

## Eseményvezérelt alkalmazások fejlesztése II

### 11. előadás

## Platformspecifikus Xamarin alkalmazások

Giachetta Roberto

roberto@inf.elte.hu  
http://people.inf.elte.hu/roberto

## Platformspecifikus Xamarin alkalmazások

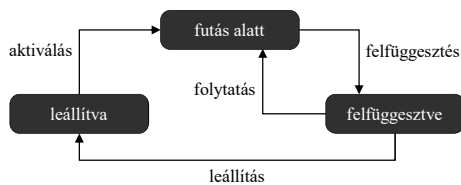
### Alkalmazások környezete

- 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*)
- Mind egyik alkalmazás számára rendelkezésre áll
  - egy beállítás/tulajdonság tároló, amelyben kulcs/érték párokat helyezhet el (a kulcs szöveg, )
  - egy lokális könyvtár, amely csak az alkalmazás fájljait tárolja
- Alkalmazás törlésekor a hozzá tartozó adatok is törlődnek

## Platformspecifikus Xamarin alkalmazások

### 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 takarékoság



## Platformspecifikus Xamarin alkalmazások

### 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

## Platformspecifikus Xamarin alkalmazások

### Alkalmazások életciklusa

- A Xamarin Forms alkalmazások egységes életciklus kezeléssel rendelkeznek
  - 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 (**Application.Properties**)
    - a fájlrendszerbe, vagy adatbázisba (pl. SQLite)

## Platformspecifikus Xamarin alkalmazások

### Alkalmazások életciklusa

- Pl.:

```
public class App : Application {
    protected override void OnSleep() {
        // felfüggesztés
        Properties["state"] = _data;
        // állapot elmentése a tulajdonság közé
        _data = null; // memória kiürítése
    }
    protected override void OnResume() {
        // folytatás
        if (Properties.ContainsKey("state"))
            _data = Properties["state"];
        // ha van elmentett állapot, visszatöltjük
    }
    ...
}
```

## Platformspecifikus Xamarin alkalmazások

### 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 (**Properties**) közé
- mivel nincs külön perzisztencia, közvetlenül a modelltől kérjük el az információkat, és tároljuk el egyenként
- 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

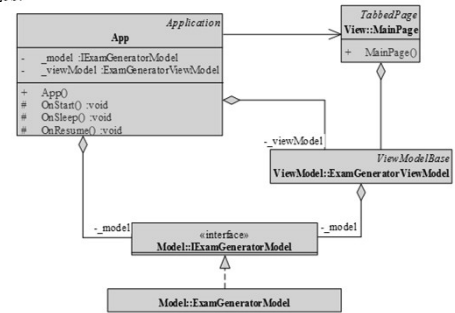
ELTE IK, Eseményvezérelt alkalmazások fejlesztése II

11:7

## Platformspecifikus Xamarin alkalmazások

### Példa

*Tervezés:*



ELTE IK, Eseményvezérelt alkalmazások fejlesztése II

11:8

## Platformspecifikus Xamarin alkalmazások

### Példa

*Megvalósítás (App.cs):*

```

protected override void OnSleep() {
    Properties["questionCount"] =
        _model.QuestionCount;
    // elmentjük a tételek számát
    Properties["periodCount"] = _model.PeriodCount;

    ...

    _model = null;
    _viewModel = null;
    MainPage.BindingContext = null;
    // a szemetgyűjtő kitörli a memóriából a
    // modellt és a nézetmodellt
}

```

ELTE IK, Eseményvezérelt alkalmazások fejlesztése II

11:9

## Platformspecifikus Xamarin alkalmazások

### Példa

*Megvalósítás (App.cs):*

```

protected override void OnResume() {
    // ha el lett mentve az állapot, akkor
    // visszatöltjük
    if (Properties.ContainsKey("questionCount"))
    {
        _model = new ExamGeneratorModel(
            (Int32)Properties["questionCount"],
            (Int32)Properties["periodCount"]);
        // az elmentett adatokkal példányosítjuk a
        // modellt
    }
    ...
}

