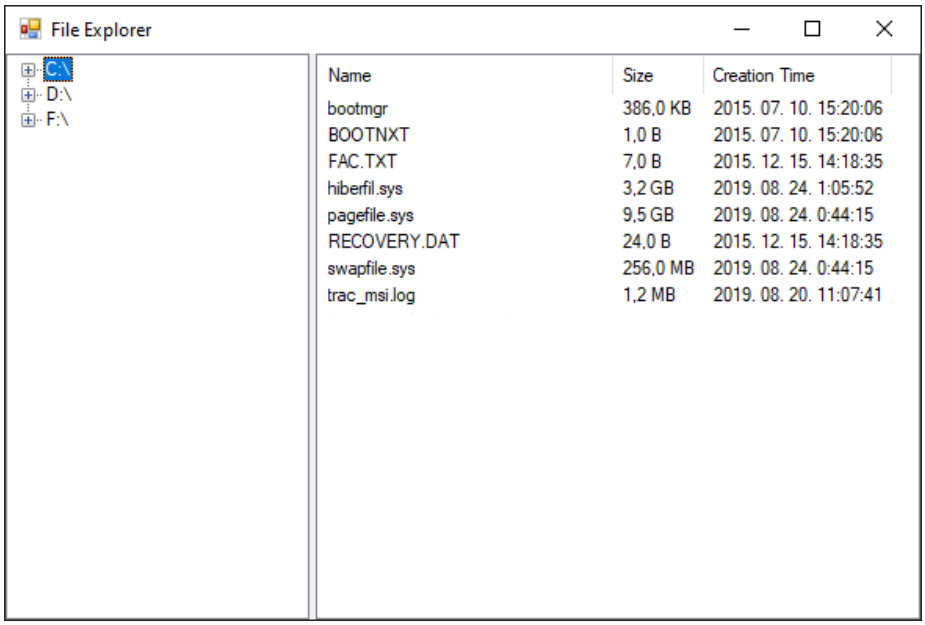
8. gyakorlat

C#/.NET alapú grafikus alkalmazás fejlesztés

Fájlkezelő

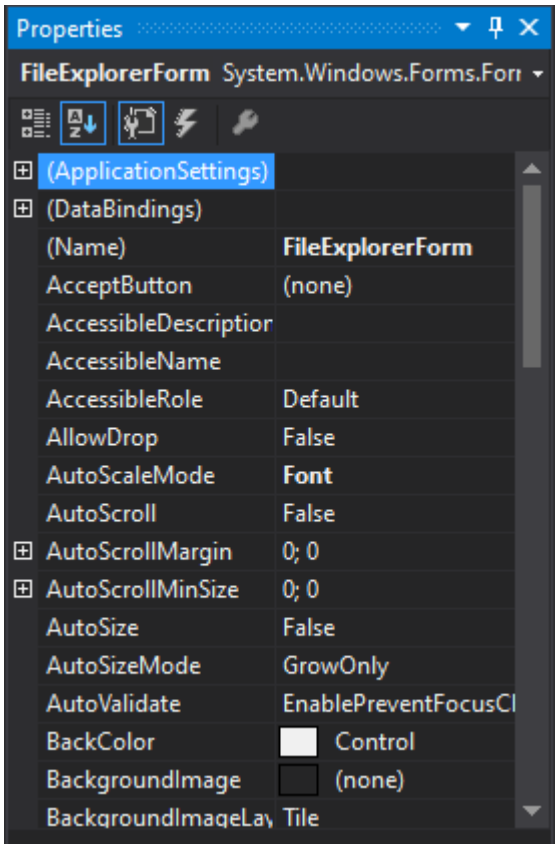
Egy fájlkezelő WinForms alkalmazást fogunk elkészíteni kétrétegű (modell-nézet) architektúrában.

A felület bal oldalán legyen látható az összecsukható könyvtárszerkezet, jobb oldalon pedig a kiválasztott könyvtár fájljai. Egy fájlra duplán kattintva az elindítható (az alapértelmezetten hozzá társított programmal). A könyvtárak tartalmát dinamikusan, lenyitáskor töltsük be, hiszen az egész fájlrendszer betöltése nagyon hosszú időt vehetne igénybe.



Statikus felület

1. Első lépésként hozzuk létre a felületet! Nyissunk meg egy új WinForms projektet, legyen a projekt és a solution neve is `FileExplorer`. A tervezőablakban megjelenő felületet fogjuk most szerkeszteni. Kattintsunk rá az ablakra! Megjelenik a jobb alsó sarokban a tulajdonságszerkesztő, amin néhány tulajdonságot be kell állítanunk.



