" Foreground="White">
<Expander Header=>
<Expander BorderBrush="Black" BorderThickness="1">
<RadioButton GroupName="Music" Content="Select your color choice"/>
<RadioButton GroupName="Music" Content="CD Player"/>
<RadioButton GroupName="Music" Content="MP3 Player"/>
<RadioButton GroupName="Music" Content="8-Track"/>
<RadioButton GroupName="Music" Content="Radio"/>

```

29. fejezet: Programozás WPF-vézetőlemekekkel

29.11. ábra: Egy személyi ListBox és ComboBox



Nem megelőzöttük a 29.11. ábrán látható rendszerrel a környezetet.

Megjegyzés A combobox típusokat feltölthetjük („listboxitem” elemek segítségével. Ezáltal hozzáérhetünk az ismighi gheted tulajdonasághoz, amelyet a listboxitem típus nem használ.

AZ itemCOLlectiOn típusa a systemCOLlectiOn típusa. Ameiy a részlemekeket tárjoló, erősen típusos itemCOLlectiOn típusa, hogy ez a szolgáltatásról döntmeli az items névű tulajdonságot, amely a részletekkel ellátott, felületeit megadók. Ha már kúp rövén szeretnék egyptiában szerepelni a lista, akkor ezt a boxattem» típusok közé helyezik meg. Példaként nezzük el a következő XAML-kódot:

A Listbox es a Combobox tipusok használata

Az egyik dolgozó, amely különösenek tűnhet a Listbox XAML-leírásban, hogy a *listboxitem* típusokat használunk az Add() metódus meghívása során. Eznek ro-

```

public partial class Mainwindow : System.Windows.Window
{
    public Mainwindow()
    {
        InitializeComponent();
        FillListbox();
    }

    private void FillListbox()
    {
        // Elmek hozzáadása a listamezőhöz.
        listView1.Items.Add("Microsoft Xbox 360");
        listView1.Items.Add("Sony PlayStation 3");
        listView1.Items.Add("Nintendo Wii");
        listView1.Items.Add("Sony PSP");
        listView1.Items.Add("Nintendo DS");
        listView1.Items.Add("Microsoft Xbox");
    }

    private void InitializeComponent()
    {
        // Többféle gamekonzolról szóló lista készítése.
        listView1.Items.Add("Sony PlayStation 3");
        listView1.Items.Add("Microsoft Xbox");
        listView1.Items.Add("Nintendo DS");
        listView1.Items.Add("Sony PSP");
        listView1.Items.Add("Nintendo Wii");
        listView1.Items.Add("Microsoft Xbox 360");
    }
}

```

És a kapcsolódó kódjában az alábbiak szerint töltethetők fel:

```
<Window X:Class="ListControl" MainWindow>
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    Title="ListControls" Height="300" Width="300" >
        <!-- Ezt kód segítségevel töltjük fel -->
        <ListBox Name="listVideoGameConsole" >
            <!-- Listboxban minden elemet külön listboxba helyezzük -->
            <StackPanel>
                <StackPanel>
```

A lista-vélezetkezésekben lehető adatok gyártan csak intrisidőben ismertek, mivel minden elemet egy adatbázis olvasásból visszakapott értékkel, WC-szövegekkel meghívása vagy különböző felülvásással keletkezik. Amikor programozott módon kell lista vágyy kombobox elemet feltölteniuk, WF-szolgáltatás megújítása vagy különböző felülvásással alapján kell feltölteniuk. Teljeszeti fejl, hogy új listcentrális névű Visual Studio 2008 WF-alkalmaztás projektünk van. A listvideoamecosolé típus előző XAML-deklarációja láthatók. Tetelezzük fel, hogy új listcentrális névű Visual Studio 2008 WF-alkalmaztás projektben, egyszerűen használjuk az ItemCollection típus tagjait (Add(), Remove(), Clear()) metódusokat. Az adott típusban a szabványos típusokhoz hasonlóan a többi típus is elérhető.

Lista-vezetőrolélemek feltöltésére programozott módon

```
<StackPanel>
    <!-- Egyp Listbox taratalomma! -->
    <!-- Litsbox Name = "ListColor's" >
    <StackPanel Orientation="Horizontal">
        <StackPanel Orientation="Horizontal" Orientation="Vertical">
            <Label FontSize="20" Height="50" Width="50"/>
            <Label FontSize="20" Height="50" Width="50"/>
        </StackPanel>
        <StackPanel Orientation="Horizontal" Orientation="Vertical">
            <Label FontSize="20" Height="50" Width="50"/>
            <Label FontSize="20" Height="50" Width="50"/>
        </StackPanel>
    </StackPanel>
</StackPanel>
```

Mivel minden a listbox, minden a combobox származtatási láncaiban megtalálható ContentControl alaposszatály, ezért azok egy egyszerű sztringhez jövővel bo- nyolultabb adatokat is tartalmazhatnak. Tehát minden objektumokat és egy leíró címkeket tartal- tipuszt, amely kétdimenziós gráfikus objektumokat és egy leíró címkeket tartal- mazó <stackpanel> tagoljuk meg a comboboxban:

Tetszőleges tartalom hozzáadása

ezelbenben, ha szerelemnek, programozott módon feltölthetünk egy Itemscontrol-konstruktorral a content tulajdonság beállításához).

A hattebén a rendszer a `TestString()` metódust hívja minden egyes `List`-boxat, amelyen a vezéreltmeny azonos. Ha a system.String nemetér használataval töltének fel a `Listbox` (vagy `ComboBox`) típusú XAML-be, akkor meg kellene határozunk egypti XML-nevetet, hogy beholzzuk az `score`-tib. d11 fájlt (további részletekért lásd a 28. fejezetet).

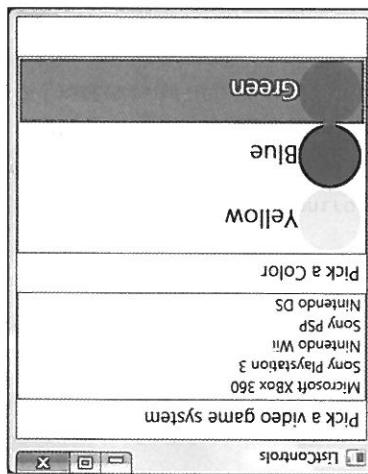
Váld magyarázza az, hogy a XML használatákor a <listboxitem> típusok kebenylemeseknek abból a szempontból, hogy azokat a http://schemas.microsoft.com/winfx/2006/xaml/presentation XML-nevűterben definiáltuk, ezért rendelkezünk a közvetlen referenciajukkal.

nyírte a ToString() metódus meghibásával).

Vállal lehetsévre teszi, hogy a kiválasztott objektum eretkezett megkaptuk (több-nehök megszerezni, arra a SelectedItem tulajdonság való. Végül, a Selected mutatja, hogy melyik kiijelölés). Ha a listában a kiválasztott objektumot szeret-totthátrúk a SelectedIndex tulajdonságát (amely nulla alapú; a -1 eretkezés az elem numerikus indexének a kidobtak előttére átmeneti érték), akkor hasz-ki a felhasználó. Minthogy a többi, erre hárrom módszer letezik. Ha a kiválasz-dés az, hogy futásidőben hogyan lehet eldönthetni, melyik elemet választotta. A lista vagy a combobox típus féléltolésé után a következő nyilvánvaló ker-

Az aktuális kiválasztás meghatározása

29.12. ábra: ItemsControl-lemezről tippusk tartalmazhatnak bármilyen jeleget tartalmat



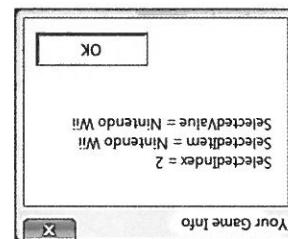
A 29.12. ábra mutatja az aktuális lista típusainak kimeneteit.

```
</StackPanel>
</ListBox>
</StackPanel>
</StackPanel>
<Label FontSize="20" HorizontalAlignment="Center">Green</Label>
<Label FontSize="20" HorizontalAlignment="Center">Blue</Label>
<Label FontSize="20" HorizontalAlignment="Center">Yellow</Label>
<Label FontSize="20" HorizontalAlignment="Center">Pick a Color</Label>
<Label FontSize="20" HorizontalAlignment="Center">Nintendo DS</Label>
<Label FontSize="20" HorizontalAlignment="Center">Sony PSP</Label>
<Label FontSize="20" HorizontalAlignment="Center">Nintendo Wii</Label>
<Label FontSize="20" HorizontalAlignment="Center">Sony PlayStation 3</Label>
<Label FontSize="20" HorizontalAlignment="Center">Microsoft Xbox 360</Label>
<Label FontSize="20" HorizontalAlignment="Center">Pick a video game system</Label>
```

29. fejezet: Programozás WPF-vézérűlegelémekkel

Azban, mi a helyzet a kiválasztott szín megszerzésével?

29.13. ábra: Megtudhatunk a kiválasztott színről



Ha a „Nintendo Wii”-t választanánk, akkor a 29.13. ábrán látható szöveg dobjozt kaphatunk.

```
{
    string data = string.Empty;
    protected void btnGetGameSystem_Click(object sender,
                                         RoutedEventArgs args)
    {
        string data = string.Format("SelectedItemIndex = {0}\n",
                                    SelectedIndex);
        ListViewItem item = new ListViewItem(data);
        ListView.Format("Format", SelectedItem);
        ListView.Items.Add(item);
    }
}
```

A bntGetGameSystem esetében a kattintási esemény kezelője megszerzi a List-View objektumot azonosítókat, és megjeleníti azokat egy úzenetdobjozban:

```
<!-- Gombok a kiválasztott elem megszerzéséhez -->
<Button Name="btnGetColor" Click="btnGetColor_Click">
    <Button>
        <Get Video Game System
            <Button Name="btnGetColor" Click="btnGetColor_Click">
                <Button>
                    <Get Color
                        <Button Name="btnGetColor" Click="btnGetColor_Click">
                            <Button>
```

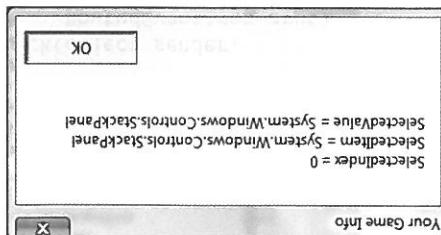
Nem hangszik bonjolultnak, igaz? Most, hogy tesszéjük az aktuális számára, amelyek egyszerűen készílik a kattintási eseményt: döntsök viselkedését, telezzük fel, hogy definiálunk lehetőleg tűsírt az aktuális számára, amelyek egyszerűen készílik a kattintási eseményt:

```

        protected void btnGetColor_Click(object sender,
                                         RoutedEventArgs args)
        {
            // A content eretkezésére szolgáló StackPanel tag
            // a Kiválasztott Label típusban.
            StackPanel selectedStack =
                (StackPanel)listColors.Items[stackColors.SelectedIndex];
            string color =
                selectedStack.Children[0].ToString();
            string data =
                ((Label)(selectedStack.Children[1])).Content.ToString();
            string data = string.Empty;
            data += string.Format("Color = {0}\n", color);
            data += string.Format("SelectedIndex = {0}\n", stackColors.SelectedIndex);
            data += string.Format("Format("Color = {0}", color));
            MessageBox.Show(data, "Your Game Info");
        }
    }
}

```

29.14. ábra: Megtaláltuk a kiválasztott... StackPanel?



Títelezzük fel, hogy a奔getColor Button típus kattintási eseményének kezelőjét. Íme implementálja a奔getColor_Click() metódust, hogy kijelöljük az IstColours objektum aktuális kijelölését, indekét és értékét. Ha most kiválasztunk az első elemet a IstColours listamezőből (és a hozzá tartozó gombra kattintanak), megfelelően kijelöljük a kijelölést, indeket és értéket. Ha most kiválasztunk az utolsó elemet a IstColours listamezőből (és a hozzá tartozó gombra kattintanak), megfelelően kijelöljük a kijelölést, indeket és értéket. Ha most kiválasztunk az utolsó elemet a IstColours listamezőből (és a hozzá tartozó gombra kattintanak), megfelelően kijelöljük a kijelölést, indeket és értéket. Ha most kiválasztunk az utolsó elemet a IstColours listamezőből (és a hozzá tartozó gombra kattintanak), megfelelően kijelöljük a kijelölést, indeket és értéket.

Az aktuális kiválasztás meghatározása a békagyázott tartalom esetében

Forráskód A ListControls Kodfüjlököt a forrásoknnytáról lásd a Bevezetés xlv. oldalát.

Noha ez a megközelítés egy kicsit tisztább, mint az elso kísérletünk, vannak tölelem eretkejt. Ebben még keel értelemben a WPF adatkötési motor működését, más módon is, ahol adatsablonok segítségével erhefűk el egy összetett vezér-

amelylet a fejezet végén vizsgálunk meg.

```
{
    string data = string.Empty;
    protected void btnGetColor_Clicked(object sender,
        RoutedEventArgs args)
    {
        StackPanel stackPanel = new StackPanel();
        stackPanel.Orientation = Orientation.Horizontal;
        foreach (var item in lstColors.Items)
        {
            StackPanel subStackPanel = new StackPanel();
            subStackPanel.Orientation = Orientation.Vertical;
            subStackPanel.Children.Add(new Label { Content = item.ToString() });
            subStackPanel.Children.Add(new TextBlock { Text = " - " });
            subStackPanel.Children.Add(new TextBlock { Text = item.ToString() });
            stackPanel.Children.Add(subStackPanel);
        }
        lstColors.SelectedItem = stackPanel;
    }
}
```

Ezen megközelítés használatával a kódunk jelentősen letisztult, minden programozott módon kiszéddhefűk a Tag tulajdonasághoz rendelt eretkejt, a követ-

kezők szerint:

```
<Listbox Name="lstColors">
    <StackPanel Orientation="Horizontal">
        ...
        <StackPanel Orientation="Vertical">
            ...
            <Label Content="Red" />
            <TextBlock Text=" - " />
            <TextBlock Text="Red" />
        </StackPanel>
        ...
        <StackPanel Orientation="Vertical">
            ...
            <Label Content="Green" />
            <TextBlock Text=" - " />
            <TextBlock Text="Green" />
        </StackPanel>
        ...
        <StackPanel Orientation="Vertical">
            ...
            <Label Content="Blue" />
            <TextBlock Text=" - " />
            <TextBlock Text="Blue" />
        </StackPanel>
        ...
        <StackPanel Orientation="Vertical">
            ...
            <Label Content="Yellow" />
            <TextBlock Text=" - " />
            <TextBlock Text="Yellow" />
        </StackPanel>
    </StackPanel>
</Listbox>
```

Noha ez a megoldás megtesszi, mégis elég kényes, mivel kódolt pozicíók vannak a StackPanel taroloban (a második gyermek a Label), és számos kész-
Tag tulajdonaságának beállítása, amelyet a FrameworkElement alaposztályban
tolás írva lehet kelli végrehabálnunk. Mivelik alternatívára minden StackPanel definícióink:

A Listbox és a ComboBox típusok használata

```

<Label FontSize="15">Is this word spelled correctly?</Label>
<StackPanel>
    <TextControls>
        <TextControl Title="Text Controls" Height="204" Width="292" />
        <TextControl Uri="http://schemas.microsoft.com/winfx/2006/xaml/presentation" Name="xamlUri" />
        <TextControl Uri="http://schemas.microsoft.com/winfx/2006/xaml" Name="xamlName" />
    </TextControls>
    <Window x:Class="TextControls.MainWindow" />

```

Textbox többsoros szövegekkel támogat, és engedélyez a helyesírás-ellenőrzést:

Az alábbiak szerint módosítunk az akutális által XAML-definícióját, hogy helytelennül írt szavakra vonakozó javaslatok listáját.

használhatunk a label, a TextBox és a Button típusokat (folyékony módon), hogy a XMLnél azonban a WPF mindenhol használhatunk a Textbox típust.

Textbox többsoros szövegekkel támogat, és engedélyez a helyesírás-ellenőrzést:

Az alábbiak szerint módosítunk az akutális által XAML-definícióját, hogy helytelennül írt szavakra vonakozó javaslatok listáját.

