Eötvös Loránd Tudományegyetem
Informatikai Kar

Eseményvezérelt alkalmazások

---

11. előadás

MAUI alkalmazások perzisztenciája

Cserép Máté
mcserep@inf.elte.hu
https://mcserep.web.elte.hu

MAUI alkalmazások perzisztenciája

Alkalmazások környezete

- Az alkalmazások egy biztonságos környezetben futnak
  - nem férhetnek hozzá más alkalmazások adataihoz
  - csak korlátozott módon férhetnek hozzá a rendszer adataihoz (pl. fájlrendszer), és azt is csak engedéllyel
  - szintén engedéllyel használhatják csak az eszközöket (*application manifest*)

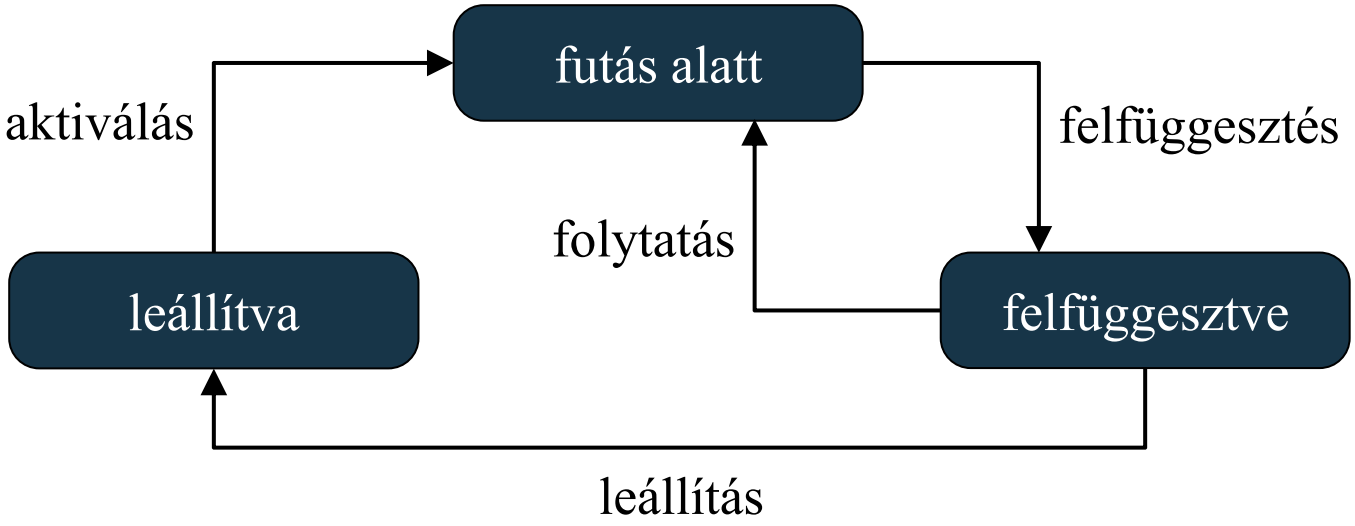
- Mindegyik alkalmazás számára rendelkezésre áll
  - egy beállítás/tulajdonság tároló, amelyben kulcs/érték párokat helyezhet el (a kulcs *string*)
  - egy lokális könyvtár, amely csak az alkalmazás fájljait tárolja

- Alkalmazás törlésekor a hozzá tartozó adatok is törlődnek

MAUI alkalmazások perzisztenciája

Alkalmazások életciklusa

- A mobil alkalmazások más életciklusban futnak, mint az asztali alkalmazások
  - a *futás alatt (running)* és a *terminált (not running)* állapotok mellett megjelenik a *felfüggesztett* (*suspended*) állapot is, amely akkor lép életbe, ha az alkalmazás a háttérbe (vagy a gép) alvó állapotba) kerül, célja a takarékosság