- Állítsuk be a form `Name` tulajdonságát `FileExplorerForm`-ra, ezzel átnevezzük a tervező által generált osztályt.
- Állítsuk be az ablak címét, ehhez a `Text` tulajdonságot kell szerkesztenünk. Legyen ez most File Explorer.
- Válasszunk egy méretet az ablaknak, a `Size` tulajdonság legyen 600; 400.

2. Ezután rakjunk fel a felületre a bal oldalon található Toolboxról egy `SplitContainert`. Ezzel szétválasztjuk az ablakot.
3. A bal oldalra helyezzünk egy `TreeView`-t, aminek a `Dock` tulajdonságát állítsuk `Fill`-re, így kitöltve a rendelkezésre álló teret.
4. A jobb térfélre helyezzünk egy `ListView`-t, aminek szintén állítsuk be a `Dock` tulajdonságát.
5. Válasszuk ki a `ListView`-t a szerkesztőfelületen, és adjuk meg az oszlopokat. Ehhez a `Columns` tulajdonságra kell kattintani, ahol feljön egy felület. Itt adjuk meg a következő feliratú fejléceket: `Name`, `Size`, `Creation time`.

6. A `ListView View` tulajdonságát állítsuk `Details`-re, hogy listanézetet kapjunk.

7. A felületre felvett elemek neveit érdemes megváltoztatni, ezt szintén a tulajdonságszerkesztőben tehetjük meg. A fájl nevét úgy tudjuk megváltoztatni, hogy ráállunk a solution explorerben a `Form1.cs` fájlra, majd a properties ablakban átírjuk a nevét, legyen ez például `FileExplorerForm.cs`.

Így egy futtatható programot kaptunk. Ha a Start gombra kattintunk, felugrik a tervezett felület.

Meghajtók listázása

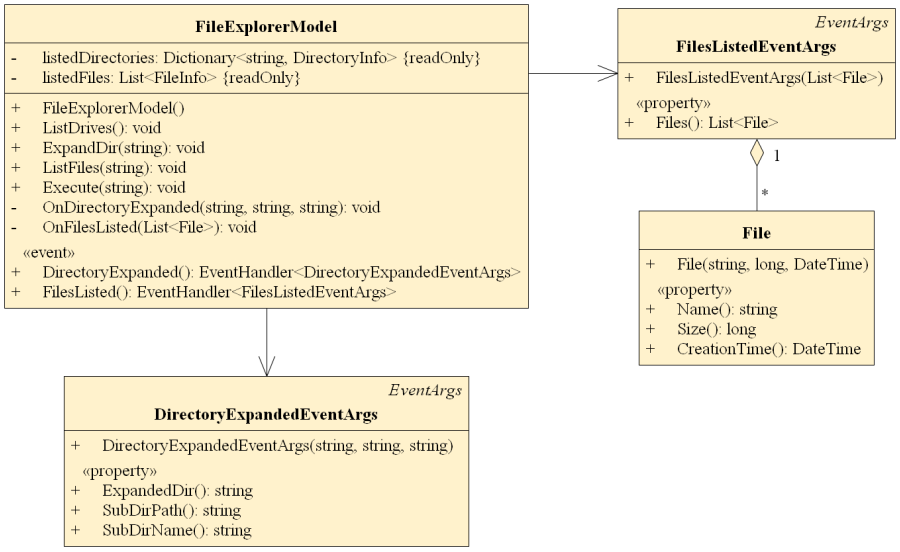


Figure 1: Modell osztálydiagramja

Első funkcióként írjuk ki a meghajtókat az alkalmazás felületén bal oldalt megjelenő `TreeView`-ban! Szeretnénk, ha a kiírás már a program elindulásakor megtörténne, így az ablak `Load` eseményére fogunk feliratkozni, ami az ablak betöltődésekor váltódik ki. Ha a felülettervezőn az ablak tulajdonságszerkesztőjét tekintjük (*Properties* ablak), a villám jelre kattintva előjönnek a kiváltott események.

- Keressük meg a `Load` eseményt, és kattintsunk rá duplán. Legenerálódik egy eseménykezelő, ahova beírhatjuk azt a kódrészletet, ami lefut az esemény hatására. Ebben a kontextusban minden felületre felvitt elem elérhető, azzal a névvel, amit megadtunk nekik. (A `Load` esemény tehát a konstruktor után kerül kiváltásra.)