```

<TextBlock Name="txtData" Text="Hello!" BorderBrush="Blue" Width="100"/>

```

lekerdezhetünk:

A multibán használt más TextBox típusokhoz hasonlóan a WPF TextBox típusát még olyan sztringtípus, amelyet a Text tulajdonosság segítségevel beállíthatunk és belül az adatokat minden karaktereket kezeljük, ezért a „tartalom” min-tulajdonosság értéke true. Ígyen esetekben láttuk fogunk, hogy – a Microsoft Office-sége révén ellenőrizheti a bevitett adatok helyesét, ha a speciális Check. Isenállt kés-

az elérhetőkben a bevitett karaktereket tartalmazzon (ez az alapértelmezett beállíthájúk, hogy egy sornyi szövegeket tartalmazzon).

A TextBox típus használata

A WPF több olyan felhasználói felület-elemet biztosít, amelyek lehetővé teszik a szövegalapú bevitelük összegyűjtését. A két leggyakrabban típus a TextBox és a PasswordBox, amelyeket most a TextControls típus a Studio 2008 WPF alkalmazás használataval vizsgálunk meg.

A többsoros szövegbeviteli mezők használata

A kód megjelentésén egeszerű. Csak megkeressük a kúrzon jelelnégi helyet a szövegdobozban a caretindext tulajdonoság használataval, hogy kiemeliünk a szövegdobozban a caretindext tulajdonoság használataval, amikor a kúrzon helyt mutat, amihez a kúrzon a helytelentül írt „automatically” szón all. Egyeszerű MessageBox.Show() kérés használataval. A 29.15. ábra lehetőségek teszt-

```

    {
        {
            MessageBox.Show(spellHints, "Try these instead");
        }
    }

    string spellingHints = string.Format("{0}\n", s);
}

foreach (string s in error.Suggestions)
{
    // Készítsük el a helyesírást javaslatok sztringjét.
    {
        if (error != null)
            txtData.GetSpellError(error);
        spellError.error =
        // helyen.
        // Próbálunk helyesírást hibát keresni a kúrzon aktuális
        // funkcióitól másik másik miad, hogy amikor helytelentül trinak be
        // string spellingHints = string.Empty;
    }
}
protected void btnOK_Click(object sender, RoutedEventArgs args)
{
    string spellingHints = string.Empty;
}

```

Ilyen funkcióval másik másik miad, hogy amikor helytelentül trinak be szavakat a TextBox-ba, akkor a rendszer megléböl a helyesírást hibákatt. Az egyszerű helyesírás-ellenőrzők befejezéséhez a következőképpen módosít-

```

<TextBox SpellCheck.IsEnabled="True" AcceptsReturn="True">
    Name="txtData" FontSize="12"
    BorderBrush="Blue" Height="100"
    <Button Name="btnOK" Content="Get Spell Suggestions" Click="btnOK_Click"/>
</TextBox>
<StackPanel>
    <Window>

```

```

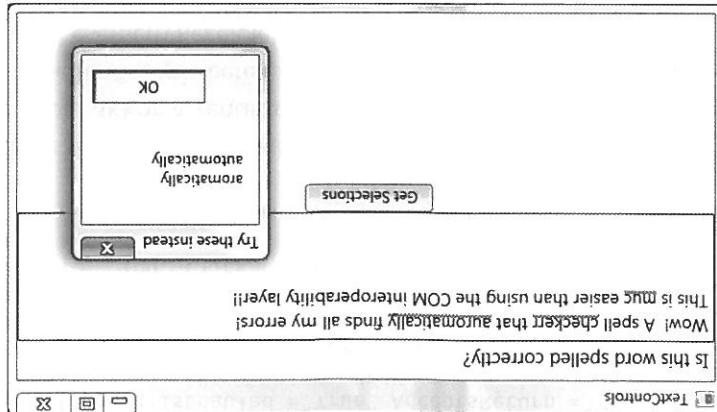
</StackPanel>
</StackPanel>
<Button Name="btnOK" Click="btnOK_Click" Width="100" Height="25" Content="Get Selections" Margin="10"/>
<PasswordBox Name="pwdText" BorderBrush="Black" Width="100" Margin="10"/>
<StackPanel Orientation="Horizontal" Margin="10">
<TextBlock Text="Is this word spelled correctly?" Margin="10"/>
<TextBlock Text="Try these instead" Margin="10"/>
<TextBlock Text="Get Selections" Margin="10"/>

```

A PasswordBox típus - nem megfelelően - egy olyan bizonáságos helyet kínál, hogy a meglévő (button) típus melle a PasswordBox mezoit. A felhasználó a megfelelő típus mellett a PasswordBox mezoit is megadhat, így azonban a meglévő típus mellett a megfelelő típusot használhatja. Ezáltal a megfelelő típusról a megfelelő típus mellett a megfelelő típusot használhatja.

A PasswordBox típus használata

29.15. ábra: Egy egyszerű felhasználói felület a PasswordBox típusról



29. fejezet: Programozás WPF-vézetőlemekekkel

Most methods such as `click()` and `sendKeys()` have side effects on the page. For example, `click()` will change the text value of the input field. To prevent such side effects, we can use `Protected void buttonClick(Object sender, RoutedEventArgs args)`. This method is protected and only accessible from the `MainWindow` class.

```

public partial class MainWindow : System.Windows.Window
{
    ...
    protected void buttonClick(object sender, RoutedEventArgs args)
    {
        if (checkPassword())
        {
            if (chekcButton("Login"))
            {
                // If the login button is clicked, it will
                // trigger the Click() event of the button.
                // To prevent this, we can use the Protected
                // method buttonClick() instead of Click().
                // This way, the Click() event will not be triggered.
                buttonClick(button);
            }
            else
            {
                MessageBox.Show("Security error!");
            }
        }
    }
}

```

`Protected void buttonClick(object sender, RoutedEventArgs args)` method checks if the password is correct using `checkPassword()` method. If the password is correct, it checks if the `Login` button is clicked. If it is, it triggers the `Click()` event of the button using `buttonClick(button)`.

Most methods such as `click()` and `sendKeys()` have side effects on the page. For example, `click()` will change the text value of the input field. To prevent such side effects, we can use `Protected void buttonClick(Object sender, RoutedEventArgs args)`. This method is protected and only accessible from the `MainWindow` class.

```

private bool CheckPassword()
{
    if (pwdText.Password == "Chucky")
    {
        return true;
    }
    else
    {
        MessageBox.Show("Security error!");
        return false;
    }
}

public partial class MainWindow : System.Windows.Window
{
    ...
    protected void buttonClick(object sender, RoutedEventArgs args)
    {
        if (checkPassword())
        {
            if (chekcButton("Login"))
            {
                // If the login button is clicked, it will
                // trigger the Click() event of the button.
                // To prevent this, we can use the Protected
                // method buttonClick() instead of Click().
                // This way, the Click() event will not be triggered.
                buttonClick(button);
            }
            else
            {
                MessageBox.Show("Security error!");
            }
        }
    }
}

```

Most methods such as `click()` and `sendKeys()` have side effects on the page. For example, `click()` will change the text value of the input field. To prevent such side effects, we can use `Protected void buttonClick(Object sender, RoutedEventArgs args)`. This method is protected and only accessible from the `MainWindow` class.

```

<!-- Ez a gomb minden az ablak között van -->
<window x:Class="MyWPFApp.MainWindow">
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    Title="Fun with Panels!" Height="285" Width="325">
        <button Name="btnOK" Height = "100" Width="80">OK</button>
    </window>

```

Ahogy az előző feljegyzésben említézettük rám, hogy amikor tartalmaztuk a használóhoz közelről, több paneltípus áll rendelkezésünkre. Azonban a használó általában, hogy biztosítuk, a WPF-vezérlőelemek meghatározott posíciójában. Ahhoz, hogy biztosítunk a felhasználói felület vezetőit a használó általában, hogy az elvártak szerint viselkedjenek, amikor a végfelhasználó általában a vezetőt vagy akár az ablak egy részét (osztott ablak esetében). Ahhoz, hogy biztosítunk a felhasználói felület vezetőit az új helyükre, általában leírni kell, hogy melyik vezetőt szeretnék a felhasználóhoz közelről, amelyiket a felhasználó általában a vezetőt szerint viselkedjenek, amikor a végfelhasználó általában a vezetőt vagy akár az ablak egy részét (osztott ablak esetében). Ahhoz, hogy biztosítunk a felhasználói felület vezetőit az új helyükre, általában leírni kell, hogy melyik vezetőt szeretnék a felhasználóhoz közelről, amelyiket a felhasználó általában a vezetőt szerint viselkedjenek, amikor a végfelhasználó általában a vezetőt vagy akár az ablak egy részét (osztott ablak esetében).

Egy valós WPF-alkalmazás mindeneképpen tartalmaz jó néhány felhasználófejlesztő-élémet (beviteli vezetőelemeket, grafikus tartalmat, menürendszereket, listák stb.), amelyeket jól kell elrendezni a felhasználóhoz közelről. Emellett, a felhasználóhoz közelről tartalmaznak a felhasználói felület vezetőit az új helyükre, általában leírni kell, hogy melyik vezetőt szeretnék a felhasználóhoz közelről, amelyiket a felhasználó általában a vezetőt szerint viselkedjenek, amikor a végfelhasználó általában a vezetőt vagy akár az ablak egy részét (osztott ablak esetében). Ahhoz, hogy biztosítunk a felhasználói felület vezetőit a használó általában, hogy az elvártak szerint viselkedjenek, amikor a végfelhasználó általában a vezetőt szerint viselkedjenek, amikor a végfelhasználó általában a vezetőt vagy akár az ablak egy részét (osztott ablak esetében). Ahhoz, hogy biztosítunk a felhasználói felület vezetőit a használó általában, hogy az elvártak szerint viselkedjenek, amikor a végfelhasználó általában a vezetőt szerint viselkedjenek, amikor a végfelhasználó általában a vezetőt vagy akár az ablak egy részét (osztott ablak esetében).

A tartalomelrendezés kezelése panelekkel

Ezzel befejeztük a WPF vezetőelemeket attérkinthetően. A fejezet későbbi részén Latin foglalkoztatásban, hogyán készítünk minden rendszereket, általában leírni kell, hogy melyik vezetőt szeretnék a felhasználóhoz közelről tartalmazzák. A forráskódoknnyitarról lásd a Bevezetés részét XL. oldalat.

Források A TextControls kodfájljukt a forrásoknaknnyitarról 29. fejezetbenek alkonyvtára tartalmazza. A forrásoknaknnyitarról lásd a Bevezetés részét XL. oldalat.

A system, windows, controls, neveret több paneltípuszt biztosít, amelyek minden-egyikhez azt szabályozza, hogy az egyes részletemek hogyan helyezkednek el. A panelok használataval megadhatók, hogyán viselkedjenek a vezérlők, amikor a vezérlőhasználat átmértezi az ablakot: pontosan úgyanott maradják, ahova tervezéskor kerültek; átmértezik a vezérlőt. Bonjövünk felhasználói felület készítéséhez összekerérehívuk a panelbeleit ve-zelelmezéssel (pl. "Egy stakkkpanel tártalemre Dokkpanel"), ezáltal nagyfokú ru-galmasságot és kezellehetőséget biztosítathatunk. Ezután a paneltípusok együt-tudnak dolgozni más dokumentumokszpontról vezérlőelemekkel (mint a View-box, a Textbox, a TextFlow és a Paragraph típusok), hogy többet lehessen fi-nomtatni a tártalem elrendezését egy adott paneleken belül. A 29.3. táblázat be-mutatja néhány gyakran használt WPF panelbelelőt szerepel.

A WPF alapvető paneltipusai

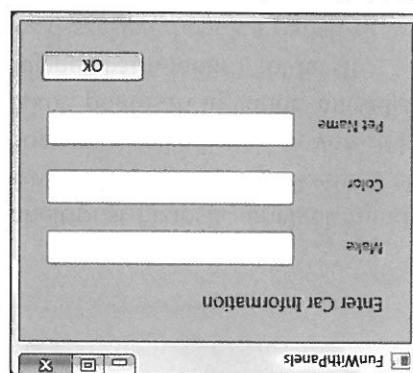
Nyilvánvaló, hogy nem sok hasznában vagy olyan ablaknak, amely csak a kontextusához rendelt objektumként használjuk. Tálatmazza az ablakot képviselő felhasználófejlesztőnek, amely után a panel-tálatmazza, akkor azokat teszszeléges számú panelebe kell rendezniuk. A panel-tálatmazza az ablakot képviselő felhasználófejlesztőnek, amely után a panel-tálatmazza, hogy egy ablak több elemet tartalmazhat. Ha arra van szükség, hogy egy ablak több elemet tartalmaz, először a kontextusához rendelt objektumként használjuk.

```
<!-- Hiba! A ConnectionStrings szövegben több sor  
    a Connent tulajdonságokat implementálta! -->  
    <window x:Class="MyWFApp.MainWindow"  
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml/presentation"  
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml"  
        Title="Fun with Panels!" Height="285" Width="325">  
        <!-- Nyugtatza! A window két közvetlen gyermekelme! -->  
        <label Name="LBInstructions"  
            Width="328" Height="27" FontSize="15">Enter  
            <br/>Car Information</label>  
        <button Name="btNOK" Height="100" Width="80">OK</button>  
</window>
```

Vithatálatban a leggyorszerűbb panel a Canvas. A Canvas az a panele, amelyben leginkább othon erazzik magunkat, úgyanis egy Windows Form alkalmazás alapjának mezt elrendezett utamozza. A Canvas panel lehetővé teszi a

Tartalom elhelyezésére Canvas paneleken belül

79. 16. Adra: A felhasználói felület megcélzott elrendezése



A kovátkrezo részben bemutathatók ezeknek a gyakran használt paneleifűsöknek, amelyeket azonban a használatait, ellégszüflik a 29.16. ábrán látható grafikus felületek tükröz-

29.3. tablázat: Alapvető WPF panelbeírózások

Paneldíheit	Klasszikus” mód a tartalom elhelyezésre. Az elemek pontosan ügyanott maradnak, ahova a tervezéskor tettük őket.
Canvás	Zárolja a tartalmat a panel megadott oldaláról (Top, Bottom, Left vagy Right).
Dockpanelek	Céllakk sorozataba rendezi a tartalmat, tablázatos rácsofrmában.
Grid	Függelégesen vagy vizsgálatra egyptiás után helyezí a tartalmat, ahogy azt az orientáció tulajdonság meghívja.
Stackpanelek	Függelégesen vagy vizsgálatra egyptiás után helyezí a tartalmat, ahogy azt az orientáció tulajdonság meghívja.
Wrappanelek	Balról jobbra pozícionálja a tartalmat, új sorba fordítva azt a többi szövegnek a többihez közel.

Ebbelen a Pedálban a (Canvas) hárkokrén belül levő összes elemet egy Canvas-left kezdetet vezetik minősítő, amelyek a tartalom felé is és bal oldali elhelyezés es egy canvas. Top érték minden belül levő összes elemet egy Canvas-left kezdetet vezetik a panelen belül, a csatolt tulajdonságoknak használataval (lásd a 28. fejezetet). Ahogy kitállíthatunk, a függvények elhelyezkedését a Top, Left, lapon belül, a csatolt tulajdonságoknak használataval lezárva (lásd a 28. fejezetet).

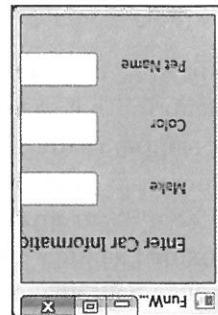
```
<Window  
  xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"  
  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"  
  Title="Fun with Panels!" Height="285" Width="325">  
  <Canvas Background="LightSteelBlue">  
    <Button Canvas.Left="80" Canvas.Top="203" Name="btnOK" Width="80" Height="22" Canvas.ZIndex="14" Click="btnOK_Click" />  
    <Label Canvas.Left="17" Canvas.Top="60" Name="lblInstructions" FontSize="15" Content="Enter Car Information</Label>  
    <Label Canvas.Left="17" Canvas.Top="94" Name="lblMake" Width="60" Height="25" Content="Text Make</Label>  
    <Label Canvas.Left="17" Canvas.Top="109" Name="lblColor" Width="60" Height="25" Content="Text Color</Label>  
    <Label Canvas.Left="17" Canvas.Top="155" Name="lblLabel" Width="60" Height="25" Content="Text Label</Label>  
    <Label Canvas.Left="17" Canvas.Top="193" Name="lblColOr" Width="60" Height="25" Content="Text Col Or</Label>  
    <Label Canvas.Left="17" Canvas.Top="25" Name="lblWidth" Width="60" Height="25" Content="Text Width</Label>  
    <Label Canvas.Left="17" Canvas.Top="94" Name="lblHeight" Width="60" Height="25" Content="Text Height</Label>  
    <Label Canvas.Left="17" Canvas.Top="60" Name="lblDepth" Width="60" Height="25" Content="Text Depth</Label>  
    <Label Canvas.Left="17" Canvas.Top="107" Name="txtColor" Width="153" Height="25" Content="Text Color</Label>  
    <Label Canvas.Left="17" Canvas.Top="153" Name="txtLabel" Width="153" Height="25" Content="Text Label</Label>  
    <Label Canvas.Left="17" Canvas.Top="193" Name="txtWidth" Width="153" Height="25" Content="Text Width</Label>  
    <Label Canvas.Left="17" Canvas.Top="25" Name="txtHeight" Width="153" Height="25" Content="Text Height</Label>  
    <Label Canvas.Left="17" Canvas.Top="94" Name="txtDepth" Width="153" Height="25" Content="Text Depth</Label>  
    <Label Canvas.Left="17" Canvas.Top="60" Name="txtMake" Width="153" Height="25" Content="Text Make</Label>  
    <Label Canvas.Left="17" Canvas.Top="109" Name="txtColor" Width="153" Height="25" Content="Text Color</Label>  
    <Label Canvas.Left="17" Canvas.Top="155" Name="txtLabel" Width="153" Height="25" Content="Text Label</Label>  
    <Label Canvas.Left="17" Canvas.Top="193" Name="txtWidth" Width="153" Height="25" Content="Text Width</Label>  
    <Label Canvas.Left="17" Canvas.Top="25" Name="txtHeight" Width="153" Height="25" Content="Text Height</Label>  
    <Label Canvas.Left="17" Canvas.Top="94" Name="txtDepth" Width="153" Height="25" Content="Text Depth</Label>  
    <Label Canvas.Left="17" Canvas.Top="60" Name="txtName" Width="153" Height="25" Content="Text Name</Label>  
    <Label Canvas.Left="17" Canvas.Top="107" Name="txtPName" Width="153" Height="25" Content="Text PName</Label>  
    <Label Canvas.Left="17" Canvas.Top="153" Name="txtXaml" Width="153" Height="25" Content="Text Xaml</Label>  
    <Label Canvas.Left="17" Canvas.Top="193" Name="txtWinfx" Width="153" Height="25" Content="Text Winfx</Label>  
    <Label Canvas.Left="17" Canvas.Top="25" Name="txtSchema" Width="153" Height="25" Content="Text Schema</Label>  
  </Window>
```

Megjegyzés Ha egy Canvas típuson belül rögzítések nem határozunk meg addit helyet a csatolt tulajdonoság szintaxis részen, akkor azok automatikusan a bal felől sarokba kerülnek.