```

ELTE IK, Eseményvezérelt alkalmazások fejlesztése II

11:10

## Platformspecifikus Xamarin alkalmazások

### Perzisztencia

- A Xamarin Forms nem biztosít egységes megoldást a perzisztenciára, mivel a megvalósítás platformonként jelentősen eltérhet
- a perzisztencia platformonként történő megvalósítását a függőség befecskendezés teszi lehetővé
- 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**)

ELTE IK, Eseményvezérelt alkalmazások fejlesztése II

11:11

## Platformspecifikus Xamarin alkalmazások

### Perzisztencia

- pl.:
 

```

// Android platformkód

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

```

ELTE IK, Eseményvezérelt alkalmazások fejlesztése II

11:12

Platformspecifikus Xamarin alkalmazások	
Perzisztencia	
<ul style="list-style-type: none"> <li>pl.:  <pre>// Xamarin Forms kód public interface IPersistence { ... }     // perzisztencia interfésze  ... IPersistence persistence =     DependencyService.Get&lt;IPersistence&gt;();     // megtalálja a perzisztencia megvalósítását     // az adott platformon  IModel model = new Model(persistence);     // így most már befejezhető ... </pre> </li> </ul>	
ELTE IK, Eseményvezérelt alkalmazások fejlesztése II	11:13

Platformspecifikus Xamarin alkalmazások	
Perzisztencia Android platformon	
<ul style="list-style-type: none"> <li>Az Android (és iOS) platform(ok) a fájlkezelés összes lehetőségét biztosítják</li> <li>könyvtárak (<b>Directory</b>) elérését, listázását (<b>GetFiles</b>, <b>GetDirectories</b>), benne könyvtárak létrehozását (<b>CreateDirectory</b>), fájlok és könyvtárak törlését (<b>Delete</b>)</li> <li>fájlok (<b>File</b>) létrehozását, megnyitását (<b>Open</b>), olvasást (<b>ReadAllBytes</b>, <b>ReadAllLines</b>, <b>ReadAllText</b>), írást (<b>WriteAllBytes</b>, ...), másolást (<b>Copy</b>), ...</li> <li>az írás és olvasás történhet adatfolyamok segítségével is (<b>StreamReader</b>, <b>StreamWriter</b>), amelyeket a megszokott módon használhatunk</li> </ul>	
ELTE IK, Eseményvezérelt alkalmazások fejlesztése II	11:14

Platformspecifikus Xamarin alkalmazások	
Perzisztencia Android platformon	
<ul style="list-style-type: none"> <li>Android platformon a perzisztálás történhet: <ul style="list-style-type: none"> <li>az alkalmazás helyi könyvtárába (<b>Environment.SpecialFolder.ApplicationData</b>)</li> <li>a felhasználó személyes könyvtárába (<b>SpecialFolder.Personal</b>), vagy tematikus könyvtárba (<b>SpecialFolder.MyMusic</b>, <b>SpecialFolder.MyPictures</b>)</li> <li>az összes felhasználó által elérhető alkalmazás könyvtárba (<b>SpecialFolder.LocalApplicationData</b>)</li> </ul> </li> <li>Az utak kezelését a <b>Path</b> osztály, speciális könyvtárak elérési útját az <b>Environment.GetFolderPath</b> metódus biztosítja</li> </ul>	
ELTE IK, Eseményvezérelt alkalmazások fejlesztése II	11:15

Platformspecifikus Xamarin alkalmazások	
Perzisztencia Android platformon	
<ul style="list-style-type: none"> <li>pl.:  <pre>String documentsPath =     Environment.GetFolderPath         (Environment.SpecialFolder.Personal);     // felhasználó könyvtárának lekérése  String filePath =     Path.Combine(documentsPath, filename);     // fájl elérési útvonalának létrehozása  File.WriteAllText(filePath, text);     // szöveg kiírása a fájlba </pre> </li> </ul>	
ELTE IK, Eseményvezérelt alkalmazások fejlesztése II	11:16