- A `TreeView` objektumok hierarchiát reprezentálnak, és a csomópontok

kulcs-érték párokat tartalmaznak. A kulcs alapján kereshetők, és az érték fog megjelenni a felületen. A meghajtók lekérdezése már nem a megjelenítéshez fog tartozni, hiszen ez a logika nem nézetfüggő. Vezessünk be egy modell osztályt (`FileExplorerModel`), egy `ListDrives` metódussal!

- A fájlrendszer könyvtárainak kezeléséhez a `DirectoryInfo` típust fogjuk használni (a `System.IO` névtérben található), amin keresztül lekérhetők a megjelenítendő infók (név, méret, módosítás dátuma). A modellben reprezentáljuk a már kilistázott könyvtárakat egy `Dictionary`-vel, a neve legyen `listedDirectories`. A kulcs a könyvtár abszolút útvonala lesz, az érték pedig a könyvtárhoz tartozó `DirectoryInfo` példány.
- Készítsük el a `ListDrives` metódust! A cél az, hogy eltároljuk a meghajtók gyökeréhez tartozó `DirectoryInfo`-t a dictionarybe, majd jelezzük a nézetnek, hogy megváltozott a modell állapota. A meghajtók lekérdezéséhez a `DriveInfo` osztály `GetDrives` metódusa használható. El kell tárolni az összes meghajtó gyökerét, a gyökér elérési útvonala szerint (`FullName` tulajdonság).

```
DriveInfo[] drives = DriveInfo.GetDrives();
foreach (DriveInfo drive in drives)
{
    string rootPath = drive.RootDirectory.FullName;
    listedDirectories.Add(rootPath, drive.RootDirectory);
    // TODO: esemény kiváltása, amellyel a modell jelzi az állapota megváltozását
}
```

- Hozzunk létre egy eseményt (`DirectoryExpanded`), amivel jelezzük a nézet számára, hogy kilistáztuk egy könyvtár egy alkönyvtárát (most speciálisan a rootét, amit jelöljünk `/`-rel). Készítsük el a hozzá tartozó eseményargumentum osztályt (`DirectoryExpandedEventArgs`), ami a szülő könyvtár útvonalát, a kilistázott könyvtár útvonalát és nevét tartalmazza. Az eseményt az `OnDirectoryExpanded` metóduson keresztül hívjuk meg!

```
private void OnDirectoryExpanded(string expandedDir, string subDirPath,
                                 string subDirName)
{
    if (DirectoryExpanded != null)
    {
        DirectoryExpanded(this,
            new DirectoryExpandedEventArgs(expandedDir, subDirPath, subDirName));
    }
}
```

- A nézetben iratkozzunk fel erre az eseményre, és hozzunk létre egy eseménykezelőt a következő szignatúrával:

```
private void DirectoryExpanded(object sender, DirectoryExpandedEventArgs e)
```

- Az eseménykezelőben keressük meg a `TreeView`-ban a kiterjesztett csomópontot, a kulcs alapján. Mikor kezdetben a meghajtókat adjuk hozzá, nem fogunk találni a `Node`-ok között semmit, ebben az esetben a `TreeView`-hoz közvetlenül kell hozzáadni egy újat. Ellenkező esetben a talált csomópont gyerekei közé vegyük fel az eseményben kapott könyvtárakat.

```
private void DirectoryExpanded(object sender, DirectoryExpandedEventArgs e)
{
    TreeNode[] expandedNode = dirTreeView.Nodes.Find(e.ExpandedDir, true);

    if (expandedNode.Length == 0) // meghajtót adunk hozzá
    {
        // ...
    }
    else  // könyvtárat adunk hozzá
    {
        // ...
    }
}
```

- Egy új könyvtár `TreeView`-hoz adása után annak érdekében, hogy megjelenjen a `+` ikon a könyvtárak mellett (jelölve ezzel, hogy az kinyitható), bele kell tennünk egy helyfoglaló (*placeholder*) elemet, helyettesítve az eredeti tartalmat. Ezt úgy tehetjük meg, hogy az adott `TreeNode` objektum `Nodes` kollekciójához adunk egy új elemet az `Add` művelettel. Ez a helyfoglaló bármi, akár egy üres karakterlánc is lehet, ugyanis a könyvtár megnyitásakor majd eltávolítjuk.

Könyvtárak kiterjesztése

- A `TreeView` vezérlő `BeforeExpand` eseményéhez kell eseménykezelőt írnunk. Figyeljünk arra, hogy kitöröljük a *placeholder* elemet, majd csak ezután határozzuk meg a kinyitott könyvtár gyerekeit a modellben. A nézet felé a `DirectoryExpanded` eseményen keresztül kommunikálhat a modell.
- Figyeljünk arra, hogy miközben elérünk egy gyerek könyvtárat, ahhoz nem biztos, hogy hozzáférhetünk. Ez egy `UnauthorizedAccessException` kivételt fog kiváltani, amit a nézetben lekezelhetünk egy dialógusablak feldobásával.