```
</Canvas>
<Button Canvas.Left="212" Canvas.Top="203" Name="btOK">
    <Canvas>
        <Label Canvas.Left="17" Canvas.Top="60" Name="lblPetName">Name</Label>
        <Label Canvas.Left="17" Canvas.Top="155" Name="lblMake">Pet Name</Label>
        <Label Canvas.Left="17" Canvas.Top="109" Name="lblColor">Color</Label>
        <Label Canvas.Left="17" Canvas.Top="107" Name="lblInstructions">
            FontSize="15">Enter Car Information</Label>
        <Label Canvas.Left="17" Canvas.Top="14" Name="lblTitle">
            Width="328" Height="27"/>
        <Label Canvas.Left="17" Canvas.Top="193" Height="25"/>
        <TextBox Canvas.Left="94" Canvas.Top="107" Name="txtTitle">
            Width="193" Height="25"/>
        <TextBox Canvas.Left="94" Canvas.Top="60" Name="txtPetName">
            Width="193" Height="25"/>
        <TextBox Canvas.Left="94" Canvas.Top="153" Name="txtColor">
            Width="193" Height="25"/>
        <Canvas Background="LightBlue">
            <Canvas>
```

Azután a sorrendje, hogyan deklarálnunk tartalmat egy Canvas típuson belül, befolyásolja az elhelyezkedés kiszámítását, úgyaniis ez a vezérlőelem minden tulajdonságokon alapul. Ezek alapján az alábbi maradván (amely csatornával hasonló vezérlőelemeket) úgyanolyan rendelkezik eredményez:

29.17. ábra: A Canvas panel tartalma lehetővé teszi az abszolút pozícióval.



Mivel minden vezérlőt a <Canvas> elemen belül helyezünk el, így fogjuk, hogy az ablak átmerejtésekor a rendszer eltalálja a vezérlőket, ha a tárroló területe kisebb lesz, mint a tartalom (lásd a 29.17. ábrát).

```

</WrapPanel>
<Button Name="btOK" Width="80">OK</Button>
<TextBox Name="txPName" Width="193" Height="25"/>
<Label Name="lblPName" Width="193" Height="25"/>
<Label Name="txColor" Width="193" Height="25"/>
<Label Name="lblColor" Width="193" Height="25"/>
<Label Name="txMake" Width="193" Height="25"/>
<Label Name="lblMake" Width="193" Height="25"/>
<Label Name="lblInstruction" Width="328" Height="27">Enter Car Information</Label>
<WrapPanel Background="LightBlue">

```

XAML-kód részletei:

Mivel a WrapPanel belül a tartalom nem „rögzül” a panel egy adott oldalra, ezért fontos a sorrend, amelyben az elemeket elrendezjük (a tartalom dallalhoz, majd a WrapPanelen belül a tartalom előtt minden más részletet). Nezzük meg a következő példát, hogy szabályozza az általános meretet a panelről:

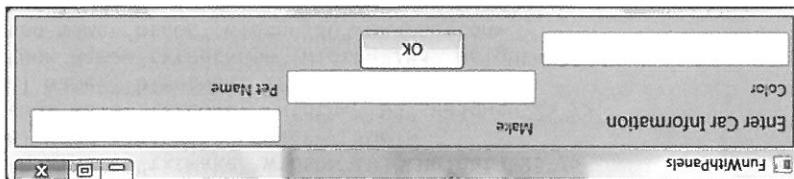
Tartalom elhelyezése a WrapPanel paneleken belül

Forráskód A SimpleCanvas.xaml fájlja forráskódjonyvtár 29. fejezetnek alkalmaztatára tár-

Noha a Canvas típus használata kívánatosnak tűnik, a grafikus tartalom elrendezésétől függetlenül, ha az alkalmazásakor (pl. a betűk mérete nem változik). A másik nyilvántárolt körül az, hogy a Canvas nem problája meg a tartóban meghatározott szablonok alkalmazásakor (pl. a betűk mérete nem változik). A másik részben a WrapPanel a tartalmat a canvasok szabványára alapozva kezeli, ha a canvasok szövege dinamikusan áthelyeződjenek, ha a felhasználó átmérítésével. Amikor a WrapPanel lehetséges tevékenységeit meghatározza a Canvas esetében a Canvas esetében tesszük. Ehelyett minden panelben, nem kell megadunk felület, csak, ha a panelről indulni szeretnék, mint attaliban a Canvas esetében tesszük. Ehelyett minden panelben, ha az alkotó átmérítésétől, amikor elhelyezünk elemeket elegendően, hogy a panelben a tartalom definíálását, amely általában a panelben a tartalom elhelyezésétől függetlenül, melyet a WrapPanel szabványára alkalmazza. A forráskódjonyvtáról lásd a Bevezetés xlvi. oldalát.

A tartalomelrendezés kezelése panelek használatával

29.19. ábra: A WrapPanel megállapítása egy adott elem szelasságát



Rendelés után a 29.19. ábrán látható kimenetet kapjuk (légylejtők meg a Button vezető mértelete es pozícióját).

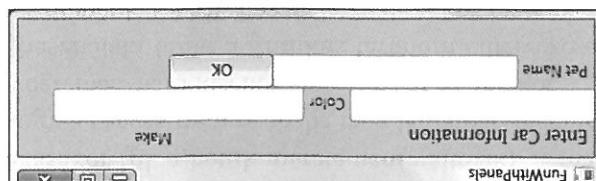
```
</WrapPanel>
<Button Name="btnOK" Width="80">OK</Button>
<TextBox Name="txtPetName" />
<Label Name="lblPetName" Text="Pet Name:</Label>
<Label Name="lblColor" Text="Color:</Label>
<Label Name="lblMake" Text="Make:</Label>
<Label Name="lblInstructions" Text="Instructions:</Label>
<Label Name="lblFontSize" Text="Font size:</Label>
<Label Name="lblHeight" Text="Item height:</Label>
<Label Name="lblWidth" Text="Item width:</Label>
<WrapPanel Background="LightBlue" ItemWidth="200">
```

A WrapPanel (valamint néhány más paneltípus) déklárálik az ItemWidth es tulajdonságát az itemek számával, amelyek az egységek között meghatározzák. Ha egy részszel megadja a saját Height es/Width értékét, akkor az panel által megállapított mérethez viszonyítva teljesít meretezt szabályozzák. Ha minden részszel megadja a saját Height es/az ItemHeight értékkel meghatározásával, minden résznek a saját alapterületei között maradnak el. Tekintse meg a következő markupot:

```
<WrapPanel Background="LightBlue" Orientation="Vertical">
```

Alapértelmezés szerint a WrapPanel tartalma barát jöbbera halad. Ha azonban megváltoztatjuk az orientation tulajdonság értékét Vertical-ra, a tartalom fentől lefelé halad:

29.18. ábra: Egy WrapPanel paneleben a tartalom úgy viselkedik, mint egy közönséges HTML-oldal



Amikor megtekinthük ezt a markupot, a tartalom változtatja a formátát a szelasségeg általékesések, miivel barát jöbbera halad az ablakban (lásd a 29.18. ábrát).

29. fejezet: Programozás WPF-vezetőelemekkel

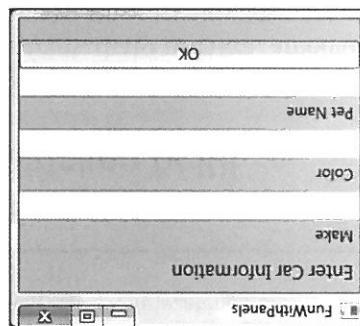
```

<TextBox Name="txtColor"/>
<Label Name="lblColor"><Color/></Label>
<TextBox Name="txtMake"/>
<Label Name="lblMake"><Make/></Label>
<Label Name="lblInfo" FontSize="15">Enter Car Information</Label>
<StackPanel Background="LightBlue">
    <Label Name="tblInstruction">Enter Car Information</Label>

```

A következő markup például a 29.20. ábrán látható kimenetet eredményez:

29.20. ábra: A tartalom függeléges egymás után helyezésére



A WrapPanelhez hasonlóan a StackPanel vezetőelem egy sorba rendezi a tartalmat, amely vizszintesen vagy függelégesen igazitható (ez az alapértelmezett) az orientáció tulajdonsághoz rendeltetik a lapjain. A különböző az elemeket a StackPanel nem kiseríti meg a tartalom becsomagolását, amikor a felhasználó átmerezte az ablakot. Ehelyett, a StackPanel panelben levő elemek egyszerűen megnyúlnak (a teljesen kiszivárgva), hogy illeszkedjenek a StackPanel méretehez.

Tartalom elhelyezése a StackPanel panelekben belül

Forráskód A SimpleWrapPanel.xaml fájl a forráskönyvtár 29. fejezetének alkonyvtára tartalmazza. A forráskönyvtárról lásd a Bevezetés xl. oldalát.

A 29.19. ábra megtekintése után egyetérthetünk abban, hogy a WrapPanel alapján nem a legjobb választás tartalom közvetlen elrendezésére egy ablakban, mivel az elemek összeforradhatnak, ha a felhasználó átmerezte az ablakot. A WrapPanel a legtöbb esetben egy másik Panel típus része, és lehetővé teszi az ablak egy kis területének, hogy átmereze skor becsomagolja a tartalmat, amely vizszintesen vagy függelégesen igazitható (ez az alapértelmezett) az orientáció tulajdonsághoz rendeltetik a lapjai. A különböző elemeket a StackPanel átmerezi az ablakban egymás után helyezve, hogy a tartalom függelégesen elrendezésére szolgál.

3. Tártalom hozzárendelése a racs minden cellájához a csatolt tulajdon-
2. Az egysések sorok meghatározása és konfigurálása.
 1. Az egysések oszlopok meghatározása és konfigurálása.

Végrehajtaniuk:
mindenekben lehet tártalom. Egy grid típusa felosztásához harom lépést kell A HTML-tablázatokhoz hasonlóan a grid típusa felosztásához cellákra, amelyek A WPF API-kon belül kínált panellek közül a Grid minden legrégebbi alkonyvtára

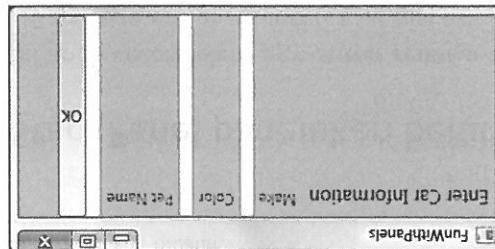
Tártalom elhelyezése a Grid paneleken belül

Forráskód A SimpleStackPanel.xaml fájlja a forráskódjánkat 29. fejezetnek alkonyvtára tartalmazza. A forráskódjánvárról lásd a Bevezetés xl. oldalat.

Ugyanúgy, mint a WrapPanel esetében, ritkán használunk majd a StackPanel tpuszt arra, hogy közvetlenül rendezzük el a tártalmat egy ablakon belül. A Stack-

Panel elhelyett jobban alkalmazható egy福音 panel részpanelekent.

29.21. ábra: A tártalom részletek egymás mellett elhelyezése



<StackPanel Background="LightSteelBlue" Orientation="Horizontal">
Ha az alábbiak szerint horizontál erteke a valtozatuk az orientation tulaj-

donságát, akkor a rendelt kieményt a 29.21. ábrán láthatóval egyezzük:

```

</StackPanel>
<Label Name="lblPetName">Pet Name</Label>
<TextBox Name="txtPetName" />
<Button Name="btnOK">OK</Button>
</StackPanel>
```

29. fejezet: Programozás WPF-vezetőelemekkel

Megjegyzés Ha nem határozunk meg sorokat vagy oszlopokat, akkor a **[Grind]** alapértelmezettségekkel szemben minden cella, amely kijelölt az aktuális felületen. Továbbá, ha nem rendelünk szövezettel, minden cella, amely része a **[Grid]** részleteinek belül, akkor az automatikusan a **[Grid]** osztályba kerül.

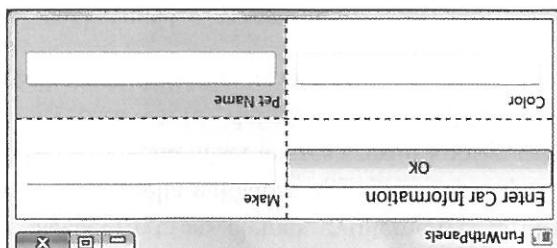
```
    <Grid>
        <Grid.ColumnDefinitions>
            <ColumnDefinition Width="Auto"/>
            <ColumnDefinition Width="Auto"/>
            <ColumnDefinition Width="Auto"/>
        </Grid.ColumnDefinitions>
        <Grid>
            <Grid.RowDefinitions>
                <RowDefinition Height="191" Width="436"/>
                <RowDefinition Height="36"/>
            </Grid.RowDefinitions>
            <Image Source="http://schemas.microsoft.com/winfx/2006/xaml/presentation" />
            <Image Source="http://schemas.microsoft.com/winfx/2006/xaml" />
        </Grid>
    </Grid>
```

A grid típusok az ügynökeinek valasztovalónakat tömögítik. Ahogy bizo-
nyára tudjuk, a valasztovalónak lehetővé teszik, hogy a végfelhasználó átme-
retezze egy rácstípus sorát vagy osztalékot. Ha ez készen van, az egyes átme-
retezhető céllakkban levő tartalom átalakítja magát a beforogtatott elemek alapján.
Nagyön könnyű valasztovalónak hozzáadni egy grid típushoz; egypterűen
definiáljuk a <grid \$p1itter> típusat a csatolt tulajdonságszintaxis használatá-
val, és határozzuk meg, mely sorra vagy osztalopra érvényes. Elgyelűünk arra,
hogy hozzá kell rendelni egy width vagy height értéket (attól függően,
hogyan fog elrendezni a körülbelül meghatározott sorokat). A grid típusnak van az első osztalopon (grid-column = "0":);
látható legyen a kepernyőn. Visszagyűlik még a következő grid típus, amely-

Rácsosk GridSplitter tipusokkal

Törzsaknok A Simplicegyhárd xamli foglalt a törzsaknokonnyvattar 29. részletekben alkonyvittarral tartal-
mazza. A forrásokkal összhangban minden részletet a törzsaknokonnyvattar 29. részletekben alkonyvittarral tartal-

29.22. abra: A Grid panel mukodes kozben



Hagyományok meg, hogy minden elem (belleterve a ráadásokat hozzáadott vlagos-szövegben) egy cellájához kapcsolja magát a Grid. Row es a Grid Recstangl elemeit is) a racs egy cellájához kapcsolja magát a Grid. Row es a Grid. Colunm castolt tulajdonsságok segítségevel. Alapértelmezés szerint a racs-bal felülről kezdődik, amelyet a Grid. Column="0", Grid. Row="0" ad meg. Mivel a mi racsunk összesen negy cellát definiál, jobb alsó cellát a Grid. Colunm="1", Grid. Row="1" azonosítja.