Platformspecifikus Xamarin alkalmazások	
Perzisztencia Windows platformon	
<ul style="list-style-type: none"> <li>Windows platformon az alkalmazás fájljait a megadott könyvtáron keresztül (<b>StorageFolder</b>) kezelhetjük</li> <li>listázhatunk (<b>GetFilesAsync</b>) létrehozhatunk (<b>CreateFileAsync</b>), betölthetünk (<b>OpenAsync</b>), törölhetünk (<b>DeleteAsync</b>) fájlokat, illetve könyvtárakat</li> <li>a könyvtárakban fájlok (<b>StorageFile</b>) helyezkednek el, amelyeknek elérhetjük a tulajdonságait (<b>Name</b>, <b>FileType</b>, <b>Properties</b>, ...)</li> <li>a fájlokat a <b>FileIO</b> osztályon keresztül írhatjuk/olvashatjuk <ul style="list-style-type: none"> <li>szöveges, illetve bináris tartalmat tudunk kezelni (<b>ReadTextAsync()</b>, <b>ReadLinesAsync()</b>, <b>ReadBufferAsync(...)</b>, <b>WriteTextAsync(...)</b>)</li> </ul> </li> </ul>	
ELTE IK, Eseményvezérelt alkalmazások fejlesztése II	11:17

Platformspecifikus Xamarin alkalmazások	
Perzisztencia Windows platformon	
<ul style="list-style-type: none"> <li>a megszokott adatfolyam kezelés is elérhető (<b>StreamReader</b>, <b>StreamWriter</b>), azonban funkcionalitása korlátozott</li> <li>Windows platform a perzisztálás történhet <ul style="list-style-type: none"> <li>lokálisan, a helyi fájlrendszerben (<b>ApplicationData.Current.LocalFolder</b>)</li> <li>központilag (roaming), a felhasználói fiókhoz társítva (<b>ApplicationData.Current.RoamingFolder</b>), amelyek a felhasználó valamennyi eszközén elérhetőek</li> <li>átmenetileg, futás közben elérhető tárhelyen (<b>ApplicationData.Current.TemporaryFolder</b>)</li> </ul> </li> </ul>	
ELTE IK, Eseményvezérelt alkalmazások fejlesztése II	11:18

## Platformspecifikus Xamarin alkalmazások

### Perzisztencia Windows platformon

```

• pl.:
StorageFolder folder =
    ApplicationData.Current.LocalFolder;
    // alkalmazás helyi könyvtárának lekérézése

StorageFile file =
    await Folder.GetFilesAsync("data.txt");
    // fájl (aszinkron) betöltése

IList<String> lines =
    await FileIO.ReadLinesAsync(file);
    // összes sor kiolvasása a fájlból
    
```

ELTE IK, Eseményvezérelt alkalmazások fejlesztése II

11:19

## Platformspecifikus Xamarin alkalmazások

### Példa

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

- elkészítjük a nézet Xamarin Forms megvalósítását, amelyben képekkel (**Image**) jelenítjük meg a mező tartalmát egy rácsban (**FlowLayoutView**), a képeket csatoljuk a platformprojektekhez (**circle.png**, **cross.png**, **empty.png**)
- megvalósítjuk a perzisztenciát Android (**AndroidFilePersistence**) és Windows (**WindowsFilePersistence**) platformokra, amelyeket függőségként kezelünk
- a mentést a főprogram hajtja végre egy közös fájlba (így mindig csak az utolsó állapotot tudjuk menteni), valamint felfüggesztés esetén is mentjük az aktuális játékalapot

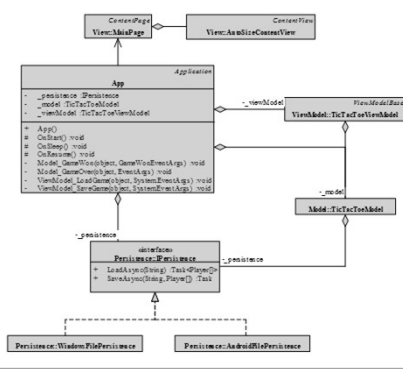
ELTE IK, Eseményvezérelt alkalmazások fejlesztése II