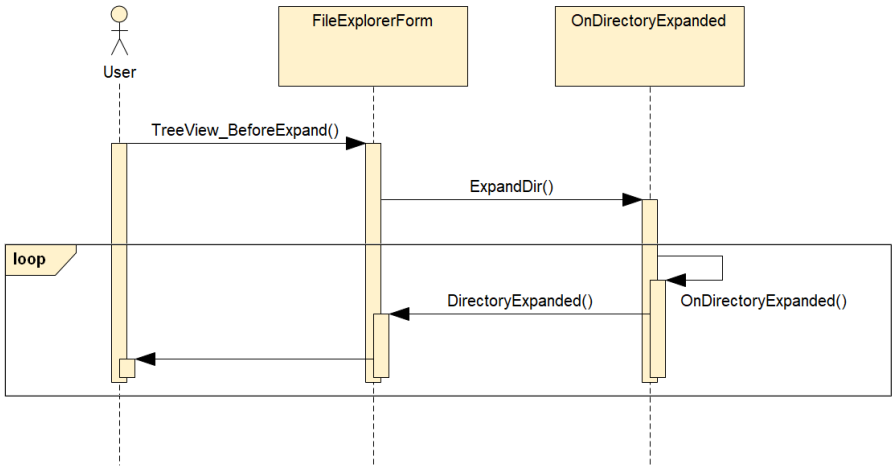


Figure 2: Nézet és modell közötti interakció szekvenciadiagramon a `TreeView` egy elemének kinyitásakor

Fájlok megjelenítése

- A modellben a definiáljunk egy `ListFiles` metódust. A `TreeView` vezérlő `AfterSelect` eseményének kiváltásakor hívjuk meg a modell ezen metódusát.
- A `ListFiles` metódus paraméterként vegye át a könyvtár útvonalát, amelyből a fájlokat meg szeretnénk jeleníteni. A modellben tárolt `listedDirectories` dictionary-ben meg kell keresnünk a kiválasztott könyvtárhoz tartozó `DirectoryInfo` objektumot, majd le kell kérdezni a tartalmazott fájlokat a `GetFiles()` eljárással (`FileInfo` objektumok tömbjével reprezentálja).
- A modell tartsa nyilván az éppen megjelenített fájlok listáját egy `FileInfo` objektumok listájában, a neve legyen `listedFiles`.
- Hozzunk létre egy eseményt, amin keresztül az összes kilistázott fájlt átadhatjuk a nézetnek (`FilesListed`). Készítsük el a hozzá tartozó eseményargumentum osztályt (`FilesListedEventArgs`), ami tartalmazza a könyvtárban található fájlok nevét, méretét és létrehozási idejét.
- A nézet adja hozzá a kapott fájlokat a `ListView`-hoz, `ListViewItem`-ekben. Ezután a szöveg méretéhez igazítjuk az oszlopokat:

```
this.fileListView.AutoResizeColumns(ColumnHeaderAutoResizeStyle.ColumnContent);
```

- A fájlok mérete alapértelmezetten byte-ban van megadva, de jobb lenne, ha automatikusan konvertálva B/KB/MB/GB-ra jelenítenénk meg. Készítsünk egy olyan metódust a nézetben, amivel az átváltást elvégezhetjük.

Kapjon paraméterként egy `long`-ot (bájtokban vett érték), és adjon vissza egy kiírandó stringet. (Ötlet: a bájtban vett méretet osztogassuk vissza 1024-gyel!)

Fájl megnyitása

- A `ListView` vezérlő `DoubleClick` eseményére kell feliratkozni.
- Egy nem végrehajtható fájlt parancssorban végrehajtva a Windows operációs rendszer az alapértelmezetten hozzárendelt alkalmazással futtattja. Készítsünk egy `ProcessStartInfo` objektumot, amelynek konstruktora várja a fájl elérési útvonalát, a `UseShellExecute` tulajdonságának igazra állításával pedig beállíthatjuk, hogy parancssoron keresztül hajtsa végre a fájlt. A `Process.Start` statikus metódussal futtathatjuk az így előkészített objektumot.

```
ProcessStartInfo startInfo = new ProcessStartInfo(path)
startInfo.UseShellExecute = true;
Process.Start(startInfo);
```