```

        FontSize="15">Enter Car Information</Label>
    <Label DockPanel.Dock="Top" Name="lblInstruction">
        <!-- Elémek önkölcsös a panelhez -->
    <DockPanel LastChildFill="True">

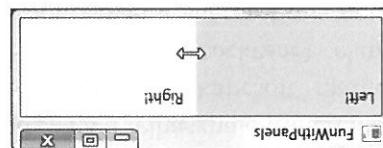
```

A Docckpanelet típusú általában olyan függetlenet használjuk, amelyet tesztelégesen értelmezett) hova csatolja magát a panelen belül. Íme, egy nagyon egyszerű döntéses szintaxis segítségével szabályozzák, hogy a bal felől sarkuk (az alap-számú tövábbi panelt foglalhat magába) a Docckpanelet panellek a csatolt tulaj-sara. Ahogyan a canvas típusnál lattuk, a Docckpanelet típusú csoporthoz tartalom száma több mint a kapcsolódó tartalomhoz.

Tartalom elhelyezése a Docckpanelet paneleken belül

Források A GridWithSplitter.xaml fájl a forrásokonnyvárat 29. fejezetének alkonyvátra tartalmazza. A forrásokonnyvárat lásd a Bevezetés ex. oldalán.

29.23. ábra: Világosítómondákokat tartalmazó Grid típusok



Mindenekigöt vegyük eszre, hogy az osztóp, amely támogatja a választóvo-ezert azok betöltik az egész cellát). Nezzük meg a 29.23. ábrát. (miivel nem adtunk meg Height vagy width tulajdonjágot a label elémekhez, tövönalat, amely lehetővé teszi, hogy átmértezzük az egyes Labels dolgozik. Ha megnevezünk ezt a kiemelést, mindenek helyt kapponk vissza- a csatolt tulajdonas szintaxis segítségével állapíthatunk meg, melyik osztópbal nálá, rendelkezik az Auto. Next width tulajdonaságával, míg a <gridsplitter>

```

        </Window>
    </Grid>
    <Label Name="lblRight" Grid.Column="1" Content="Right!"/>
    <!-- Címke hozzáadása az 1 cellához -->
    <gridSplitter Grid.Column="0" Width="5"/>
    <!-- A választóvonal meghatározása -->
    <Label Name="lblLeft" Background="GreenYellow" Grid.Column="0" Content="Left!"/>
    <!-- Címke hozzáadása a 0 cellához -->

```

```

</scrollviewer>
</stackpanel>
<button Content="First" Background="Blue" Height="40"/>
<button Content="Fourth" Background="Yellow" Height="40"/>
<button Content="Third" Background="Pink" Height="40"/>
<button Content="Second" Background="Red" Height="40"/>
<button Content="First" Background="Green" Height="40"/>
<stackpanel>
</scrollviewer>

```

Erdeemes rámutatni, hogy a WPF biztosítja a `<scrollviewer>` típusú, amely aztomatikusan lapozási viselkedést kínál a beágazott paneltípusok számára:

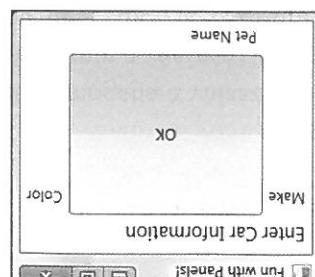
A lapozás engedélyezése a paneltípusoknál

Forráskód A SimpleDockPanel.xaml fájlját a forráskódoknnyájtér 29. fejezetének alkonyvtára tartalmazza. A forráskódoknnyájtárról lásd a Bevezetés Xiv. oldalat.

A DockPanel típusok használatának előnye, hogy ha a felhasználó átmerezi az ablakot, akkor minden elem a panel megeadott oldalahez „kapcsolt” marad trüe értekeré állítja a LastChildFirst attribútumot. Mivel a Button típus nem adott meg semmilyen DockPanel. Dock értéket, ezért kitalált megegyezőt helyet.

Megjegyzés Ha több elemet adunk hozzá egy DockPanel sorrendjében halmozza fel a rendszer, megeadott olyan mellékelt a dékláració sorrendjében halmozza fel a rendszer,

29.24. ábra: Egy egyszerű DockPanel



```

<label DockPanel.Dock="Left" Name="lblMake">Make</label>
<label DockPanel.Dock="Bottom" Name="lblColor">Color</label>
<button Name="btnOK">OK</button>
</dockpanel>

```

29. fejezet: Programozás WPF-vezetőelemekkel

potssaval. Az állapotot még a panelről elérhető, amely megjelenít egy menürendszerrel, vagy eszközökkel, valamint az ablak alján elhelyezett álla-

Célnak olyan elrendezés kialakítása, ahol a főablak rendelkezik egy felső

állítáni.

oldalakon, tehát pillanatnyilag az alapelrendezést és működést fogja kezdeni. Azt alkalmazás frissített verzióját (amely az új, MySpelliCheker nevű Visual

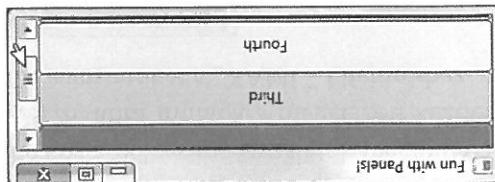
ABLAK KERETEINÉK KÉSZITÉSE BÉÁGYZÁZOTT PANELEK HASZNÁLATÁVAL

egy fómenüt, egy állapotot és egy eszköztárat. (Vagyis a helyesírásteletről alkalmazás funkcióinak használata, hogy támogassa a rendezés-rendezést a főablak számára. Ehhez bóniúk a TextControls projekt bágyázott panellek használatának egy példáját, hogy leterhölzzük a cíornálását szolgáló különöző módszerek attelkintését. Ezután megnezzük a Ezzel befejezük a WPF több paneltűpussinak, valamint a tartalmuk pozí-

ciójában, hogy minden második részet körülvevő extra helyet. Hogyan bánsamodott igényel. Pontosabban a Padding tulajdonosság szabályozza, hogy milyen mékkora plusz helyet kell kiüríteni a részletek között a panelt arrol, hogy a tartalom elhelyezésénél mindenki két erdekes tulajdonságot (Padding és Margin) a WPF-vezérlőelemek mindenike két erdekes tulajdonságot (Padding és Margin) Ahoz, hogy varthatók, minden panel több tagot biztosít, amelyek lehetővé teszik a

Források A ScrollViewer-xam hozzájárult a Bevezetés Xvi. oldalához. A forrásokonvárium lásd a Bevezetés Xvi. oldalát.

29.25. ábra: A ScrollViewer tipus használata



Az előző XAML-definició eredménye a 29.25. ábrán látható.

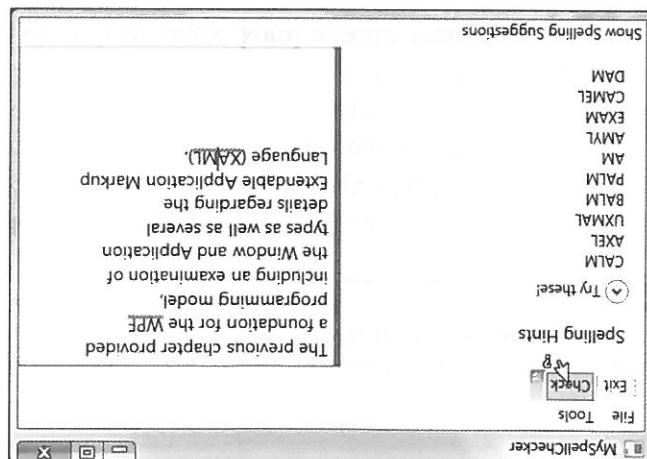
Ablak kereteinek készítése bágyázott panellek használatával

```
<!-- Ez a panele alakítja ki az ablak tartalmát -->
<!DOCTYPE html>
<html>
<head>
<title>Windows -</title>
<meta name="viewport" content="width=device-width, initial-scale=1.0, maximum-scale=1.0, user-scalable=0" />
<link href="css/style.css" rel="stylesheet" type="text/css" />
<script src="js/script.js" type="text/javascript" />
</head>
<body>
<div class="window">
<div class="titlebar">
<span class="close">&amptimes
<span class="minimize">&#9633;
<span class="maximize">&#9635;
</div>
<div class="content">
<p>Windows -</p>
</div>
</div>
</body>
</html>
```

```
    window.x:Class="MySPellCheckr.MainWindow" 
    window.x:Title="MySPellCheckr" Height="331" Width="508"
    window.x:XMLLinks="http://schemas.microsoft.com/infopath/2006/xar"
    window.x:XMLLinks="http://schemas.microsoft.com/windows/2006/xaml"
```

Hogy jelekük meg, hogy a két eszközöt általában nem egy várt képet, hanem egy egész széria szöveges értéket jelent meg. Noha ez nem lenne elégendő egy termék-színkódhoz, képekkel hozzárendelése az eszközök részére gyakorlatilag általában beágyazott ergoforrasok használatát igényli, amelynek temakörte a 3D-jelzetben foglalkzik megviszgálni (tehet most megteszi a szöveges adat). Azt is végyük előre, hogy mikor az egyéb a Check button role kerül, a kúzor megváltozik, az állapot- és a felhasználó részén panelre helyezett úzenetet jeleli meg.

79. 76. ábra: Beágyazott panelek használata egy ablak felhasználói felületeinek letrehozásához



szöveges sort akkor, ha a felhasználó kiválaszt egy menülemezt (vagy) esz-
közöt (pl. gombot), míg a menürendszer es az eszköztár grafikusfelület-kapcsoló-
kakat kiül fel az alkalmazás bezzárásához és a helyesírás-javaslatok megmutatá-
sahez egy Expander vezetőlőben. A 29.26. ábra mutatja a megjelölt kezdeti
elrendezést, megjelenítve a helyesírási javaslatokat a „XAML” kifejezéshez.

```

private int currvalue = 0;
}

public partial class Mainwindow : System.Windows.Window
{
    public Mainwindow()
    {
        InitializeComponent();
        HorizontalAlignment = HorizontalAlignment.Center;
        VerticalAlignment = VerticalAlignment.Center;
        Content = new StackPanel
        {
            Children =
            {
                new Label
                {
                    Name = "repeatmovevaluebutton_Click",
                    Click = "repeatmovevaluebutton_Click",
                    Content = "+",
                    Height = 25,
                    Width = 25
                },
                new Label
                {
                    Name = "repeatadvaluebutton_Click",
                    Click = "repeatadvaluebutton_Click",
                    Content = "-",
                    Height = 25,
                    Width = 25
                }
            }
        };
    }

    private void repeatmovevaluebutton_Click(object sender, RoutedEventArgs e)
    {
        if (currvalue < 300)
        {
            currvalue += 15;
            label1.Content = currvalue.ToString();
        }
    }

    private void repeatadvaluebutton_Click(object sender, RoutedEventArgs e)
    {
        if (currvalue > 0)
        {
            currvalue -= 15;
            label1.Content = currvalue.ToString();
        }
    }
}

```

Windows Formsszal ellentétben – a WPF kezdeti kiadása nem biztosít leptelő- gomb-vezetőelemet, amely lehetővé teszi a felhasználó számára, hogy a fel- és lefelé nyílak segítségével álltson be numerikus értékket. A repeatbuton funkció- onaliásának közönbözőkön belül viszonylag könnyedén elkezthetünk egy leptelő- gomb-vezetőt XAML-ben.

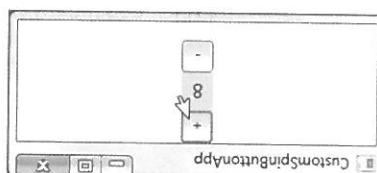
Elnék bemutatásához hozzunk létre új Visual Studio WPF-alakmazás projekt CustomSpinbuttonapp néven. Cseréljük le a kezdeti <grid> definí- ciót olyan <stackpanel> típusnal, amely az alábbi maradványt tarthatmazza:

Ahogy korábban említettük, a **CheckBox**, **“az-egy”** **ToggleButton**, **“az-egy”** **ButtonBase**, ami elég különösenek trühet, mivel egy gomb felhasználói felülete megfelelősen különbözik egy jelölőnégyzetetől. Azonban, ha a **checkbox** típusú fejlesztőknek a **CustomSpinButtonApp** kódjaikat a forrásoknával szerelik.

A jelölőnégyzetelek és a radiogombok használata

Források A **CustomSpinButtonApp** kódjaikat a **RepeatButton** használataival tara tartalmazza. A forrásokonnyvátról lásd a Bevezetés XV. oldalát.

29.6. ábra: Léptetőgomb készítése, vezérlőponthoz a RepeatButton használataival



Ahogy van láthatók, amikor a felhasználó rakkattint valamelyik **RepeatButton** gombra, annak megfelelően novelljük vagy csökkenítik a privat cursorral. Ez teket, es bennéljük a label típus content tulajdonát. A 29.6. ábra mutatja az egyedi léptetőgombunkat működés közben.

```
// Kivon egyet az aktuális értéköt, és megmutatja a címkeben.
    currVal -=;
    tblCurrentValue.Content = currVal;
}
```

```
protected void repeatRemoveValueButton_Click(object sender,
                                             EventArgs e)
{
    protected void repeatAddValueButton_Click(object sender,
                                              EventArgs e)
    {
        // Hozzáad egyet az aktuális értékhez, és megmutatja a címkeben.
        currVal +=;
        tblCurrentValue.Content = currVal;
    }
}
```

```
public MainWindow()
{
    InitializeComponent();
   tblCurrentValue.Content = currVal;
}
protected void repeatAddValueButton_Click(object sender,
                                              EventArgs e)
{
    // Hozzáad egyet az aktuális értékhez, és megmutatja a címkeben.
    currVal +=;
    tblCurrentValue.Content = currVal;
}
protected void repeatRemoveValueButton_Click(object sender,
                                             EventArgs e)
{
    // Kivon egyet az aktuális értéköt, és megmutatja a címkeben.
    currVal -=;
    tblCurrentValue.Content = currVal;
}
}
```

29. fejezet: Programozás WF-vézetőlemekekkel

Két különbsőzű csoporthoz van (rádió és szín lehetőségek).
tunk a hat lehetőség közül, ami valóságnak nem áll számaunkban, mivel a tesztelni ezet a XMAŁ-t, azt tapasztalnánk, hogy csak egyet választhat-

```
</StackPanel>
<RadioButtons StackPanel színeváltászhoz -->
    <Label FontSize = "15" Content = "Select your Color Choice"/>
    <!-- RadioButtons típusok színeváltászhoz -->
    <StackPanel>
        <Label FontSize = "15" Content = "Select your Music Media"/>
        <!-- RadioButtons típusok színeváltászhoz -->
        <StackPanel>
            <Label FontSize = "15" Content = "Select your CD Player"/>
            <!-- RadioButtons típusok színeváltászhoz -->
            <StackPanel>
                <Label FontSize = "15" Content = "Select your MP3 Player"/>
                <!-- RadioButtons típusok színeváltászhoz -->
                <StackPanel>
                    <Label FontSize = "15" Content = "Select your CD Player"/>
                    <!-- RadioButtons típusok színeváltászhoz -->
                    <StackPanel>
                        <Label FontSize = "15" Content = "Select your Radio Button"/>
                        <!-- RadioButtons típusok színeváltászhoz -->
                    </StackPanel>
                </StackPanel>
            </StackPanel>
        </StackPanel>
    </StackPanel>
</StackPanel>
```

szünlakol. Vizsgáljuk meg a kovetkezőt:
A RadioButton "az-egy" másik Togglerebuton típus. A checkbox típusral ellent-
hogy az ügyanabban a tárrolóban (StackPanel), grid stb.) levele összes radió-
tételekben azonban rendelkezik azaz a termesztes kepesességgel, amely biztosítja,
button típus körösösen kizárolagos, és nem igényel tövábbi mutatásra rö-
szünlakol. Vizsgáljuk meg a kovetkezőt:

29.7. ábra: Egyszerű Checkbox típusok

- Contact me over the phone
- Send me more information

```
</StackPanel>
<Checkboxes típusok -->
    <Checkbox Name = "checkboxName" >Send me more information</Checkbox>
    <Checkbox Name = "checkboxName" >Contact me over the phone</Checkbox>
</StackPanel>
```

Látható ki mindenre eredményezik:
Tekintésük attól a kovetkező `<checkbox>` deklarációt, amelyek a 29.7. ábrán
hogy elválasszuk egy vezetőelem meglényestet annak funkciójával).
nemcsak es működését (ússzon egyszerűen a WPF f6 mozgatórugója az,
button különbső virtuális tagjai), hogy leterhözze a jelenlegi zérőt meglé-
szönhetően kidérül, hogy a checkbox típus egyszerűen felülírja a Togglere-
bevitelle, valamint koveti a WPF-taralommodellt. A hasonlatoságoknak kö-
pura (mint a Button típusra) rakkattintathatunk, reagál az egérrel -es billentyűzel-
től.

Amikor rádiogombok vagy jelenleg vizualis tárrolóval körülvenni kell annak felzéséről, még szokott dolgozni vizelkedni. Az általában másdzer ehhez a Groupbox vezetőjével használata. Mivel a header tulajdonság alapvetően system-object típusú szolgáltatók, hozzárendelhetünk bármilyen objektumot (egyszerű szövegek, szövegök, tömbök stb.), hogy feljelölhetjük tulajdonságát. Visszajelzik meg a körülbelül 10 karakteres karakterláncot, amelyek különféle módon foglalják el a Groupbox-t.

A kapcsolódó elemek összeállítása GroupBox

Megjegyzés Alapértelmezés szerint a GroupName értékkel nem rendelkező tárrolóban levo összes Radiobuttontól eltérően fizikai csoporthatáron belül minden Radiobuttonnak a saját pontjának logikai csoporthoz tartozik. Ezért, az eljelzés a saját csoporthoz tartozik, míg a jelentések közöttinél.

Ezáltal egy másikra felügyeltetni lehetőséget kínálhatunk az egységes logikai csoporthoz tartozókat, még akkor is, ha azok ugyanabban a fizikai tárrolóban vannak.