11:20

## Platformspecifikus Xamarin alkalmazások

### Példa

*Tervezés:*



ELTE IK, Eseményvezérelt alkalmazások fejlesztése II

11:21

## Platformspecifikus Xamarin alkalmazások

### Példa

*Megvalósítás (App.cs):*

```

protected override void OnSleep()
{
    // elmentjük a jelenleg folyó játékot
    try
    {
        Task.Run(() => await
            _model.SaveGameAsync("SuspendedGame"));
        // az aszinkron tevékenységet taszkban
        // futtatjuk
    }
    catch { }
}
    
```

ELTE IK, Eseményvezérelt alkalmazások fejlesztése II

11:22

## Platformspecifikus Xamarin alkalmazások

### Példa

*Megvalósítás (AndroidFilePersistence.cs):*

```

public async Task<Player[]> LoadAsync(String path)
{
    // a betöltés a személyes könyvtárból történik
    String filePath =
        Path.Combine(Environment.GetFolderPath(
            Environment.SpecialFolder.Personal), path);

    // a fájlműveletet taszk segítségével végezzük
    String text = await Task.Run(() =>
        File.ReadAllText(filePath));
    return text.Split().Select(number =>
        (Player)Int32.Parse(number)).ToArray();
}
    
```

ELTE IK, Eseményvezérelt alkalmazások fejlesztése II

11:23

## Platformspecifikus Xamarin alkalmazások

### 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**), amint navigáció (**NavigationPage**) segítségével kapcsolunk a főoldalhoz
- a nézetek mellett meg kell valósítanunk a fájlböngészés
  - perzisztenciáját (**IStore**), amely beolvassa a könyvtárak tartalmát (platformonként eltérő módon)
  - modelljét (**StoredGameBrowserModel**), amely kezeli a játékok adatait
  - nézetmodelljét (**StoredGameBrowserViewModel**), amely biztosítja a frissítést és a parancsok végrehajtását

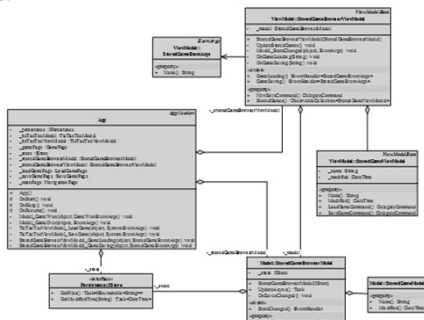
ELTE IK, Eseményvezérelt alkalmazások fejlesztése II

11:24

## Platformspecifikus Xamarin alkalmazások

### Példa

Tervezés:



ELTE IK, Eseményvezérelt alkalmazások fejlesztése II

11:25

## Platformspecifikus Xamarin alkalmazások

### 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 végrehajtható
  - a metódusokban mindig meg kell hívunk az ős metódusát

ELTE IK, Eseményvezérelt alkalmazások fejlesztése II

11:26

## Platformspecifikus Xamarin alkalmazások

### 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
            }
```

ELTE IK, Eseményvezérelt alkalmazások fejlesztése II

11:27

## Platformspecifikus Xamarin alkalmazások

### Viselkedések

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

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

ELTE IK, Eseményvezérelt alkalmazások fejlesztése II

11:28

## Platformspecifikus Xamarin alkalmazások

### 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 platformonként eltérő módon kérhető le, ezért egy interfészen keresztül (**IDeviceProperties**) fűtünk hozzá a platformfüggő megvalósításhoz, mint függőséghez (**DependencyService**)
- két viselkedést veszünk fel
  - egyik tetszőleges vezérlő elhelyezését befolyásolja (**ViewOrientationBehavior**)
  - a másik a **StackLayout** elem elrendezési módját befolyásolja (**StackLayoutOrientationBehavior**)

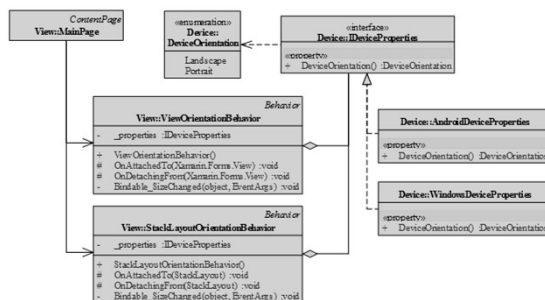
ELTE IK, Eseményvezérelt alkalmazások fejlesztése II

11:29

## Platformspecifikus Xamarin alkalmazások

### Példa

Tervezés:



ELTE IK, Eseményvezérelt alkalmazások fejlesztése II

11:30

## Platformspecifikus Xamarin alkalmazások

### Példa