MAUI alkalmazások perzisztenciája

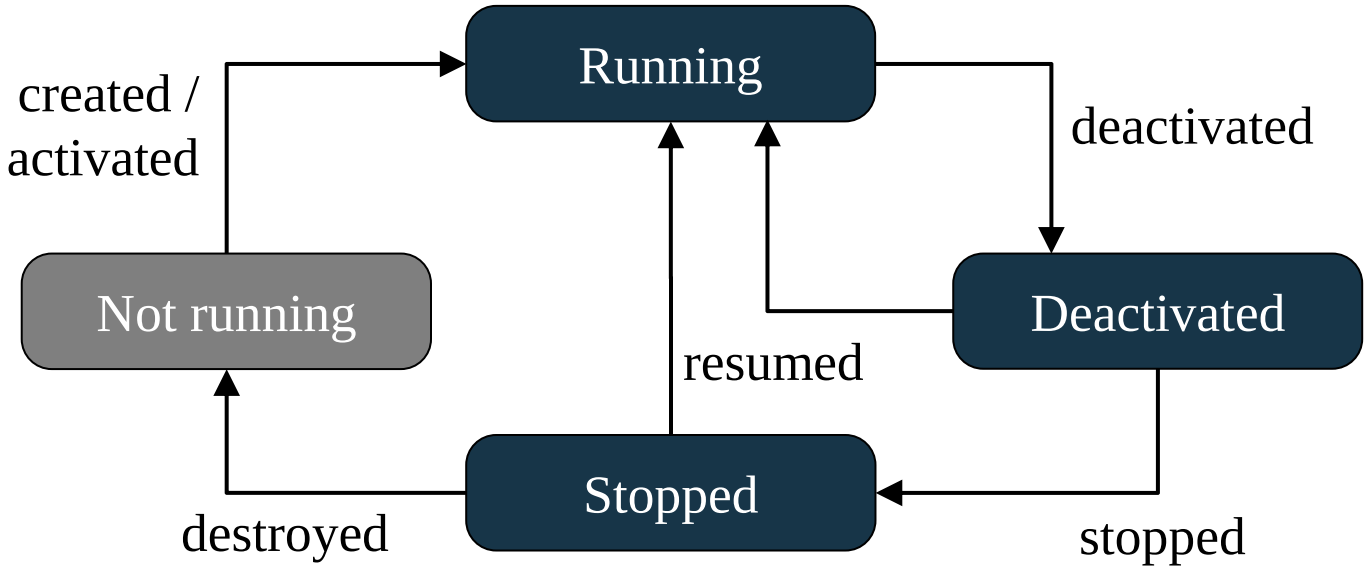
Alkalmazások életciklusa

- A felfüggesztés célja az erőforrásokkal való takarékoskodás
  - a fejlesztőnek törekednie kell rá, hogy felfüggesztett állapotban az alkalmazás minél kevesebb erőforrást igényeljen
  - a futó tevékenységeket célszerű leállítani, az adatokat perzisztálni
- A rendszer úgy is dönthet (pl. ha kevés a memória), hogy a felfüggesztett alkalmazást leállítja, majd újraindítja, ha a felhasználó visszaváltott rá
  - célszerű, hogy a felhasználó ennek ellenére olyan állapotban kapja vissza az alkalmazást, amelyben hagyta, így ezt az állapotot vissza kell állítanunk
  - az állapot eltárolását felfüggesztéskor kell elvégeznünk

MAUI alkalmazások perzisztenciája

Alkalmazások életciklusa

- A MAUI alkalmazások egységes életciklus kezeléssel rendelkeznek:
  - az **`App`** alkalmazás osztályunk **`CreateWindow`** metódusát felüldefiniálva, eseménykezelőkkel adható meg az életciklus váltáskor végrehajtandó tevékenység (**`Created`**, **`Activated`**, **`Deactivated`**, **`Stopped`**, **`Resumed`**, **`Destroying`**).