```
<!-- A Zené Csoporthoz -->
<StackPanel>
    <!-- A Zené Csoporthoz -->
    <Label FontSize = "15" Content = "Select your Music Media!"/>
    <Radiobutton GroupName = "Music" >CD Player</Radiobutton>
    <Radiobutton GroupName = "Music" >MP3 Player</Radiobutton>
    <Radiobutton GroupName = "Music" >8-Track</Radiobutton>
</StackPanel>
```

Ha egyptelen tárrolat szeretnénk több Radiobutton típusával, amelyek különálló fizikai csoporthoz köthetők, ez a Groupname tulajdonság beállításával lehetővé válik a Radiobutton típus nyitó elemein:

Logikai csoporthozok létrehozása

```

<RadioButton GroupName = "Music" MP3 Player/>/RadioButton>
<RadioButton GroupName = "Music" CD Player/>/RadioButton>
<StackPanel>
<Expander Header = "Select your Music Media" BorderBrush = "Black">
<StackPanel>

```

A szokásos GroupBox típus mellé a WPF rendelkezik olyan új feházasztálokkal, amelyek csoporthoztak a GroupBox típusra módszerűen (lásd a 29.8. ábrát). A feházasztálok lehetővé teszik, hogy az expandálás során a felület elemeket megjelenítsenek, illetve meghúzzathatnak a felületet, amelyek segítségével meghatározható az elemek meglétülésének irányát (fél, le, balra vagy jobbra). Tekintettel XAML-leírásban (amely a StackPanelban) a StackPanel típusa megfelelően a felületek elrendezését meghatározza az Expandable típus használata. Ez az elem, az Expandable típus lehetővé teszi, hogy az expandálás során a felületet megnyílik, amelyekben a felületet elhelyezhetők.

Kapcsolódó elemek bővítő tipusokba csoportosítása

29.8. ábra: RadioButton típusokat körülvevő GroupBox típusok



A kijelent a 29.8. ábrán látható.

```

<StackPanel>
<GroupBox Header = "Select your color choice">
<RadioButton GroupName = "Music" Content = "Select your color choice"/>
<RadioButton GroupName = "Music" Content = "Red" Selected = "True"/>
<RadioButton GroupName = "Music" Content = "Green"/>
<RadioButton GroupName = "Music" Content = "Blue"/>
<StackPanel>
<Expander Header = "Select your Music Media" BorderBrush = "Black">
<StackPanel>

```

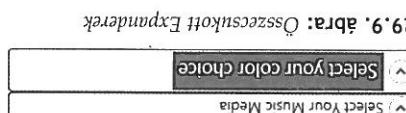
Ahogyan azt remélhetünk, a WPF biztosít olyan típusokat, amelyek választ-
ható elemek csoportját tartalmazzák –ilyen a Listbox és a ComboBox, ezek
mindegyikére az Itemscntrorl absztrakt osztályból származik. A legfonto-

A Listbox és a Combobox típusok használata

Források A ChecKradiodGroup-xamr fajt a forrásokdokonyavtár 29. fejezetnek alkonyavtara tartalmazza. A forrásokdokonyavtáról lásd a Bevezetés xlvi. oldalát.

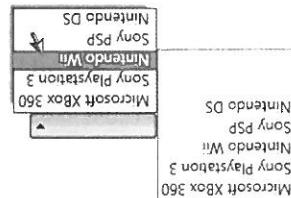
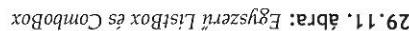


A 29.10. ábra mutatja az egyes expandereket kinyitott állapotban.



A 29.9. ábra mutatja az egyes expandereket összecsukott állapotban.

```
<StackPanel>
    <Expander Header="Expander BorderBrush="Black">
        <Label Background="Blue" Foreground="White">
            FontSize = "15" Content = "Select your color choice!"</Label>
        <Expander Header="Expander Header">
            <RadioButton GroupName = "Music" >8-Track</RadioButton>
            <RadioButton GroupName = "Music" >Blue</RadioButton>
            <RadioButton GroupName = "Music" >Green</RadioButton>
            <RadioButton GroupName = "Music" >Red</RadioButton>
            <RadioButton GroupName = "Music" >Yellow</RadioButton>
            <RadioButton GroupName = "Music" >Purple</RadioButton>
            <RadioButton GroupName = "Music" >Orange</RadioButton>
            <RadioButton GroupName = "Music" >Brown</RadioButton>
            <RadioButton GroupName = "Music" >Grey</RadioButton>
            <RadioButton GroupName = "Music" >Black</RadioButton>
        </Expander>
    </Expander>
</StackPanel>
```



Nem megelőzöttük a 29.11. ábrán látható rendelési listátuk.

Megjegyzés A combobox típusokat feltölthetjük („listboxitem” elemek helyett a listboxitem plus nem használ.

AZ itemcollection típusa a system. object típusosokkal dolgozik, ezért gyakorlatilag bármilyen tartalmazhat. Ha már kúp részen szeretnék egyptezni szöveges adatokkal felülözni egy itemscontrol-lezármaot típusit, akkor ezt a lista boxot írni kell. A típusok között mindenek szerint a használataval tehetsük meg. Példaként nézzük a következő XML-kódot:

A Listbox és a Combobox típusok használata

Az egyik dolog, amely különösenek támhez a ListBox XAML-leírásban, hogy sztringtípusokat használunk az add() metódus meghibásá során. Ennek rö-

```
{
    {
        {
            {
                {
                    {
                        {
                            {
                                {
                                    {
                                        {
                                            {
                                                {
                                                    {
                                                        {
                                                            {
                                                                {
                                                                    {
                                                                        {
                                                                            {
                                                                                {
                                                                                    {
                                                                                        {
                                                                                            {
                                                                                                {
                                                                                                 {
                                                                                                 {
                                                                                                 {
                                                                                                 {
                                                                                                 {
................................................................
```

es a kapcsolódó kodájban az alábbiak szerint töltethető fel:

```

</Window>
</StackPanel>
</ListBox>
<!-- Ezt kid segítségevel töltjük fel -->
<StackPanel>
    <!-- ListControls -->
    <MultiBinding x:Class="ListControls.MainWindow" xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml" Title="ListControls" Height="300" Width="300">
        <!-- ListControls.xaml -->
        <!-- ListControls.xaml -->
        <!-- ListControls.xaml -->
        <!-- ListControls.xaml -->
```

A lista-vézerőelemek levő adatok gyakran csak futásidőben ismertek; lehet a következők szerint lehet XAML-been definiálni:

mazás projektünk van. A ListViewameConsolle típus elölöz XAML-deklarációja tetelezzük fel, hogy új ListControls nevű Visual Studio 2008 WPF-alkal-teniük, egyszerűen használjuk az ItemCollection típus tagjait (add(), Remove()). Amikor programozott módon kell ListView vagy ComboBox elérni felül-teniük. Például, hogy a ListControls elemét egy adatbázis olvasásból visszakapott erre-kek, WCF-szolgáltatás meghívása vagy különböző fájl beolvasása alapján kell felül-például, hogy a listamező elemét egy adatbázis olvasásból visszakapott erre-

Listávezérőelemek felületese programozott módon

```
<StackPanel>
    <!-- Egy Listbox tartalomma -->
    <!-- Listbox Name = "ListControls" >
    <StackPanel Orientation="Horizontal">
        <StackPanel Orientation="Vertical">
            <Label FontSize="20" Height="50" Width="50"/>
            <TextBlock Text="Hello World!">
                <TextBlock>
```

Mivel minden a **listbox**, mind a **combobox** származtatásai lancaban megtalálható a **ContentControl** alaposszatály, ezért azok egy egyszerű szerkezetű struktúrát kaptak. A **listbox** tipusú adatokat is tartalmazhatnak. Tekintse meg a következő combobox példáját:

Ietszölelgeős tartalom hozzáadása

Ellenben, ha szeretnék, programozott módon felróthatunk egy itemscontrol-konstruktorral a content tulajdonság beállításaihoz!:

```
<StackPane>
    <ListBox Name = "ListVideoGameConsole">
        <Corlib:NameSpace>System;assembly=mscorlib</Corlib:NameSpace>
        <Corlib:Type>System.Windows.Forms.ListBox</Corlib:Type>
        <Corlib:Constructor>new System.Windows.Forms.ListBox()
        <Corlib:Properties>
            <Corlib:Property Name = "Items">
                <Corlib:Value>{"Xbox 360", "Sony PlayStation 3", "Microsoft Xbox", "Sony PlayStation 2", "Sony PlayStation", "Nintendo DS", "Corlib:String"}</Corlib:Value>
            </Corlib:Property>
        </Corlib:Properties>
    </ListBox>
</StackPane>
```

A hatteiben a rendszer a `Tostring()` metódust hívja minden egyes `List`-
boxxához, így a végrehedmény azonos. Ha `System.String` névűtert hasz-
nálhatóval töltének fel a `Listbox` (vagy `combobox`) típusú XML-be, akkor
meg kellene határozunk feltüntetett, hogy békuzzuk az mscor-
lób. A fajlt (további részletekért lásd a 28. fejezetet):

magyarázata az, hogy a XML használatákor a `<listboxitem>` típusok kebenylemesébék abból a szempontból, hogy azokat a `http://schemas.microsoft.com/winfx/2006/xaml/presentation` XML-névterben definiálunk, ezért mindenek közötten referenciálunk.

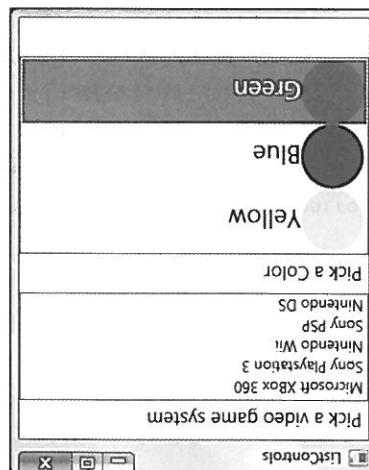
nyírte a `Tostring()` metódus meghívásával).

Változó lehetővé teszi, hogy a kiválasztott objektum ellenére megkapjuk (több-nehely meg szeretni), arra a `SelectItem` tulajdonoság való. Végül, a `SelectItem` mutatja, hogy melyik `MenuItem` a kiválasztott objektumot szereti nállhatóvá a `SelectIndex` tulajdonoság (amely nulla alapú; a -1 ellenére több elem numerikus indexenek a kidobtak el nem válik, akkor használható). Minthogy többet szeretnék, Ha a kiválasztás minden részénél használja a `MenuItem` típusát, minden részben előfordulhat, hogy másik részben a kiválasztásra hivatkozzon.

A `ListBox` vagy a `ComboBox` típus feletti része után a következő nyilvánvaló ker-

Az aktuális kiválasztás meghatározása

29.12. ábra: `ItemsControl`-leszármazott típusok tartalmazhatnak bármilyen jeleget tartalmat

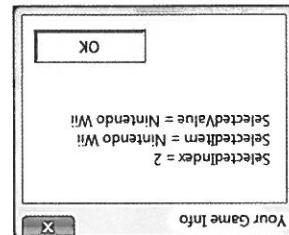


A 29.12. ábra mutatja az aktuális lista típusainak kiemelését.

```
</StackPanel>
</Listbox>
</StackPanel>
<Label FontSize="20" HorizontalAlignment="Center">Green</Label>
<Label FontSize="20" HorizontalAlignment="Center">Blue</Label>
<Label FontSize="20" HorizontalAlignment="Center">Yellow</Label>
<StackPanel Orientation="Horizontal" VerticalAlignment="Center">
<Label Height="50" Width="50" VerticalAlignment="Center">Pick a Color</Label>
<Label Height="50" Width="50" VerticalAlignment="Center">Nintendo DS</Label>
<Label Height="50" Width="50" VerticalAlignment="Center">Sony PSP</Label>
<Label Height="50" Width="50" VerticalAlignment="Center">Microsoft Xbox 360</Label>
<Button>ListControls</Button>
```

Azban, mi a helyzet a kiválasztott szín megszerezésével?

29. 13. ábra: Megtaláltauk a kiválasztott szíringet



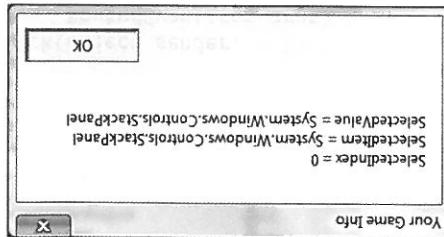
Há a „Nemtendo Wii”-t valasztanak a játékkonzolok listájából, és a kapcsolado gómba kattintanak, akkor a 29.13. ábrán látható szövegdobozt kapnak.

A britegőgamesszystem esetében a kattintási esemény kezelője megszerezza a videogramconsol es objektum selectrendítxét, selecteditem és selectdevállue tulajdonságainak ertekeit, és megjeleníti azokat egy üzenetdobozban:

```
<!-- Gombok a kíválasztott elem megszervezéséhez -->
<Button Name = "btnGetGameSystem" Click = "btnGetGameSystem_Click" ->
    Get Video Game System
    </Button>
    <Button Name = "btnGetColor" Click = "btnGetColor_Click" ->
        Get Color
        </Button>
    </Button>
```

Nem hangozik bonjövöllyük, ígaz? Most, hogy tesztjeiük az egyes trials-donságok viselkedését, teljeszük fel, hogy definíálunk két új button típusát aktuális ablak számára, amelyek egyaránt kezelik a kattintási eseményt:

29. 14. ábra: Megtaláltauk a kiválasztott... StackPanel?



Letrehozuk a tel, hogy a button-tól a button-ig a kattintási eseményenek kezelője implementálja a `onGetColorClick()` metódust, hogy kiírja a `isColor`-t, ha megfelelő színkére a 29-14. ábrán látható kimennettel szembenések.

Az aktuális kválasztás meghatározása a békagyázott tartalom esetében

Források A Litscicontrols kódjaihoz a forrásokhoz köthetően a Bevezetés 29. fejezetben alkonyvátra lár-talmazza. A forrásokhoz köthetően a Bevezetés 46. oldalán.

Noha ez a megeközelítés egy kicsit hszatabb, mint az elso kiserelétnak, vannak mások modok is, ahol adatszablonok segítségével érhetjük el egy összetett vezérlelőt. Ebben meg kell ertelünk a WPF adatkötési motor működését, amelyet a fejezet végén vizsgálunk meg.

Tizen megközeli és használhatóval a kódunk lehetőségen leteszít, mivel programozott módon kiszélethetők a Tag tulajdonosaihoz rendelt értekeit, a következők szerint:

Tárgy tulajdonságának bennállítása, amelyet a framework elemek alapoztatjákban definiálunk:

A WPF TextBox típusnak egyik nagyon egyszerű felülete, hogy kezdetben később több soros szövegeket támogat, és engedélyezi a helyesírás-ellentározást:

```

<Window x:Class="TextControls.MainWindow"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml"
        Title="TextControls" Height="292" >
    <StackPanel>
        <Label FontSize="15">Is this word spelled correctly?</Label>
        <TextBox Name="txtData" Text="Hello!" BorderBrush="Blue"
                Width="100"/>
    </StackPanel>
</Window>
```

Az alábbiak szerint módosításuk az aktuális által XAML-definícióját, hogy helytelennül írja szavakra vonatkozó javaslatok listáját.

használhatunk a Label, a TextBox és a Button típusokat (légeljük meg, hogy a használhatásuk a több soros szövegekkel szemben milyen különbsége van).

Gevel hozzájárhatunk a helyesírástároló-motorhoz, ezáltal megkaphatjuk a tulajdonoság értékét true. Lényen esetekben Latin foglalk, hogy - a Microsoft Office szége révén ellenőrizheti a bármilyen adatot helyesírással, ha a SpellCheck. Isenabla'ded hoz hasonlóan - a helytelennül írt szavakat az alkalmazás több hullámos vonalával körülkerítve tünteti ki.

A multibán használt más TextBox típusokhoz hasonlóan a WPF TextBox típusát használhatunk.

A TextBox típus használata

A WPF több olyan felhasználófelület-élémet biztosít, amelyek lehetségesek a szík a szövegállapú bármelyek összegyűjtésére. A két leggyakrabban típus a TextBox és a PasswordBox, amelyeket most a TextControl típusnak nevezünk. Viszont Studio 2008 WPF alkalmazás használataval vizsgálunk meg.

A több soros szövegbeviteli mezők használata

A Kod megelhetésen szövegdobozban a Carterindek tulajdonsgáhszabályt, hogy ki nyerjük helyet a szavvegezőknek. Csak megkeressük a kurzor jeléhez a szövegheztesen egyszerű. Szóval, A 29.15. ábra lehetséges teszt-egyszerű megoldás. Ez a kiszolgálásnak köszönhetően mindenki elérheti a szövegheztesen egyszerű használatát.

