Eseményvezérelt alkalmazások: 4. gyakorlat

Szavak számlálása (grafikus felülettel)

Készítsünk grafikus felületet a 2. gyakorlaton elkészített, szöveges állományokban szó előfordulás számlálást megvalósító alkalmazáshoz, amelyen megjelenítjük a betöltött fájl szövegét és a szavak előfordulásainak számát!

A Solution Explorerben a solution nevére jobb klikk után válasszuk ki az **Add –> New project...** menüpontot. Válasszuk a **Windows Forms App** sablont! A projektnek adjuk a *VisualizeWordCounter* nevet.

> Ne a *Windows Forms App (.NET Framework)* sablont válasszuk, ugyanis az .NET 5 / .NET Core helyett a korábbi .NET Frameworkre épülő projektet fog létrehozni.

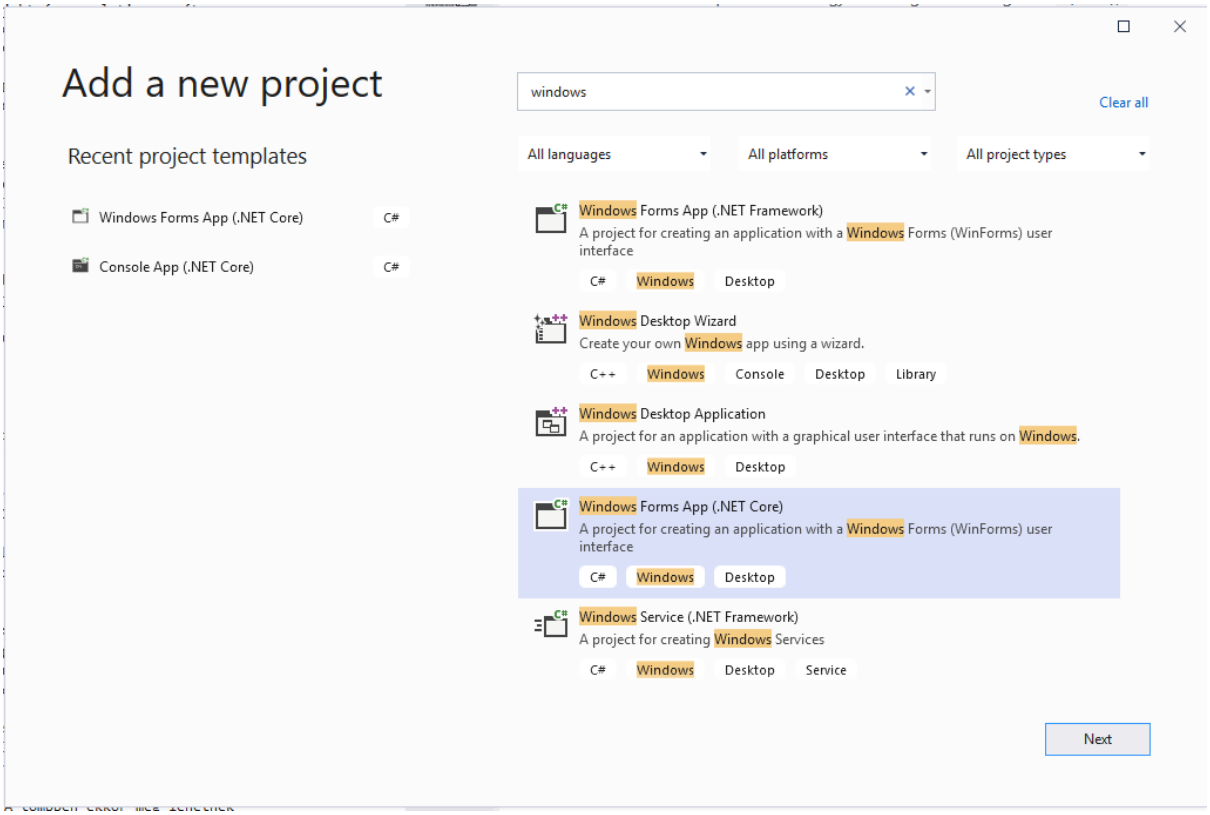

Figure 1: Windows Forms projekt létrehozása

Nevezzük át a Form1.cs fájlt `WordCountDialog`-ra!
A Designerben helyezzük fel a következő vezérlőket az ablakra:

- `MenuStrip`: az ablak tetején, a címsor alatt helyezkedjen el, ez lesz az alkalmazás menüsávja. A vezérlőre jobb klikkelve válasszuk ki az *Edit items...* menüpontot. Az *Add* gombbal adjunk új

menüelemet a menüsávhoz.