MAUI alkalmazások perzisztenciája

Alkalmazások életciklusa

Esemény	Leírás
`Created`	Az alkalmazás elindításakor váltódik ki, az ablak még nem feltétlenül került megjelenítésre.
`Activated`	Az alkalmazás fókuszba kerülésekor (vagy közvetlenül előtte) váltódik ki.
`Deactivated`	Az alkalmazás fókuszvesztésekor váltódik ki. (Egyes platformokon még részlegesen látható lehet.)
`Stopped`	Az alkalmazás felfüggesztésekor váltódik ki.
`Resumed`	A felfüggesztett alkalmazás állapotból visszatéréskor kerül kiváltásra.
`Destroying`	Az alkalmazás leállításakor váltódik ki.

MAUI alkalmazások perzisztenciája

Alkalmazások életciklusa

- Kompatibilitási megfontolásokból használhatjuk a *Xamarin Forms* életciklus kezelését is
  - ekkor az alkalmazás (**`App`**) tartalmaz metódusokat indításkor (**`OnStart`**), felfüggesztéskor (**`OnSleep`**) és folytatáskor (**`OnResume`**) futtatandó tevékenységek végrehajtására
- Emellett az egyes platformokon külön is kezelhetjük az életciklust
- A perzisztálás történhet
  - az alkalmazás beállításai/tulajdonságai közé, amelyet a rendszer automatikusan tárol (**`Preferences`**)
  - a fájlrendszerbe, vagy adatbázisba (pl. SQLite)

MAUI alkalmazások perzisztenciája

Alkalmazások életciklusa

- Pl.:

```
public class App : Application {
    protected override Window CreateWindow(
        IActivationState activationState)
{
      Window window =
        base.CreateWindow(activationState);

      window.Stopped += (s, e) => {
        // felfüggesztés
        Preferences.Set("state", _data);
        // állapot elmentése a tulajdonság közé
        _data = null; // memória kiürítése
      };

      …
```

MAUI alkalmazások perzisztenciája

Alkalmazások életciklusa

```
            …

        window.Resumed += (s, e) => {
          // folytatás
          if (Preferences.ContainsKey("state"))
              _data = Preferences.Get("state", 0);
          // ha van elmentett állapot, visszatöltjük
          // alapértelmezett értéket meg kell adni
        };
    }
}
```