```
protected void buttonClick(object sender, RoutedEventArgs args)
{
    string spellingHints = string.Empty;
    // Probabilunk helyesirasi hibat keressni a kurzor aktualis
    // helyenel.
    spellingError = txtData.GetSpellingError(txtData.CaretIndex);
    if (error != null)
    {
        // Keszitsuk el a helyesirasi javaslatok sztringjét.
        foreach (string s in error.Suggestions)
        {
            spellingHints += string.Format("{0}\n", s);
        }
    }
    // Mutassuk meg a javaslatokat!
    MessageBox.Show(spellings, "Try these instead");
}
```

ilyen funkcionálattassal már látuk majd, hogy amikor helytelenül irunk szavakat a **Textbox**-ba, akkor a rendszer megjelöl a helyesírás törökjeit. Az egyszerű helyesírástellenorzónk befejezéséhez a következőképpen módosítunk: **such a good kattintási eseménykezelője!**

```
    <Textbox SpellenCheck="True" IsEnabled="True" AcceptsReturn="True"
        Name="txtdata" FontSize="12" BorderBrush="Blue" Height="100">
        <Textbox>
            <Button Name="btnOK" Content="Get Selections" Click="btnOK_Click" />
            <StackPanel>
                <WindowName>/Windows</WindowName>
            </StackPanel>
        </Textbox>
    </Textblock>
```

```

        </StackPanel>
    </StackPanel>
</StackPanel>
</TextBlock>
<TextBlock Name="label1" Content="Is this word spelled correctly?">
    <TextBlock>
        <TextBlock>
            <TextBlock>
                <TextBlock>
                    <TextBlock>
                        <TextBlock>
                            <TextBlock>
                                <TextBlock>
                                    <TextBlock>
                                        <TextBlock>
                                            <TextBlock>
                                                <TextBlock>
                                                    <TextBlock>
                                                        <TextBlock>
                                                            <TextBlock>
                                                                <TextBlock>
                                                                    <TextBlock>
                                                                        <TextBlock>
                                                                            <TextBlock>
                                                                                <TextBlock>
                                                                                    <TextBlock>
                                                                                        <TextBlock>
                                                                                            <TextBlock>
                                                                                                <TextBlock>
                                                                                                 

```

Ilyen a megfelelő <button> típus mellett a PasswordBox mezőt:

<Label FontSize="15">Is this word spelled correctly?</Label>

<textbox SpellCheck.IsEnabled="True" AcceptText="True">

<TextBlock SpellingError="True" Word="spelled" Corrected="Corrected">

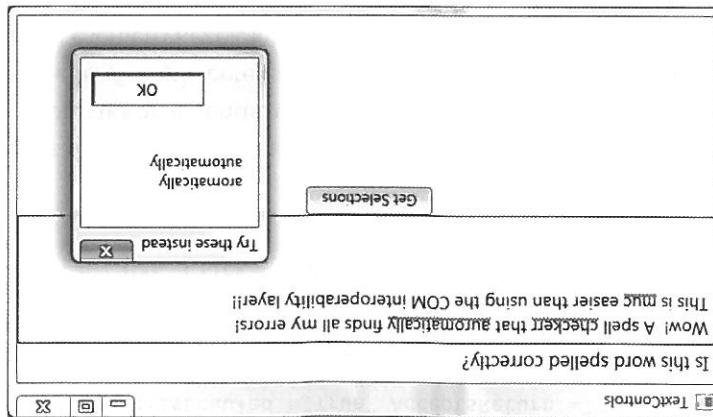
<TextBlock Name="label1" Content="Is this word spelled correctly?">

Fontszínek: 15px, 14px

Először modosítunk a másik mezőt a helyesírást javaslatok listáját. Először modosítunk a másik mezőt a helyesírást alkalmazásunkat a helyes írásra keresve, hogy láthatassuk a részellenorizzel a Password tulajdonságot. Modosítunk a helyesírásban elérhető használataval. A végrehajtásnál általában betek megszerezéséhez egy döntés karakkerekkel típusít, ez azonban megváltoztatható a PasswordChar tulajdonságban. Ahová erzékeny szöveges adatokat írhatunk be. Alapértelmezés szerint a jelszó keretek között írunk.

A PasswordBox típus használata

29.15. ábra: Egy egyszerűbb részellenoriző!



29. fejezet: Programozás WPF-vézetőelemekkel

Most modosituk vagy a jelenlegi Button kattintási eseménykezelőt, hogy megküldjük a checkpassword() segédfüggvényt, amely teszeli a kodolt szöveget. Gyöződjünk meg arról, hogy a javaslatok csak akkor jelenhetnek meg, ha a jelszóellenőrzés sikeres volt. Íme a releváns modositások:

```
<!-- Ez a gomb minden az ablak középen van -->
<window x:Class="MyWPFApp.MainWindow">
    <x:Bind Uri="http://schemas.microsoft.com/winfx/2006/xaml/presentation" x:Name="xaml" />
    <x:Bind Uri="http://schemas.microsoft.com/winfx/2006/xaml" x:Name="xaml2" />
    <Grid>
        <Grid.ColumnDefinitions>
            <ColumnDefinition Width="100" />
            <ColumnDefinition Width="100" />
        </Grid.ColumnDefinitions>
        <Grid.RowDefinitions>
            <RowDefinition Height="50" />
            <RowDefinition Height="*"/>
        </Grid.RowDefinitions>
        <TextBlock Grid.Column="0" Grid.Row="0" Text="A" />
        <TextBlock Grid.Column="1" Grid.Row="0" Text="B" />
        <TextBlock Grid.Column="0" Grid.Row="1" Text="C" />
        <TextBlock Grid.Column="1" Grid.Row="1" Text="D" />
        <TextBlock Grid.Column="0" Grid.ColumnSpan="2" Grid.Row="2" Text="E" />
        <TextBlock Grid.Column="0" Grid.ColumnSpan="2" Grid.Row="3" Text="F" />
        <TextBlock Grid.Column="0" Grid.ColumnSpan="2" Grid.Row="4" Text="G" />
        <TextBlock Grid.Column="0" Grid.ColumnSpan="2" Grid.Row="5" Text="H" />
        <TextBlock Grid.Column="0" Grid.ColumnSpan="2" Grid.Row="6" Text="I" />
        <TextBlock Grid.Column="0" Grid.ColumnSpan="2" Grid.Row="7" Text="J" />
        <TextBlock Grid.Column="0" Grid.ColumnSpan="2" Grid.Row="8" Text="K" />
        <TextBlock Grid.Column="0" Grid.ColumnSpan="2" Grid.Row="9" Text="L" />
        <TextBlock Grid.Column="0" Grid.ColumnSpan="2" Grid.Row="10" Text="M" />
        <TextBlock Grid.Column="0" Grid.ColumnSpan="2" Grid.Row="11" Text="N" />
        <TextBlock Grid.Column="0" Grid.ColumnSpan="2" Grid.Row="12" Text="O" />
        <TextBlock Grid.Column="0" Grid.ColumnSpan="2" Grid.Row="13" Text="P" />
        <TextBlock Grid.Column="0" Grid.ColumnSpan="2" Grid.Row="14" Text="Q" />
        <TextBlock Grid.Column="0" Grid.ColumnSpan="2" Grid.Row="15" Text="R" />
        <TextBlock Grid.Column="0" Grid.ColumnSpan="2" Grid.Row="16" Text="S" />
        <TextBlock Grid.Column="0" Grid.ColumnSpan="2" Grid.Row="17" Text="T" />
        <TextBlock Grid.Column="0" Grid.ColumnSpan="2" Grid.Row="18" Text="U" />
        <TextBlock Grid.Column="0" Grid.ColumnSpan="2" Grid.Row="19" Text="V" />
        <TextBlock Grid.Column="0" Grid.ColumnSpan="2" Grid.Row="20" Text="W" />
        <TextBlock Grid.Column="0" Grid.ColumnSpan="2" Grid.Row="21" Text="X" />
        <TextBlock Grid.Column="0" Grid.ColumnSpan="2" Grid.Row="22" Text="Y" />
        <TextBlock Grid.Column="0" Grid.ColumnSpan="2" Grid.Row="23" Text="Z" />
    </Grid>

```

Egy valós WPF-alkalmazás mindenekppen tarálma az néhány felhasználóhoz. A hosszú időtartamra vonatkozóan a felhasználók számára a legfontosabb a teljesítmény, a felhasználói felület és a felhasználók biztonsága. A felhasználók többsége nem használ webes alkalmazásokat, így a felhasználók számára a legfontosabb a felhasználói felület és a felhasználók biztonsága. A felhasználók többsége nem használ webes alkalmazásokat, így a felhasználók számára a legfontosabb a felhasználói felület és a felhasználók biztonsága.

A tartalomelrendezés kezelése panelek használatával

Ezzel betegesztük a WiFi vezetőelemekszel attékintést. A feljezet későbbi re-szabében látí fogluk, hogyan kezelihetünk menetréndszerkötésekkel. A következőkben azonban annak attékintése, hogyan eszközöttrakat. A kovetkező feladatunk azonban annak attékintése, hogyan rendezhetők a felhasználói felület elemeket minden window típuson belül tervezőleges számu paneltípus használataval.

Főrásokkal A Textconnotors kodfájokat a főrásokkal összefüggően a fejlesztések alkonyvtára tar-talmazza. A forrásokkal összefüggően a Bevezetés lásd a Bevezetés xli., oldalat.

A system. Windows. Controls. Nevetr több paneltípusit biztosít, amelyek minden-
egyike azt szabályozza, hogy az egyes részletemek hogyan helyezkednek el.
A panellek használataval megadhatjuk, hogyan viselkedjenek a vezérlők.
Amikor a vezérlőhasználat átmértezi az ablakot: Pontosan úgyanot maradjá-
nak, ahova tervezésük kerültek; átrendeződjenek bara-jobjra, fél-le; stb.

Bonyolult felhasználói felületek esztétikájához osszekerthetők a panellelű vevők.
Zerlegelmekek (pl. "Egy stacskapanel töratlamaço Dokkpanel"), ezáltal nagyjából ru-
galmaságát és kezelhetőségét biztosítathatunk. Eznefélül a paneltípusok egyszer-
tudnák dolgozni más dokumentumokszámánál vezérlőelemekkel (mint a View-
Box), a TextLock, a TextFlow és a Paragraph típusok), hogy tövább lehessen fi-
nomtatni a tartalom elrendezését egy adott panelellen belül. A 29.3. táblázat be-
mutatja néhány gyakran használt WPF panellelű vezérlőelem szerepéit.

A WPF alapvető paneleitipusai

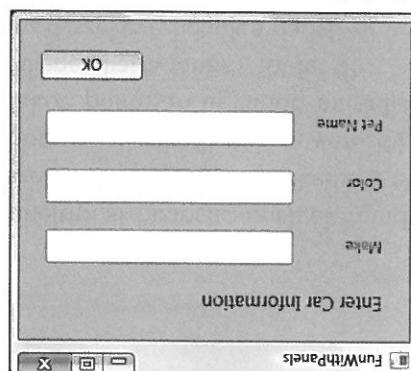
Nyilvánvaló, hogy nem sok haszná van egypti olajban áblaknak, amely csak egypti tarratmazhat. Ha arra van szüksége, hogy egy áblak több elemet tarral-
elemezt tarratmazhat. Ha arra van szüksége, hogy egy áblak több elemet tarrat-
mazzon, akkor azokat tézszőleges számú panelebe kell rendezniuk. A panel tarr-
talmazza az áblakot kepvisele felhasználófejlesztőnek, amely után a pa-
nett magát csak a kontext tulajdonoságba rendelt objektumként használjuk.

```
<!-- Hilba! A Content tulajdonasaigot implementirol modon többeszer  
aljittottuk be! -->  
window x:Class="MyWPFApp.MainWindow"  
xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"  
xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"  
Title="Fun with Panels!" Height="285" Width="325">  
    <!-- Nyugtatza! A Window kozvetlen gyermekelme! -->  
    <Label Name="LBInstructions"  
        Width="328" Height="27" FontSize="15">Enter  
        Car Information</Label>  
    <Button Name="btNOK" Height = "100" Width="80">OK</Button>  
</Window>
```

Vitathatatlanul a leggyorsabb panel a Canvas. A Canvas az a panel, amely mindenkihez elérhető, és mindenki által használható. A Canvas panel lehetségei között a Windows Forms alkalmazásokhoz hasonlóan többféle szolgáltatóval rendelkezik.

Tartalom elhelyezése a Canvas paneleken belül

29.16. ábra: A felhasználói felület megjelenítése elhelyezésre



A következő részben bemutatjuk ezeknek a gyakran használt paneltípusoknak a használatát, elkezdtük a 29.16. ábrán látható grafikus felületet külön-külön. A felületekkel szemben általában ezeknek a gyakran használt paneltípusoknak a használatát, elkezdtük a 29.16. ábrán látható grafikus felületet külön-külön. A felületekkel szemben általában ezeknek a gyakran használt paneltípusoknak a használatát, elkezdtük a 29.16. ábrán látható grafikus felületet külön-külön.

29.3. táblázat: Alapvető WPF panelbeli vezetékű elemek

Canvas	"Klasszikus" mód a tartalom elhelyezésére. Az elemek pontosan ugyanott maradnak, ahova a tervezéskor tettük őket.
Grid	Célalk sorozatba rendezi a tartalmat, tablázatos rácsozásban. Vagy minden sorban a tartalmat a sorba sorolja.
StackPanel	Függelégesen vagy viszintesen egy más után helyezi a tartalmat, ahogy azt az orientáció tulajdonoság megkövönök.
WrapPanel	Balról jobbra pozicionálja a tartalmat, új sorba tördelve azt a törököt, amelyre a következő sorban kerül.
DockPanel	Zárólag a tartalmat a panel megadott oldalára (Top, Bottom, Left, Right).
Grid	Célalk sorozatba rendezi a tartalmat, tablázatos rácsozásban. Vagy minden sorban a tartalmat a sorba sorolja.
StackPanel	Függelégesen vagy viszintesen egy más után helyezi a tartalmat, ahogy azt az orientáció tulajdonoság megkövönök.
WrapPanel	Balról jobbra pozicionálja a tartalmat, új sorba tördelve azt a törököt, amelyre a következő sorban kerül.
DocumentViewer	Zárolja a tartalmat a panel megadott oldalára (Top, Bottom, Left, Right).
Image	Előnézetet mutat a képre, amelyet a program előre generált.
TextBlock	Előnézetet mutat a szöveget, amelyet a program előre generált.

29. fejezet: Programozás WPF-vezetékűelemekkel

Ebben a példában a `<Canvas>` hatókörön belül levő összes elemet egy `Canvas`-re leteve a bottom tulajdonság, míg a vizszintes elhelyezkedést a Left, illetve a Right (lásd a 28. fejezetet). Ahogy kitalálhatunk, a függvények elhelyezkedést a Top, illetve a Bottom tulajdonság, míg a vízszintes elhelyezkedést a Left, illetve a Right (lásd a 28. fejezetet).

Kedveset vezérlik a panelen belül, a csatolt tulajdonságoknak használataval és egy canvas-t. Top érték mindenkit, amelyek a tartalom felére esnek a bal oldali elhelyezésre.