- A jobb oldalon látható ablakban be tudjuk állítani a menüelem nevét (`(Name)`), ez legyen `fileMenu`, és szövegét (`Text`), ez legyen `File`. A `fileMenu`-t kijelölve keressük meg a tulajdonságok között a `DropDownItems` elemet, és a `(Collection)`-re kattintva válasszuk ki a mellette megjelenő három pontot. Ezzel egy hasonló menüt kapunk, ahol a `fileMenu`-hoz adhatunk új menüelemeket az *Add* gombbal.
  * Adjunk a `fileMenu`-hoz egy új elemet, ennek neve legyen `openFileDialogMenuItem`, felirata pedig legyen `Open file dialog`.
  * Egy második menüelem neve legyen `countWordsFileDialog`, a felirata pedig legyen `Count words`.

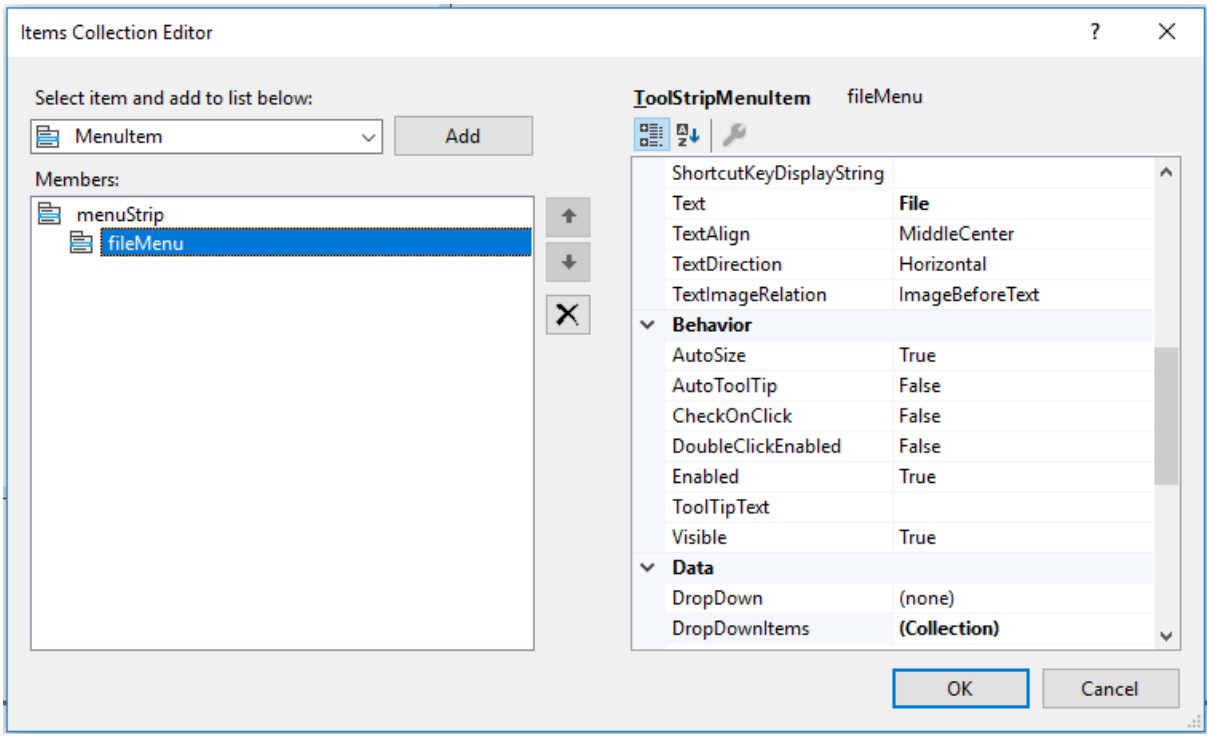

Figure 2: Menüpont hozzáadása a menüsávhoz.

- `TextBox`: ebben fogjuk megjeleníteni a szöveges fájl tartalmát. Helyezzük az ablak bal oldalára, és méretezzük, hogy kb. az ablak felét foglalja el. A Solution Explorer alatt megjelenik a Designerben aktuálisan kijelölt elemhez tartozó tulajdonságok listája (`Properties`), amelyben módosítani tudjuk a vezérlő viselkedését, eseménykezelőket adhatunk az eseményeihez stb. A `TextBox (Name)` tulajdonságát írjuk át `textBox`-ra, majd a `ScrollBars` tulajdonságot állítsuk `Vertical`-ra, hogy görgethető legyen a mező. Állítsuk igazra a `MultiLine` tulajdonságot, hogy a magassága több soros is lehessen, ezután terjeszzük ki az ablak teljes magasságára. A `ReadOnly` tulajdonságot is állítsuk igazra, hogy a felületen keresztül ne legyen szerkeszthető a szöveg.
- `ListBox`: ebben jelenítjük meg a szavakat és előfordulásaik számát. Helyezzük az ablak jobb oldalára, és méretezzük át, hogy kitöltse a maradék helyet. A `Properties` ablakban adjuk neki a `listBoxCounter` nevet.