MAUI alkalmazások perzisztenciája

Példa

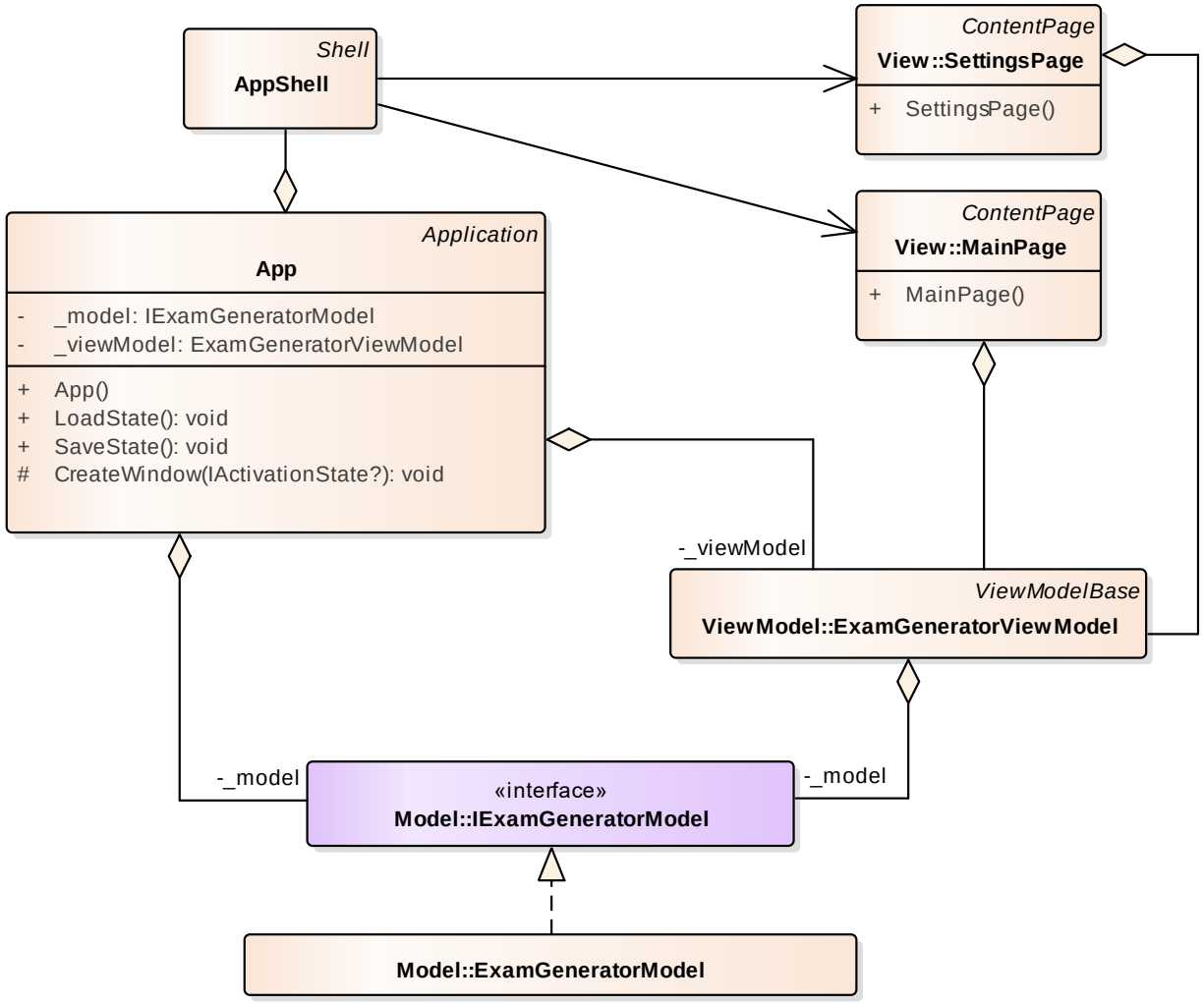
Feladat: Készítsünk egy vizsgatétel generáló alkalmazást, amely ügyel arra, hogy a vizsgázók közül ketten ne kapják ugyanazt a tételt.

- kiegészítjük életciklus kezeléssel az alkalmazást, eltároljuk a modell állapotát, valamint a generálás állapotát az alkalmazás beállításai (**`Preferences`**) közé
- mivel nincs külön perzisztencia, közvetlenül a modelltől kérjük el az információkat, és tároljuk el egyenként
- az akalmazás életciklusát az **`Application`** vezérli, ezért itt definiálunk egy **`SaveState`** és egy **`LoadState`** eljárást
- indításkor, illetve folytatáskor a korábbi állapotot betöltjük, és ennek megfelelően inicializáljuk a modellt
- a hatékony erőforrás gazdálkodás érdekében felfüggesztéskor felszabadítjuk a modellt és nézetmodellt is

MAUI alkalmazások perzisztenciája

Példa

Tervezés:



MAUI alkalmazások perzisztenciája

Példa

Megvalósítás (`App.xaml.cs`*):*

```
Public void SaveState() {
    Preferences.Set("questionCount",
        _model.QuestionCount);
        // elmentjük a tételek számát
    Preferences.Set("periodCount",
        _model.PeriodCount);

    …
    _model = null;
    _viewModel = null;
    (MainPage as AppShell)!BindingContext = null;
    // a szemétgyűjtő kitörli a memóriából a
    // modellt és a nézetmodellt
}
```

MAUI alkalmazások perzisztenciája

Példa

Megvalósítás (`App.xaml.cs`*):*

```
public void LoadState() {
    // ha el lett mentve az állapot, akkor
    // visszatöltjük
    if (Preferences.ContainsKey("questionCount") {
        _model = new ExamGeneratorModel(
            Convert.ToInt32(
              Preferences.Get("questionCount", 10)),
            Convert.ToInt32(
              Preferences.Get("periodCount", 0)),;
        // az elmentett adatokkal példányosítjuk a
        // modellt
    }
    …
}
```