```

</window>

</Canvas>

<Canvas Background="LightSteelBlue">
    <Canvas>
        <Label Canvas.Left="17" Canvas.Top="14" Name="btOk">
            <Button Canvas.Left="22" Canvas.Top="203" Name="btOk">
                <Label Canvas.Left="17" Canvas.Top="14" Name="lblInstructions">
                    <Label Canvas.Left="17" Canvas.Top="80" Name="lblButtons">
                        <Label Canvas.Left="17" Canvas.Top="28" Name="lblButtons">
                            <Textbox Canvas.Left="94" Canvas.Top="60" Name="txtMake">
                                <Label Canvas.Left="17" Canvas.Top="60" Name="lblInstructionCarInformation"/>
                                <Label Canvas.Left="17" Canvas.Top="109" Name="lblColor"/>
                                <Textbox Canvas.Left="94" Canvas.Top="107" Name="txtColor"/>
                                <Label Canvas.Left="17" Canvas.Top="155" Name="lblLabel"/>
                                <Textbox Canvas.Left="94" Canvas.Top="153" Name="txtPetcName"/>
                            </Textbox>
                        </Label>
                    </Label>
                    <Label Canvas.Left="17" Canvas.Top="109" Name="lblLabel"/>
                    <Textbox Canvas.Left="94" Canvas.Top="107" Name="txtColor"/>
                    <Label Canvas.Left="17" Canvas.Top="155" Name="lblLabel"/>
                    <Textbox Canvas.Left="94" Canvas.Top="153" Name="txtPetcName"/>
                </Textbox>
            </Label>
        </Label>
        <Label Canvas.Left="17" Canvas.Top="14" Name="lblButtons">
            <Label Canvas.Left="17" Canvas.Top="60" Name="lblInstructionCarInformation"/>
            <Label Canvas.Left="17" Canvas.Top="109" Name="lblColor"/>
            <Textbox Canvas.Left="94" Canvas.Top="107" Name="txtColor"/>
            <Label Canvas.Left="17" Canvas.Top="155" Name="lblLabel"/>
            <Textbox Canvas.Left="94" Canvas.Top="153" Name="txtPetcName"/>
        </Textbox>
    </Label>
</Canvas>

```

Ahhoz, hogy a `Canvas` típushoz tartalmat adunk, hatozzuk meg a szükséges részleteket a nyílt `<Canvas>` és a zárt `</Canvas>` címkek hatókörében. Valamint adjuk meg a rendelés helyét (egyezzük meg, hogy a tartalom pozíciója relatív lehet a `Canvas` jobb/bal vagy alsó/feloldalhoz képest, de nem mindenhetőző). Ha azt szeretnénk, hogy a `Canvas` területe a tartalomhoz valamit adjuk meg a rendelés helyét (egyezzük meg, hogy a belül, valamint a nyílt `<Canvas>` és a zárt `</Canvas>` címkek hatókörében). Ezáltal a részleteket a nyílt `<Canvas>` és a zárt `</Canvas>` címkek területe.

Köt, a belső tartalom addig nem lesz látható, amíg a tartolót úgyánakkorra a `Canvas` panelen rendeltetett elrendezésűk is lesznek területeire méretezi az álla-

Vagy nagyonból nem terjeszt ki, mint a `Canvas` területe.

Canvas panelen rendeltetett elrendezésűk is lesznek területeire méretezi az álla-

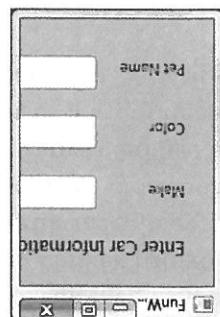
Felhasználói felület tartalmának abszolút elhelyezését. Ha a végfelhasználó a felhasználói felület területére tárlik ki a panelt, amelyet a `Canvas` területre me-

Megjegyzés Ha egy Canvas típuson belüli részletek nem határozunk meg addig helyet a csatolt tulajdonsságoknak, akkor azok automatikusan a bal felől sarokba kerülnek.

```
</Canvas>
<Button Canvas.Left="212" Canvas.Top="203" Name="btOK">
    <Label Canvas.Left="17" Canvas.Top="60" Name="lblPetName">Name</Label>
    <Label Canvas.Left="17" Canvas.Top="107" Name="lblMake">Pet Name</Label>
    <Label Canvas.Left="17" Canvas.Top="155" Name="lblColor">Color</Label>
    <Label Canvas.Left="17" Canvas.Top="193" Name="lblType">Type</Label>
    <Label Canvas.Left="17" Canvas.Top="25" Name="lblInfo" FontSize="15">Enter Car Information</Label>
    <Label Canvas.Left="17" Canvas.Top="44" Name="txtPetName" Width="193" Height="25" />
    <Label Canvas.Left="17" Canvas.Top="60" Name="txtMake" Width="193" Height="25" />
    <Label Canvas.Left="17" Canvas.Top="107" Name="txtColor" Width="193" Height="25" />
    <Label Canvas.Left="17" Canvas.Top="155" Name="txtType" Width="193" Height="25" />
    <Canvas Background="LightBlue">
```

A Canvas típuson belüli részleteket úgyanolyan rendszeresen el lehet meghívni, mint a Canvas. Bottom, a Canvas. Left és a Canvas. Right tulajdonságokon alapul. Ezek alapján az alábbi maradványt (amely csuportosítja a hasonló vezetőlemekeket) írhatunk le:

29.17. ábra: A Canvas panel tartalma lehetőleg tetszi az abszolút pozícióval



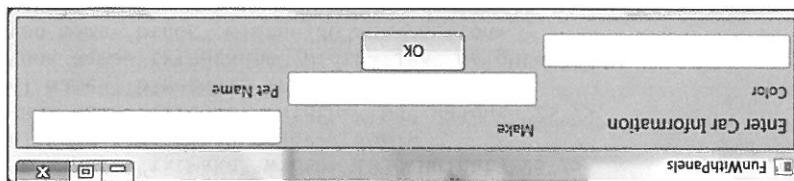
Mivel minden vezetőt a `(Canvas)` elemen belül helyeztük el, látni fogjuk, hogy az ablak átmértezésekor a rendszer eltalakítja a vezetőket, ha a táróhoz terültek kisebb lesz, mint a tartalom (lásd a 29.17. ábrát).

Tartalom elhelyezésére a WrapPanel panelekben belül

Noha a canvas típus használata kiválasztásnak tönhet a táralom eredményese-
khez (mivel olyan ismerősnék tűnik), mégis vanmak korlátai. Először is, egy
canvas típuson belüli elemek nem merevizik át magukat dinamikusan stíli-
sik nyilvánvaló korlát az, hogy a canvas nem probabilis meg lathatóan megtar-
tani az elemeket, amikor a végfelehasználó kisebbre meretezi át az ablakot.
Például, ha egységi képet készítenek XAML használatával, akkor bizonyára
azt akarnak, hogy a vonalak, a formák és a szövegek ügyanazon a helyen ma-
radjanak ahol voltak, hogy dinamikusan áthelyezdjenek, ha a fehérhasználó átmere-
tezi az ablakot. Újra megvizsgáljuk a canvas típus a következő feljelzetben,

A tartalomelrendezés kezeliése panellek használatával

29.19. ábra: A WrapPanel megállapítása egy adott elem szélességét és magasságát



Rendelés után a 29.19. ábrán látható kimenetet kapjuk (figyeljük meg a Button vezető mérétet és pozícióját).

```
</WrapPanel>
<Button Name="btnOK" Width="80">OK</Button>
<TextBox Name="txTPetName"/>
<Label Name="lblPetName">Pet Name</Label>
<TextBox Name="txtColor"/>
<Label Name="lblColor">Color</Label>
<TextBox Name="txtMake"/>
<Label Name="lblMake">Make</Label>
<Label Name="lblInstruction">Enter Car Information</Label>
<Label Name="lblTitle" ItemHeight="30">
<WrapPanel Background="LightSteelBlue" ItemWidth="200">
```

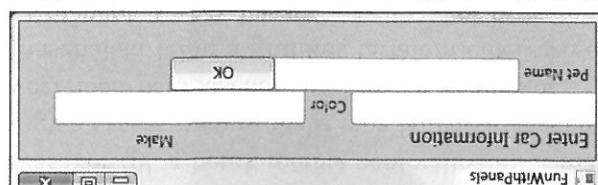
A WrapPanel (valamint néhány más panel típus) déklátható az itemWidth es tulajdonság erőkkel. Amelyek az egységek alapján a panel méreteit szabályozzák. Ha egy részletet meghatározásával, amelyek az itemHeight az itemWidth es tulajdonság erőkkel meghatározzák. Ha azonban megállapításával az orientációt vertikálra, a tárta-

```
<WrapPanel Background="LightSteelBlue" Orientation="Vertical">
```

lom rendgel lefelé halad:

Alapértelmezés szerint a WrapPanel tartalma balról jobbra halad. Ha azon-

29.18. ábra: Egy WrapPanel paneleben a tartalom úgy viselkedik, mint egy közönséges HTML-oldal



seg általánosított, minden barátot jobbra halad az alakban (lásd a 29.18. ábrát).

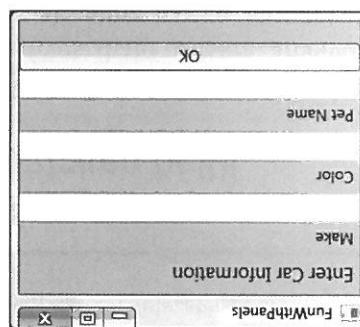
```

<StackPanel Background="LightBlue">
    <Label Name="lblTitle" Content="Enter Car Information"/>
    <Label Name="lblMake" Content="Make"/>
    <Label Name="lblColor" Content="Color"/>
    <Label Name="lblPetName" Content="Pet Name"/>
    <TextBox Name="txtColor" />
    <TextBox Name="txtMake" />
    <TextBox Name="txtTitle" />
    <Label Name="lblFontSize" Content="Font Size: 15" />
    <Label Name="lblInstruction" Content="Enter Car Information" />
</StackPanel>

```

A következő markup például a 29.20. ábrán látható kimenetet eredményez:

29.20. ábra: A tartalom függeléges egymás után helyezése



Egy szerűen megnyúlnak (a rajolasuk alapján), hogy illeszkedjenek a StackPanel-nek átmerevezői az ablakot. Ehelyett, a StackPanel panelben levő elemek használó átmerevezői az orientáció tulajdonságaihoz rendelt eretk alapján. A különbség az, hogy a StackPanel nem kiseri meg a tartalom becsomagolását, amikor a felmeztet) az orientáció tulajdonságaihoz rendelt eretk alapján. A különbség az, tartalmat, amely vizszintesen vagy függelégesen igazította (ez az alapterelmezettséghez hasonlóan a StackPanel vezetőjellem egy sorba rendezi a WrapPanel-hez).

Tartalom elhelyezése a StackPanel panelekben belül

Forrásokból A SimpleWrapPanel.xaml fájl a forrásokbonyvtár 29. fejezetének alkonyvtára tartalmazza. A forrásokbonyvtárral lásd a Bevezetés xlvi. oldalát.

A 29.19. ábra megtékinthető után egyetérthetünk abban, hogy a WrapPanel alapban nem a legjobb választás tartalom közvetlen elrendezésre egy ablakban, mivel az elemek összefoglalóan elrendezésre nem alkalmasak. Lakott. A WrapPanel a legtöbb esetben egy másik paneltipus részéleme lesz, és lehetővé teszi az ablak egy kis területének, hogy átmerevezéskor becsomagolja a tartalmat.

- Ság szintaxis használataval.
3. Taralom hozzárendelése a racs minden cellájához a csatolt tulajdon-
 2. Az egysé sorok meghatározása és konfigurálása.
 1. Az egysé oszlopok meghatározása és konfigurálása.

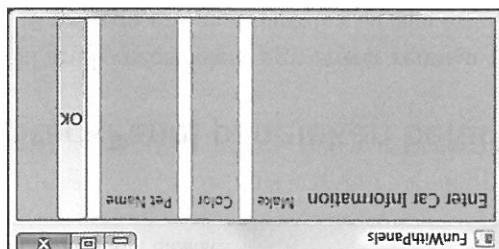
Végrehejtanunk:
 mindenekben lehet taralom. Egy grid típus felosztáshoz harom lepés történik: A HTML-táblázatokhoz hasonlóan a grid típus felosztáshoz cellákra, amelyek A WPF API-kon belül kínált panellek közül a Grid minden legrugalmassabban.

Taralom elhelyezése a Grid paneleken belül

Forráskód A SimpleStackPanel.xaml fájl a forráskönyvtár 29. fejezetnek alkonyvtára tartalmazza. A forráskönyvtárral lásd a Bevezetés 46. oldalát.

Ugyanúgy, mint a WrapPanel esetében, ritkán használjuk majd a StackPanel tpuszt arra, hogy közvetlenül rendezzük el a taralmat egy ablakon belül. A StackPanel elhelyezett jobbán alkalmazható egy őrpanel részpanellekent.

29.21. ábra: A taralom elszíntes egymás mellé helyezése



Ha az alábbiak szerint horizontál ellenkező változatuk az orientation tulajdonágot, akkor a rendelt kiemene a 29.21. ábrán láthatóval egyszerűbb:

```
<StackPanel Background="LightSteelBlue" Orientation="Horizontal">
```

```

<Label Name="lblPetName">Pet Name</Label>
<TextBox Name="txtPetName" />
<Button Name="btnOK">OK</Button>
</StackPanel>

```

29. fejezet: Programozás WPF-vezetőlemekekkel

AZ eljso kert lepest (az oszlopok es a sorok dertmialaszt) a <Grid>.ColumnDefinitions-nak, mivel ezeket az elemek hasznalatval erhejlik el, amelyek a <Grid>.RowDefinitions-nak, illetve a <RowDefinition> elmekek gyuiteményt tartalmaznak. Mivel egy rácson belül minden cella valójában egy igazi NFT-típus, ezért az elmekek megjelenését viselkedését teszik szint konfigurálhatók. Ime, egy meglehetősen egyszerű <Grid> definició, amely a 29.22. ábra szerint az elmekek megjelenését és viselkedését szint konfigurálható módon rendezi el a felhasználói felület tartalmát:

Megjegyzés: Ez a nem működőszínen lévő sorokat vagy szövegeket egy részletekhez a <Grid> típuson belül, akkor az automatikusan a 0. oszlopban elhelyezik.