```
Megvalósítás (StackLayoutOrientationBehavior.cs):
private void Bindable_SizeChanged(object sender,
    EventArgs e)
{
    // az eszköz tájolásának függvényében
    // változtatunk a StackLayout tájolásán
    switch (_properties.DeviceOrientation) {
        case DeviceOrientation.Landscape:
            if (stackLayout.Orientation !=
                StackOrientation.Horizontal)
                stackLayout.Orientation =
                    StackOrientation.Horizontal;
            break;
            ...
    }
```

ELTE IK, Eseményvezérelt alkalmazások fejlesztése II

11:31

## Platformspecifikus Xamarin alkalmazások

### Lokalizáció

- Általános elvárás, hogy az alkalmazások a felhasználó nyelvén kommunikáljanak, és annak megfelelő nyelvi környezetet használják (pl. számformátum, pénznem, ...), ezt nevezzük az *alkalmazás lokalizációjának (application localization)*
- A .NET alkalmazások a lokalizációt egységes formában kezelik (a **System.Globalization** névtérben)
  - az alkalmazás nyelvi környezete (**CultureInfo**) bárhol elérhető, módosítható, illetve az alkalmazás átveheti a rendszer nyelvi környezetét (**CultureInfo.CurrentCulture**)
  - a megjelenített, nyelvfüggő tartalmakat (szöveg, képek) lehetőségünk van lokalizált erőforrásokból (*resource file*) betölteni

ELTE IK, Eseményvezérelt alkalmazások fejlesztése II

11:32

## Platformspecifikus Xamarin alkalmazások

### Lokalizáció

- A lokalizációt két lépésben határozhatjuk meg, nyelv, illetve terület (ország szintjén)
  - pl.: angol (Egyesült Államok): **en-US**, angol (Egyesült Királyság): **en-GB**, magyar (Magyaország): **hu-HU**
  - az adott lokalizációhoz kérhetjük le az erőforrásokat (**CultureInfo**), pl.:  
`CultureInfo info = new CultureInfo("en-US");`  
`// amerikai lokalizáció betöltése`
  - így lehetőségünk van elérni a lokális információkat, pl.: naptár (**Calendar**), időformátum (**DateTimeFormat**), számformátum (**NumberFormat**), tizedes elválasztó (**NumberFormat.NumberDecimalSeparator**)

ELTE IK, Eseményvezérelt alkalmazások fejlesztése II

11:33

## Platformspecifikus Xamarin alkalmazások

### Lokalizált erőforrások

- Az alkalmazás erőforrásait elhelyezhetjük erőforrás fájlokban (.resx), majd közvetlenül, statikus tulajdonságokként hivatkozhatjuk meg őket
  - az erőforrás fájl neve az osztálynév, a kulcs a tulajdonság neve lesz, pl.:  
`nameLabel.Text = AppText.NameText;`  
`// az AppText.resx fájl NameText kulcsú`  
`// szövegének betöltése, és a címkére állítása`
  - az erőforrásfájlt több nyelvre is elkészíthetjük úgy, hogy a lokalizáció nevét hozzáillesztjük a fájlnevhez, pl.:  
`AppText.en-US.resx`  
`// szövegek amerikai nyelvi környezetre`  
`AppText.hu-HU.resx // magyar környezetre`

ELTE IK, Eseményvezérelt alkalmazások fejlesztése II

11:34