MAUI alkalmazások perzisztenciája

Platformfüggetlen perzisztencia

- A MAUI megkötésekkel, de egységes, platformfüggetlen megoldást kínál a fájlrendszerben történő perzisztenciára
  - minden platformon rendelkezésünkre áll egy könyvtár, amelyet csak az alkalmazásunk érhet el (rendszeradminisztrátor szintű fájlrendszer hozzáférést leszámítva)
  - az egységes fájlrendszer kezelést a **`Microsoft.Maui.Storage.FileSystem`** osztály biztosítja
    - az **`AppDataDirectory`** az alkalmazás adatkönyvtára, amely mentésre és szinkronizálásra is kerülhet eszközök között
    - a **`CacheDirectory`** az alkalmazás gyorsítótár könyvtára, itt ideiglenes állományokat célszerű elhelyezni

MAUI alkalmazások perzisztenciája

Platformfüggetlen perzisztencia

- A **`System.IO`** névtérben korábban már megismert osztályok és eljárások a fájlkezelés összes lehetőségét biztosítják
  - könyvtárak (**`Directory`**) elérését, listázását (**`GetFiles`**, **`GetDirectories`**), benne könyvtárak létrehozását (**`CreateDirectory`**), fájlok és könyvtárak törlését (**`Delete`**)
  - fájlok (**`File`**) létrehozását, megnyitását (**`Open`**), olvasást (**`ReadAllBytes`**, **`ReadAllLines`**, **`ReadAllText`**), írást (**`WriteAllBytes`**, …), másolást (**`Copy`**), …
  - az írás és olvasás történhet adatfolyamok segítségével is (**`StreamReader`**, **`StreamWriter`**), amelyeket a megszokott módon használhatunk

MAUI alkalmazások perzisztenciája

Példa

Feladat: Készítsünk egy Tic-Tac-Toe programot, amelyben két játékos küzdhet egymás ellen.

- elkésztjük a nézet MAUI megvalósítását, amelyben képekkel (**`Image`**) jelenítjük meg a mező tartalmát egy rácsban (**`Grid`**), a képeket erőforrásként töltjük be (**`circle.png`**, **`cross.png`**, **`empty.png`**)
- megvalósítjuk a platformfüggetlen perzisztenciát az alkalmazás adatkönyvtárába (**`FileSystem.AppDataDirectory`**)
- a mentést a főprogram hajtja végre egy közös fájlba (így mindig csak az utolsó állapotot tudjuk menteni), valamint felfüggesztés esetén is mentjük az aktuális játékállapotot

MAUI alkalmazások perzisztenciája

Példa

Megvalósítás (`App.cs`*):*

```
protected override Window CreateWindow(…)
{
    // …
    window.Stopped += async (s, e) => {
        // elmentjük a jelenleg folyó játékot
        try
        {
            await _model.SaveGameAsync(Path.Combine(
                FileSystem.AppDataDirectory,
                "SuspendedGame"));
        }
        catch { }
    };
}
```

MAUI alkalmazások perzisztenciája

Platformfüggő perzisztencia

- Speciálisabb esetekben a megvalósítás már platformonként jelentősen eltérhet
  - a perzisztencia platformonként történő megvalósítását a MAUI projekt megfelelő platform könyvtárában helyezzük el, így csak az adott platformra kerül lefordításra
  - a megvalósítás betöltését függőségi befecskendezéssel végezzük
  - a hasonló platformfüggő függőségek kezelésének megkönnyítésére egy egységes módszer biztosított (**`DependencyService`**)
    - a függőségek megvalósítását megjelölhetjük (**`Dependency`** attribútum), így a rendszer automatikusan regisztrálja és betölti őket
    - a függőségeket az interfészen keresztül betölthetjük (**`DependencyService.Get`**)

MAUI alkalmazások perzisztenciája

Platformfüggő perzisztencia

- pl.:

```
// Android platformkód

[assembly: Dependency(typeof(AndroidPersistence))]
   // megjelöljük a függőséget
namespace App.Android
{
   public AndroidPersistence : IPersistence
   {
      public void SaveData(String data)
      {
         … // perzisztencia megvalósítása
      }
      …
```