A teljes ablak kijelölésével láthatjuk a `Properties` ablakban az ablak tulajdonságait. Legyen a fejléc szövege (`Text`) `Word counter`! A `FormBorderStyle` tulajdonságot állítsuk `FixedSingle`-re, a `MaximizeBox` tulajdonságot pedig hamisra, így az ablak fix méretű lesz.

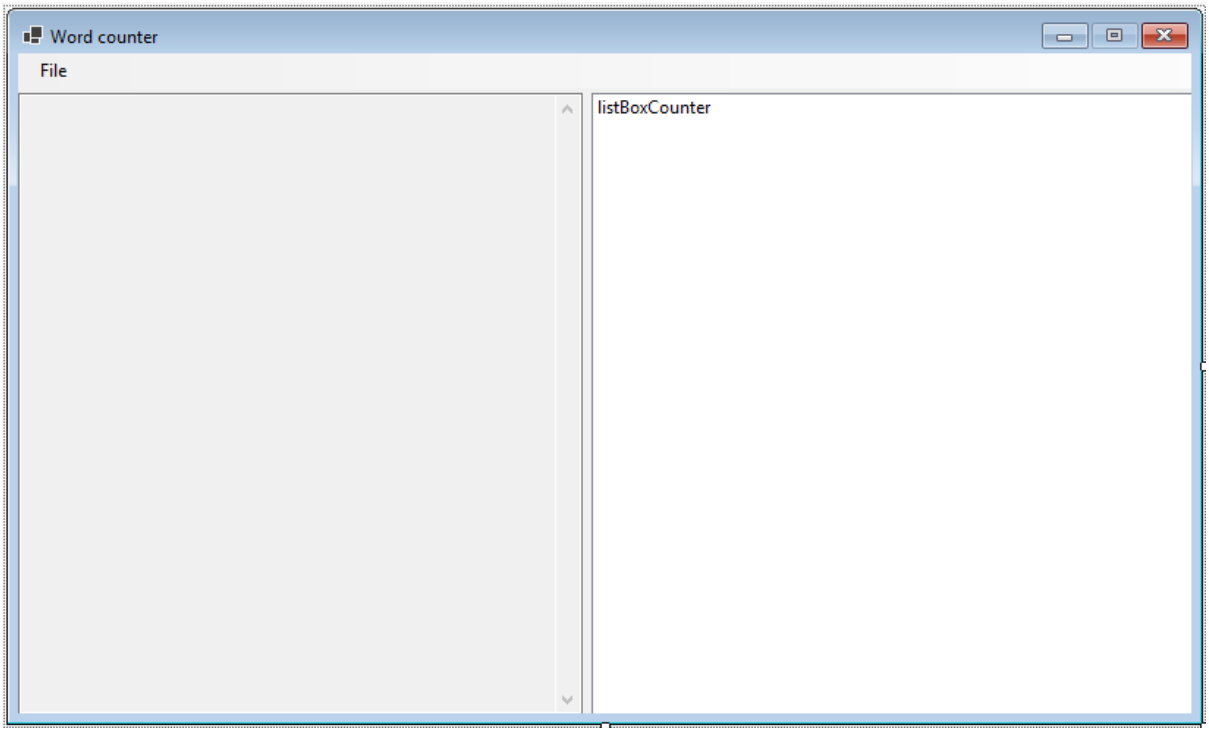

Figure 3: Az alkalmazáshoz tartozó ablak nézete a Designerben.

Mivel fel fogjuk használni az előző feladat `Statistics` osztályát, el kell érnünk a másik projekt névterét. Jobb klikkeljünk a Solution Explorerben a `VisualizeWordCounter` projektre, majd válasszuk ki az *Add –> Project Reference...* menüpontot. Pipáljuk be a `WordCounter` projektet, majd okézzuk le az ablakot.

Nyissuk meg a `WordCountDialog.cs` fájlt (ami semmképpen nem a `WordCountDialog.Designer.cs`)! Ebben a fájlban írjuk meg a menüelemekhez tartozó eseménykezelőket, amelyek a fájlbeolvasást és a szavak számlálását fogják végezni, és kitöltik a `textBox`-ot és a `listBoxCounter`-t. A usingok közé vegyük fel a `WordCounter` névteret!