```

</grid.ColumnDefinitions>
<grid.ColumnDefinition Width="Auto"/>
<grid.ColumnDefinition Width="Auto"/>
<!-- Az osztópok meghatározása -->
<grid.Background="AliceBlue">
<grid.Title="Funkciópanells" Height="191" Width="436">
<grid.Xml="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
<grid.Yml="http://schemas.microsoft.com/winfx/2006/xaml" />
<grid.Window
ben valasztoval van az első osztópon (grid.Column = "0"):

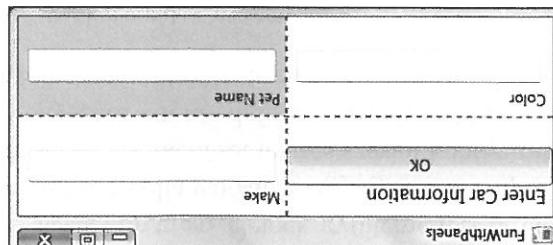
Iátható legyen a kepernyőn. Vizsgáljuk meg a kódot ezeket a grid típus, amely-
hogy függelékes vagy vizszintes osztásról van szó) annak erdekeben, hogy
hogy hozzá kell rendelni egy width vagy height értéket (attól függően,
és határozta meg, mely sorra vagy osztóra elvénysel. Figyeljük arra,
val, hogy hozzávalók a gridspáitterrel típuszt a castolt tulajdonosságzintaxis használatá-
definiáljuk a gridspáitterrel típuszt a castolt tulajdonosságzintaxis használatá-
Nagyön könnyű valasztovalat hozzáadni egy grid típushoz; egszerűen
retezhető cellákban levő tartalom átalakítja magát a befooglalt elemek alapján.
rettezze egy racs típus sort a valasztovalat hozzáadni. Ha ez készén van, az egységek átme-
nyára tudjuk, a valasztovalak lehetségeit teszik, hogy a végfelhasználó átme-
A grid típusok az ügynevezett valasztovalakat támogatják. Ahogy bizo-

```

Rácsok GridSplitter típusokkal

Forráskód A SimpleGrid.xaml fájlja a forráskódokonvárt 29. fejezetnek alkonytvátra tartal-
mazza. A forráskódokonvártól lásd a Bevezetés xlv. oldalát.

29.22. ábra: A Grid panel mutatók közben



Hagyjuk meg, hogy minden elem (beleértve a ráadásokat hozzáadott világos-
zöld Rectangl elemeit is) a racs egy cellájához kapcsolja magát a grid. Row es a
grid.Column csatolt tulajdonosságok segítségével. Alapértelmezés szerint a racs-
ban levő cellák rendezésére a bal felülről sarokban kezdődik, amelyet a grid.
grid.Column="0", grid.Row="0" ad meg. Mivel a mi rácsonk összesen négy cellát defi-
niál, a jobb alsó cellát a grid.Column="1", grid.Row="1" azonosítja.

```

        FontSize="15">Enter Car Information</Label>
    <Label DockPanel="Top" Name="lblInstruction">
        <!-- Elémek dockolása a panelhez -->
    <DockPanel LastChildFill="True">

```

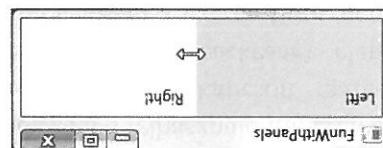
A DockPanel típusú általában olyan függetlenet használjuk, amelyet többelégesen kezelnek. A DockPanel definíciója a 29.24. ábrán látható kimeneteleit eredményez: a DockPanel definíciója, amely a 29.24. ábrán látható kimeneteleit eredményez:

(ettemezett) hova csatolja magát a panelen belül. Íme, egy nagyon egyszerű döntésesztályt is segítségével szabályozzák, hogy a bal felés sarkuk (az alapszámú tövábbi panelt foglalhat magában a kapcsolódó tartalom csoportosítássára. Ahogy az a canvas típusnál látunk, a DockPanel panelek a csatolt tulajdonaságokkal szemben a többi panelt foglalhat magában a kapcsolódó tartalom csoportosítássá.

Tartalom elhelyezése a DockPanel panelekben belül

Forráskód A GridWithPanels.xaml fájl a forráskódoknyitárról lásd a Bevezetés xl. oldalat.

29.23. ábra: Valasztovalakat tartalmazó Grid típusok



Mindenekelőtt vegyük észre, hogy az oszlop, amely támogatja a valasztvonalat, rendelkezik az Auto. Next width tulajdonsegéval, míg a <GridSplitter> nál, rendelkezik az Auto. Next width tulajdonsegéval, míg a <GridSplitter> ezért azok betölthik az egész cellát. Nezzük meg a 29.23. ábrát. (mivel nem adtunk meg Height vagy minden másnak a label típusokat eltemetni, amely lehetővé teszi, hogy átmértezzük az egyes label típusokat tövönél, hogy a kimenetet, látunk egyötkező választálogot. Ha megnezzük ezt a kimenetet, látunk egyötkező választálogot. Ha megnezzük a csatolt tulajdonaságokat, segítségével állapíthat meg, melyik oszlopbal rendelkezik a többi panelrel, hogy az oszlop, amely támogatja a valasztvonalat, rendelkezik az Auto. Next width tulajdonsegéval, míg a <GridSplitter>

```

</Window>
</Grid>
<Label Name="lblRight" Grid.Column="1" Content="Right!"/>
<!-- Címke hozzáadása az 1. cellához -->
<!-- A valasztvonal meghatározása -->
<GridSplitter Grid.Column="0" Width="5"/>
<!-- Címke hozzáadása a 0. cellához -->
<Label Name="lblLeft" Background="Yellow" Content="Left!"/>
<!-- Címke hozzáadása a 0. cellához -->
<Grid.Column="0" Content="Left!"/>
<!-- A valasztvonal meghatározása -->
<GridSplitter Grid.Column="1" Content="Right!"/>
<!-- Címke hozzáadása az 1. cellához -->
<Label Name="lblRight" Grid.Column="1" Content="Right!"/>
<!-- Címke hozzáadása az 1. cellához -->
<Label Name="lblLeft" Grid.Column="0" Content="Left!"/>

```

```

</StackPanel>
</ScrollViewer>
<StackPanel Content="First" Background="Blue" Height="40"/>
<Button Content="Fourth" Background="Yellow" Height="40"/>
<Button Content="Third" Background="Pink" Height="40"/>
<Button Content="Second" Background="Red" Height="40"/>
<Button Content="First" Background="Green" Height="40"/>
<StackPanel Content="Second">
<StackPanel Content="First" Background="Blue" Height="40"/>
<StackPanel Content="Second" Background="Yellow" Height="40"/>
<StackPanel Content="Third" Background="Pink" Height="40"/>
<StackPanel Content="Fourth" Background="Red" Height="40"/>
<StackPanel Content="Fifth" Background="Green" Height="40"/>
</StackPanel>
</ScrollViewer>

```

Erdemes részletezni, hogy a WPF biztosítja a `<ScrollViewer>` típusú, amely automatikusan lapozási viselkedést kínál a böngészőt paneltípusok számára:

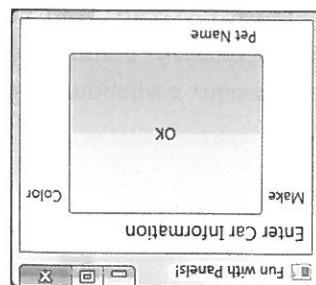
A lapozás engedélyezése a paneltípusoknál

Forráskód A SimpleDockPanel.xaml fájljának forráskódjának másolata a 29. fejezetbenek alkonyvatárát tartalmazza. A forráskódoknnyájtárol lásd a Bevezetés Xvi. oldalat.

A DockPanel típusuk használatanak előnye, hogy ha a felhasználó átmértezi az ablakot, akkor minden elem a panel meágadott oldalához „kapcsol”, marad türe értekeré alattja a lastchillidfül attribútumot. Minden a button típus nem adott meg semmilyen DockPanel, Dock értéket, ezért kijötöti megmaradt helyét. (a DockPanel, Dock minden elem a panel meágadott oldalához „kapcsol” marad az ablakot, akkor minden elem a panel meágadott oldalához „kapcsol”, marad a DockPanel mellett a deklaráció sorrendjében halmozza fel a rendszer.

Megjegyzés Ha több elemet adunk hozzá egy DockPanel ügynökaon oldalához, azokat a meágadott oldal mellett a deklaráció sorrendjében halmozza fel a rendszer.

29.24. ábra: Egy egyszerű DockPanel



```

<Label DockPanel.Dock="Left" Name="lblName"><Label>Pet Name</Label>
<Label DockPanel.Dock="Bottom" Name="lblColor"><Label>Color</Label>
<Label DockPanel.Dock="Right" Name="lblMake"><Label>Make</Label>
<Form DockPanel.Dock="Center" Name="frmCarInfo"><Form>Enter Car Information</Form>
<Button Name="btnOK"><Button>OK</Button>
<Label DockPanel.Dock="Top" Name="lblPetName"><Label>Pet Name</Label>
</DockPanel>

```

Célnak olyan elrendezés kialakításra, ahol a főablak rendelkezik egy felső menürendszerrel, egy eszköztárral, valamint az ablak alján elhelyezett allla- potssával. Az állapotssáv magában foglal egy panelt, amely megjelenít egy alírban.

Az alkalmazás frissített verzióját (amely az új, MySpellCheker nevű Visual Studio 2008 WPF alkalmazás projekt) kibővítiük és véglegesítik a következő oldalakon, tehát pillanatnyilag az alaprendezést és működést fogluk össze-

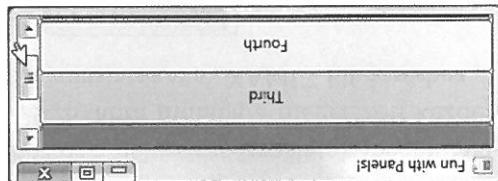
Ablak keretének készítése beágyazott panelek használatával

Ezzel befejezti a WPF főbb paneletpusztának, valamint a tartalmuk pozí- ciójához szabállyozza a vezérlőelem különböző részeit korábban extra helyet. Ezáltal minden résznek a helye közötti kölcsönös elhelyezése garantálható. A vezérlőelemek mindenike két erdekes tulajdonságot (padding és margin)

Aholgy varhafűk, minden panel több tagot biztosít, amelyek lehetővé teszik a tartalom elhelyezésének finomhangolását. Ehhez kapcsolódó megegyezés, hogy a WPF-vezérlőelemek mindenike két erdekes tulajdonságot (padding és margin)

Forráskód A ScrollViewer.xaml fájlja a forrásokonványtár 29. fejezetének alkonyvtára tar- talmazza. A forrásokonványtáról lásd a Bevezetés XIV. oldalát.

29.25. ábra: A ScrollViewer típus használata

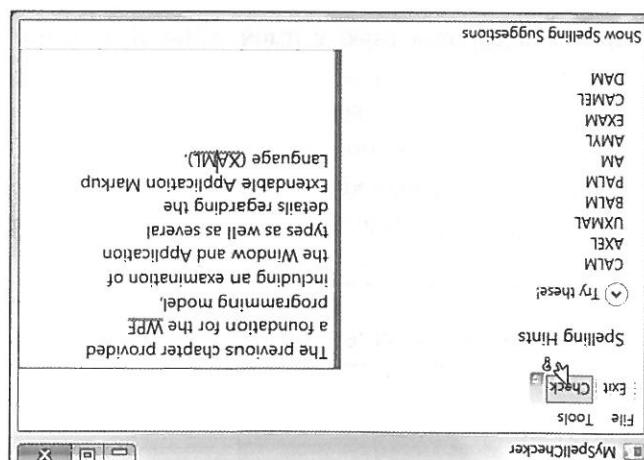


Az előző XAML-definíció eredménye a 29.25. ábrán látható.

Ablak keretének készítése beágyazott panelek használatával

Hagyjuk meg, hogy a két eszközöt arra számíthatunk nem egy várt képet, hanem egy egyszerű szöveges értéket jelelni meg. Noha ez nem lenne elégendő egy termekszámlázásban, képekkel hozzárendelésre az eszközök gombjaihoz általában szintű alkalmazásban, kevésbé gyakorlati igényük, amelynek temakörét a 30. fejezetben foglalkoztatja. Azt is végülik ezre, hogy amikor az egyéb button fölé kerül, a kúzor megváltozik, az alapötthagy megvizsgáláti (tehet most megtezni a szöveges adat). Azt is felhasználhatjuk a részletekkel, hogy a részletekhez mindenféle használóval lehessen elérni.

79.76. Adra: Beágyazott panelek használata egy ablak felházasztály felületének leterhözataloz



Szöveges sort alkot, ha a felhasználó kiválaszt egy menülemezt (vagy) ez-szablonnal, megelőzetes, meglévő írásbeli szövegeket használva. A kiegészítőkkel a felhasználó az adott menüpontban elérhető lehetőségeket látja meg.

A kész menürendszer-delimitcival most a különböző eseménykezelőkkel implementálunk. Először is, adott a File > Exit kezelő, a Fileexit-Click(), amely egyszerűen bezárja az alkalmazást az AppliCation.Current.Shutdown() metódus részén. Az egyes részletek mouseenter és mouseexit eseménykezelői az alábbosakból fogják résztelenül, most azonban csak hétkéket fogunk megeadni. Végül, a Tools > Spelling Hints menülelem ToolAsspelLighthints-Click() kezelője pli- lannabyilag szintén egy shell lesz. Itt van a mögöttes kodralj alkutáls módosítása:

Láthatóink, hogy a mindenrendszerű DokCPanél tetejéről helyeztük. Emellett a <separátor> elemet használtuk arra, hogy beszüíjunk egy végkönny vizsgítésre azonállat a menürendszerbe, közvetlenül az Excel lehetősége elő. Azt is véggyük zására, hogy melyik betű legyen aláhúzott, ha a végfelehasználó lenyomja az agyazott aláhúzás tokenet (például EXIT). Ezт használjuk annak meghatározásra, hogy az egész menütem elemeik header értékei tartalmaznak egy belsejre, vagy azok minden rendszere, hogy minden rendszert a mindenrendszerű DokCPanél tetejéről helyeztük. Emellett a <gyorsbillentyűkötöző> billentyűt (a gyorsbillentyűkötöző).

```
<!-- Menüramdászter elhelyezése az ablak tetéjén -->
<Menu DocPaneL_Dock="Top" >
    <HorizontalGripment="Left" Background="White" >
        <Menutitem Header="Exit" MouseEnter="FileExit_Click" MouseLeave="FileExitArea" >
            <Separator/>
        <Menutitem Header="Tools" MouseEnter="FileExit_Click" MouseLeave="FileExitArea" >
            <Menutitem Header="Help" MouseEnter="FileExit_Click" MouseLeave="FileExitArea" >
                <Menutitem Header="About" MouseEnter="FileExit_Click" MouseLeave="FileExitArea" >
                    <Separator/>
                <Menutitem Header="Help" MouseEnter="FileExit_Click" MouseLeave="FileExitArea" >
            </Menutitem>
        </Menutitem>
    </HorizontalGripment>
</Menu>
```

A WPF-beh a menürendszerreket a Menü típus készítésével, amely Menütetem objektumok gyűjtöttémenyét tarifa karban. Amikor XAML-ben készítünk menürendszert, minden menüpontot a menürendszerrel szemben különöző - többek között kattintási - eseményeket kezeli, amely akkor forrul elő, ha a végfelhasználó rakkattint egy részletekre. A Pelelánkban két félös menülemelet (File és Tools) hozunk létre, amelyek az Exit, illetve a Spelling Hints részlemekeket tartalmazzák. Amellel, hogy kezeljük a kattintási eseményt az egységes módon, hozzá kell követhető kódot a zejelekhez, amelyeket arra használunk, hogy egy későbbi a mousebejár eseményt az egységes módon kezeljük a mousenter eseményt. Ezért, minden menüpont különöző - többek között kattintási - eseményeket szolgáltat, amelyeket a menüpontok az alapvetően szövegeit. Adójuk meg a következőkkel a hatékonyabb belülről:

A menürendszer készítése

```

        </ToolBar>
    Cursor="Help" />
    Click="ToolSspellingHintsClick"
    MouseLeave="MouseLeaveArea"
    MouseEnter="MouseEnterToolShintArea"
    Button Content="Check" MouseEnter="MouseEnterToolShintArea"
    Separator />
    MouseLeave="MouseLeaveArea" Click="FileExitClick"
    Button Content="Exit" MouseEnter="MouseEnterExitArea"
    ToolBar DockPanel.Dock="Top" >
    <!-- Az eszköztár elhelyezése a menü alá -->

```

Következő markupot, kiszervezzük a Menü definíció záró hatókörre után:
 termatiiv módszert biztosítanak egy menülelem alkiválására. Adjuk hozzá a
 Az eszköztárak (amelyeket a WPF-ben a Toolbar típus keppvisel) általában al-

A Toolbar típus készítése

```

    {
    {
    {
    protected void MouseLeaveArea(object sender, RoutedEventArgs args)
    {
    }
    {
    protected void MouseEnterToolShintArea(object sender, RoutedEventArgs args)
    {
    }
    {
    protected void FileExitClick(object sender, RoutedEventArgs args)
    {
    }
    {
    protected void ToolSspellingHintsClick(object sender, RoutedEventArgs args)
    {
    }
    {
    public partial class MainWindow : System.Windows.Window
    {
        public MainWindow()
        {
            InitializeComponent();
        }
        // Az alkalmazás bezárása.
        // Az alkalmazás Current.Shutdown();
        protected void FileExitClick(object sender, RoutedEventArgs args)
        {
    }

```