MAUI alkalmazások perzisztenciája

Platformfüggő perzisztencia

- pl.:

```
// MAUI cross-platform kód
public interface IPersistence { … }
   // perzisztencia interfésze


…
IPersistence persistence =
   DependencyService.Get<IPersistence>();
   // megtalálja a perzisztencia megvalósítását
   // az adott platformon

IModel model = new Model(persistence);
   // így most már befecskendezhető
…
```

MAUI alkalmazások perzisztenciája

Platformfüggő perzisztencia

- Függőségi befecskendezés helyett használhatunk *parciális metódusokat* (*partial methods*) a platformfüggő perzisztencia implementálására
  - parciális metódusokat parciális osztályokban használhatunk
  - az egyik fájlban csak deklarálájuk a parciális metódust, egy másikban pedig megvalósítjuk
  - platformonként eltérő implementációt adhatunk meg, és ezeket a platformok saját könyvtárában elhelyezve mindig pontosan egy megvalósítás kerül fordításra

MAUI alkalmazások perzisztenciája

Platformfüggő perzisztencia

- Szüksége esetén platformfüggő működés preprocesszor direktívákkal is megadható, pl.:

```
#if ANDROID
    // …
#elif IOS
    // …
#elif WINDOWS
    // …
#endif
```

- A preprocesszor direktívákkal történő platformspecifikus kódrészletek gyorsan nehezen átláthatóvá és karbantarthatóvá tehetik alkalmazásunk forráskódját.

MAUI alkalmazások perzisztenciája

Platformfüggő perzisztencia

- pl. (**Persistence.cs**):

```
// MAUI cross-platform kód
public class Persistence {
    public partial void SaveData(String data);
    // perzisztencia deklarációja
}
```

- pl. (**Platforms/Android/Persistence.cs**):

```
// Android platformspecifikus kód
public class Persistence
{
    public partial void SaveData(String data)
    {
      … // perzisztencia megvalósítása
    }
```

MAUI alkalmazások perzisztenciája

Példa

Feladat: Készítsünk egy Tic-Tac-Toe programot, amelyben két játékos küzdhet egymás ellen.

- lehetőséget adunk a felhasználónak a betöltött/mentett fájl kiválasztására két új oldal segítségével (**`LoadGamePage`**, **`SaveGamePage`**), amit az **`AppShell`**-ből végzett navigáció (**`Navigation`**) segítségével kapcsolunk a főoldalhoz
- a nézetek mellett meg kell valósítanunk a fájlböngészés
  - perziszenciáját (**`Store`**), amely beolvassa az adatkönyvtár (**`FileSystem.AppDataDirectory`**) tartalmát
  - modelljét (**`StoredGameBrowserModel`**), amely kezeli a játékok adatait
  - nézetmodelljét (**`StoredGameBrowserViewModel`**), amely biztosítja a frissítést és a parancsok végrehajtását

MAUI alkalmazások perzisztenciája

Jogosultságkezelés

- Az alkalmazások tulajdonságait, képességeit az *alkalmazás leíró* (*application manifest*) segítségével írhatjuk le
  - Android esetén az **`AndroidManifest.xml`**, Windows esetén a **`Package.appxmanifest`**, iOS esetén az **`Info.plist`** állomány tartalmazza az alkalmazás által kért jogosultságokat
- A jogosultságokat a felhasználó vagy az operációs rendszer visszavonhatja, ezért az alkalmazás futtatásakor is szükséges ellenőrizni meglétüket
  - ehhez a MAUI platformfüggetlen **`Permissions`** osztálya nyújt támogatást (**`CheckStatusAsync`**, **`RequestAsync`**)
  - pl. pozíció használatához való hozzáférés kérése:

```
PermissionStatus status = await
Permissions.RequestAsync<Permissions.LocationWhenInUse>();
```

MAUI alkalmazások perzisztenciája

Példa

Feladat: Készítsünk egy Tic-Tac-Toe programot, amelyben két játékos küzdhet egymás ellen.