## Platformspecifikus Xamarin alkalmazások

### Lokalizált erőforrások

- a rendszer a beállított nyelvi környezetnek megfelelő erőforrásfájl tartalmát tölti be
- amennyiben a lokalizációnak megfelelő fájl nem található, vagy nincs benne megadott kulcsú elem, akkor a lokalizáció nélkül megadott erőforrásfájl tartalmát tölti be
- így célszerű minden esetben egy alapértelmezett erőforrás fájlt is készíteni, pl.:  
`AppText.resx // alapértelmezett szövegek`  
`AppText.hu-HU.resx`  
`// lokalizáció magyar környezetre`
- az erőforrások kezelése további lehetőségeket a **ResourceManager** típus biztosít

ELTE IK, Eseményvezérelt alkalmazások fejlesztése II

11:35

## Platformspecifikus Xamarin alkalmazások

### Lokalizált erőforrások a nézetben

- Amennyiben a lokalizációt nem kódban, hanem a grafikus felületen (XAML-ben) szeretnénk betölteni, ki kell bővítenünk a Xamarin funkcionalitását
  - a felületre történő beírásához szükséges egy bővítés a leíró nyelvhez (**MarkupExtension**), amelyben megadjuk a lokalizált tartalom betöltésének módját (**ProvideValue**)
  - Android és iOS platformokon ezen felül a beépített nyelvi környezetet meg kell feleltetnünk .NET nyelvi környezetnek, így azt platformfüggő módon kell lekérnünk
    - pl. ki kell cserélnünk az összekötőt (`_` helyett `-`), egyes környezetek helyett mászt kell használnunk (`gsw-CH` helyett `de-CH`)

ELTE IK, Eseményvezérelt alkalmazások fejlesztése II

11:36

## Platformspecifikus Xamarin alkalmazások

### Példa

*Feladat:* Készítsünk egy egyszerű számológépet, amellyel a négy alapszámítást végeztethetjük el, illetve láthatjuk korábbi műveleteinket is.

- jelenítsük meg a szövegeket és a tizedes elválasztót a nyelvi beállításnak megfelelően angolul, vagy magyarul
- a tizedes elválasztót a nézetmodell közvetlenül elkérheti a környezettől
- a szövegeket erőforrásban (**ApplicationText**) tároljuk, amelyből készítünk alapértelmezett és magyar változatot
- kiegészítjük az eszköztulajdonságokat (**IDeviceProperties**) a nyelvi környezet kezelésével **Android** platformra
- megvalósítunk egy XAML bővítést, ami szintén elvégzi a konverziót (**TranslateExtension**)

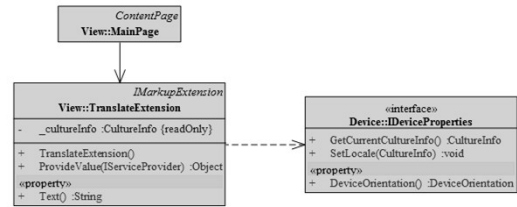
ELTE IK, Eseményvezérelt alkalmazások fejlesztése II

11:37

## Platformspecifikus Xamarin alkalmazások

### Példa

*Tervezés:*



ELTE IK, Eseményvezérelt alkalmazások fejlesztése II

11:38

## Platformspecifikus Xamarin alkalmazások

### Példa

*Megvalósítás (MainPage.xaml):*

```
<ContentPage ...
  xmlns:view="clr-namespace:ELTE.Calculator.View"
  ...>
...
<Label Text="{view:Translate HistoryListTitle}"
  Style="{StaticResource
    NumberListViewLabelStyle}" />
<!-- nyelvfüggő szöveg betöltése a
  TranslateExtension típus segítségével -->
...
```

ELTE IK, Eseményvezérelt alkalmazások fejlesztése II

11:39