Vegyünk fel egy privát `Statistics` típusú adattagot (`statistics`), és inicializáljuk a konstruktorban. Definiáljuk az `openFileDialogMenuItem`-hez tartozó eseménykezelőt! Az eseménykezelő egy privát, `void` visszatérési típus metódus, amely a küldő objektumot és egy eseményargumentumot vár paraméterül. A neve legyen `OpenDialog`.

```
private void OpenDialog(object sender, EventArgs e)
```

A fájlt most egy fájltallózó dialóguson keresztül fogjuk megnyitni. Használjuk ehhez az `OpenFileDialog` osztályt. A dialógus kezdeti könyvtárát az `InitialDirectory`, a megengedett fájltípusokat a `Filter` tulajdonságon keresztül állíthatjuk be. Ha azt szeretnénk, hogy minden megnyitáskor a kezdeti könyvtár jelenjen meg, állítsuk a dialógus `RestoreDirectory` tulajdonságát igazra.

A dialógus `ShowDialog()` metódusát meghívva megjelenítjük a fájltallózót. Ha itt sikeresen kiválasztottunk egy fájlt, hívjuk meg a `statistics Load()` metódusát. Paraméterként az `openFileDialog FileName` propertyjét adjuk át. Ne feledjük, hogy itt el kell kapnunk az esetleges kivételt. A `catch` ágban jelenítsünk meg egy `MessageBox`-ot a kivétel szövegével (`Message`), majd térjünk vissza a függvénnyel.

Megspórolhatjuk a szöveg ismételt betöltését abban az esetben, ha ugyanazt a fájlt ugyanazzal a tartalommal próbáljuk betölteni, mint ami már a `textBox`-ban van. Vizsgáljuk meg, hogy a `statistics.FileContent` egyezik-e a `textBox` tartalmával (`Text`), feltéve, hogy előbbi nem üres! Ha a betöltendő szöveg egyezik a korábbival, térjünk vissza.

A `textBox Text` tulajdonságának adjuk értékül a `FilContent` tartalmát, így megjelenik a `textBox`-ban a fájlból beolvasott szöveg. A `listBoxCounter Items` kollekciójának tartalmát töröljük a `Clear()` metódussal.

```
private void OpenDialog(object sender, EventArgs e)
{
   using (OpenFileDialog openFileDialog = new OpenFileDialog())
   {
      openFileDialog.InitialDirectory = "C:\\";
      openFileDialog.Filter = "txt files (*.txt)|*.txt|All files (*.*)|*.*";
      openFileDialog.RestoreDirectory = true;

      if (openFileDialog.ShowDialog() == DialogResult.OK)
      {
         try
         {
            statistics.Load(openFileDialog.FileName);
         }
         catch (System.IO.IOException ex)
         {
            MessageBox.Show("File reading is unsuccessful!\n" + ex.Message,
            "Error", MessageBoxButtons.OK);
            return;
         }

         if (!String.IsNullOrEmpty(statistics.FileContent)
             && statistics.FileContent == textBox.Text)
              return;

         listBoxCounter.Items.Clear();
         textBox.Text = statistics.FileContent;
      }
   }
}
```

Az `OpenDialog`-ot kössük össze a konstruktorban az `openFileDialogMenuItem` `Click` eseményével.

```
openFileDialogMenuItem.Click += OpenDialog;
```

Eseménykezelőt a `-=` operátorral vehetünk le egy eseményről.

Definiáljunk egy új eseménykezelőt `CalculateStatistics` néven! Láthatósága, visszatérési típusa, paraméterei ugyanazok legyenek, mint az előző eseménykezelőnek.

Ha még nem olvastunk be fájlt, azt szeretnénk, ha ez a menüpont feldobna egy üzenetet. Ha a `statistics.FileContent` üres (amit a `String.IsNullOrEmpty()` metódussal tudunk megvizsgálni), dobjunk fel egy `MessageBox`-ot, amely tájékoztat, hogy még nincs beolvasott szöveg, majd térjünk vissza.

Hívjuk meg a `statistics` modell `CountWords()` metódusát, majd az előző feladathoz hasonlóan rendezzük a `WordCount` elemeit.

A `listBoxCounter` vezérlő `BeginUpdate()` metódusát meghívva kezdhetjük a párok kiírását. A rendezett párokon végig iterálva adjuk hozzá az `Items`-hez szövegként a párok kulcsát és értékét, majd az `EndUpdate()` metódussal zárjuk le a `listBoxCounter` szerkesztését.

```
listBoxCounter.BeginUpdate();
foreach (var pair in pairs)
{
    listBoxCounter.Items.Add(pair.Key + ": " + pair.Value);
}
listBoxCounter.EndUpdate();
```

A lista frissítését a `BeginUpdate()` és az `EndUpdate()` keretébe zárása nélkül is elvégezgetjük, ez esetben azonban minden elem hozzáadása a megjelenítés frissítését váltja ki, amely így egy villogó hatást okozhat a felhasználói felületen.

A `CalculateStatistics` eseménykezelőt az előzőhöz hasonlóan kössük rá a `countWordsMenuItem` vezérlő `Click` eseményére.