- a lehetőséget adunk a felhasználónak a betöltött/mentett fájl kiválasztására a Dokumentumok könyvtárból
- ennek megadása platformfüggő módon lehetséges
  - Windows: **`Environment.GetFolderPath(Environment.SpecialFolder.MyDocuments)`**
  - Android: **`Android.OS.Environment.GetExternalStoragePublicDirectory(Android.OS.Environment.DirectoryDocuments)`**
  - iOS: **`Environment.GetFolderPath(Environment.SpecialFolder.MyDocuments)`**
- Androidon a kezeléshez szükséges jogosultságot igényeljük meg

MAUI alkalmazások perzisztenciája

Viselkedések

- Lehetőségünk van a grafikus felület működését deklaratív módon bővíteni viselkedések (**`Behavior`**) segítségével
  - a viselkedés egy olyan típus, amely adott vezérlőhöz biztosít hozzáférést, és adott típusú vezérlőhöz csatlakoztatható, további tevékenységeket
    - pl. tulajdonságok megváltoztatása, állapot ellenőrzése
  - megadhatunk vezérlő csatlakoztatásakor (**`OnAttachedTo`**) és lecsatlakoztatásakor (**`OnDetachingFrom`**) végrehajtható tevékenységet
    - általában egy eseménykezelőt társítunk, amely egészen a lecsatlakoztatásig végrehajtódik
    - a metódusokban mindig meg kell hívnunk az ős metódusát

MAUI alkalmazások perzisztenciája

Viselkedések

- pl.:

```
public class NumericValidationBehavior :
    Behavior<Entry>
    // egy viselkedés beviteli mezőkre
{
    protected override void OnAttachedTo(
        Entry entry)
    {
        entry.TextChanged += OnEntryTextChanged;
        // eseménykezelő hozzárendelése szöveg
        // megváltoztatásakor
        base.OnAttachedTo(entry);
    }
    … // az eseménykezelő validálja a tartalmat
```

MAUI alkalmazások perzisztenciája

Viselkedések

- a viselkedések XAML kódban hasznosíthatóak a vezérlőkre példányosítva (**`Behaviors`**)
  - így amennyiben a megfelelő viselkedés rendelkezésre áll, tetszőleges módon bővíthető deklaratívan a működés
- pl.:

```
<Entry Placeholder="Enter number">
    <Entry.Behaviors>
        <local:NumericValidationBehavior />
        <!-- az ellenőrzés automatikusan lefut a
             szöveg megváltoztatásakor -->
    </Entry.Behaviors>
</Entry>
```

MAUI alkalmazások perzisztenciája

Példa

Feladat: Készítsünk egy egyszerű számológépet, amellyel a négy alapműveletet végezhetjük el, illetve láthatjuk korábbi műveleteinket is.

- valósítsuk meg a tájoláskezelést viselkedések segítségével
- a tájolás MAUI-ban platformfüggetlen módon kérhető le (**`DeviceDisplay.MainDisplayInfo.Orientation`**)
- két viselkedést veszünk fel
  - egyik tetszőleges vezérlő elhelyezését befolyásolja (**`ViewOrientationBehavior`**)
  - a másik a **`StackLayout`** elem elrendezési módját befolyásolja (**`StackLayoutOrientationBehavior`**)

MAUI alkalmazások perzisztenciája

Példa

Megvalósítás (`StackLayoutOrientationBehavior.cs`*):*

```
private void Bindable_SizeChanged(object sender,
    EventArgs e)
{
    // az eszköz tájolásának függvényében
    // változtatunk a StackLayout tájolásán
    switch (DeviceDisplay
            .MainDisplayInfo.Orientation) {
       case DisplayOrientation.Landscape:
          if (stackLayout.Orientation !=
              StackOrientation.Horizontal)
             stackLayout.Orientation =
                StackOrientation.Horizontal;
             break;
            ...
```