

Dr. Hajnal Éva: Haladó
programozás

HALADÓ PROGRAMOZÁS

Reflexion
Attributumok

Reflexió

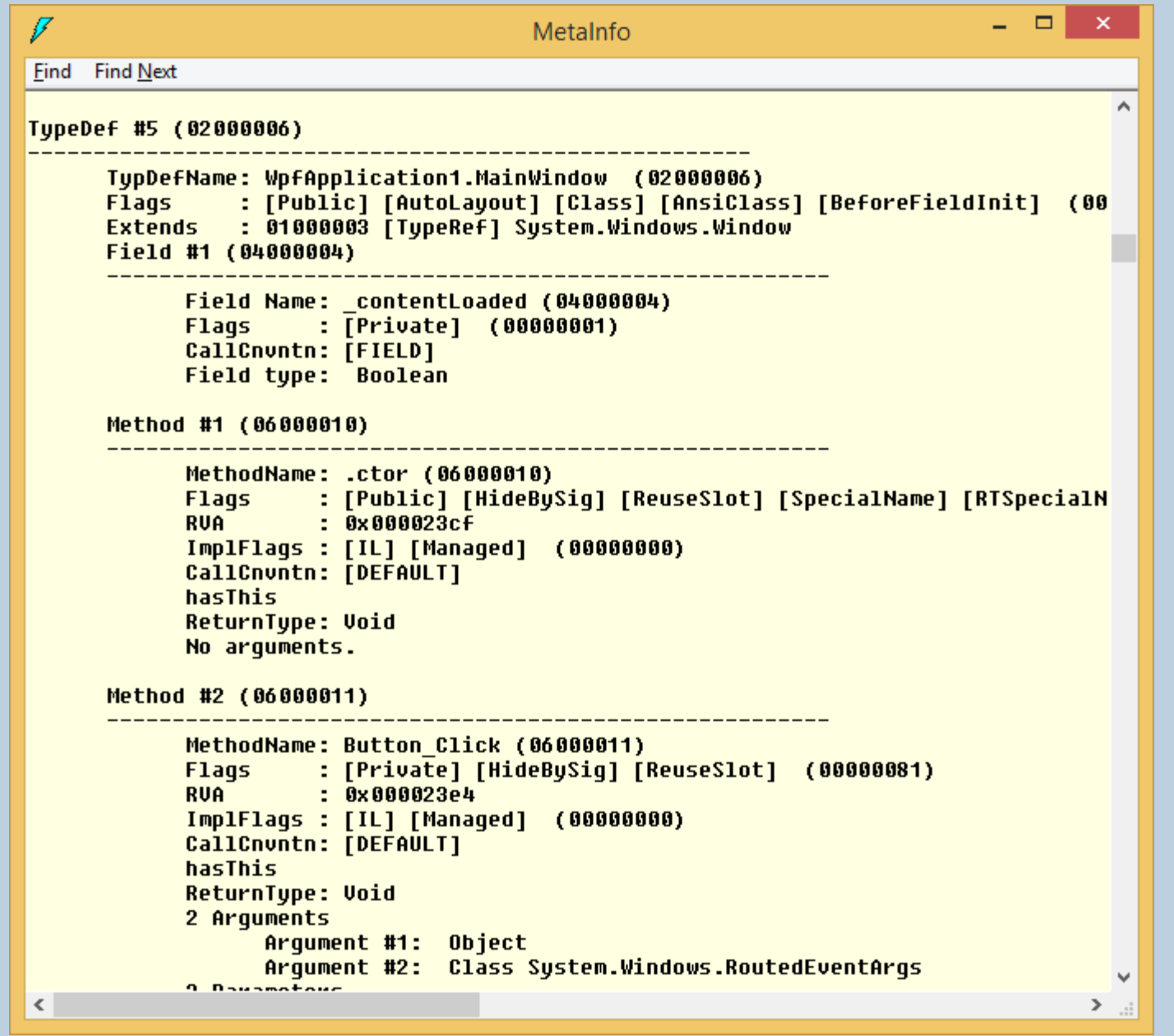
- A reflexió az a képesség, amellyel a program önmaga struktúráját és viselkedését futásidőben analizálni és alakítani tudja
 - Magas szintű nyelv kell hozzá (Java, PHP, ... C#)
 - Különböző mértékű támogatás a nyelvekben
- C#: System.Reflection névtér
- C#-ban a leggyakoribb használati módja a futásidőben végrehajtott típusanalízis
 - De lehetséges típusok futásidejű létrehozása is: System.Reflection.Emit

Háttérinformáció

- A típusok futásidejű vizsgálatát a fordító által a szerelvényben elhelyezett leíró információ (metaadat) teszi lehetővé
 - Szerelvény = assembly: .exe, .dll (egyszerűsítés)
 - Szerelvény metaadatai: függőségek, benne található típusok stb.
 - Típus metaadatai: általa implementált interfészek, őszosztályok; tagjai stb.
 - Tag metaadatai: láthatóság; metódusok paraméterei; tulajdonságokat alkotó metódusok stb.

Metaadat

- Visual Studio Command Prompt / Ildasm.exe, Ctrl+M



The screenshot shows a window titled "MetalInfo" with a search bar at the top containing "Find" and "Find Next". The main content area displays the following metadata:

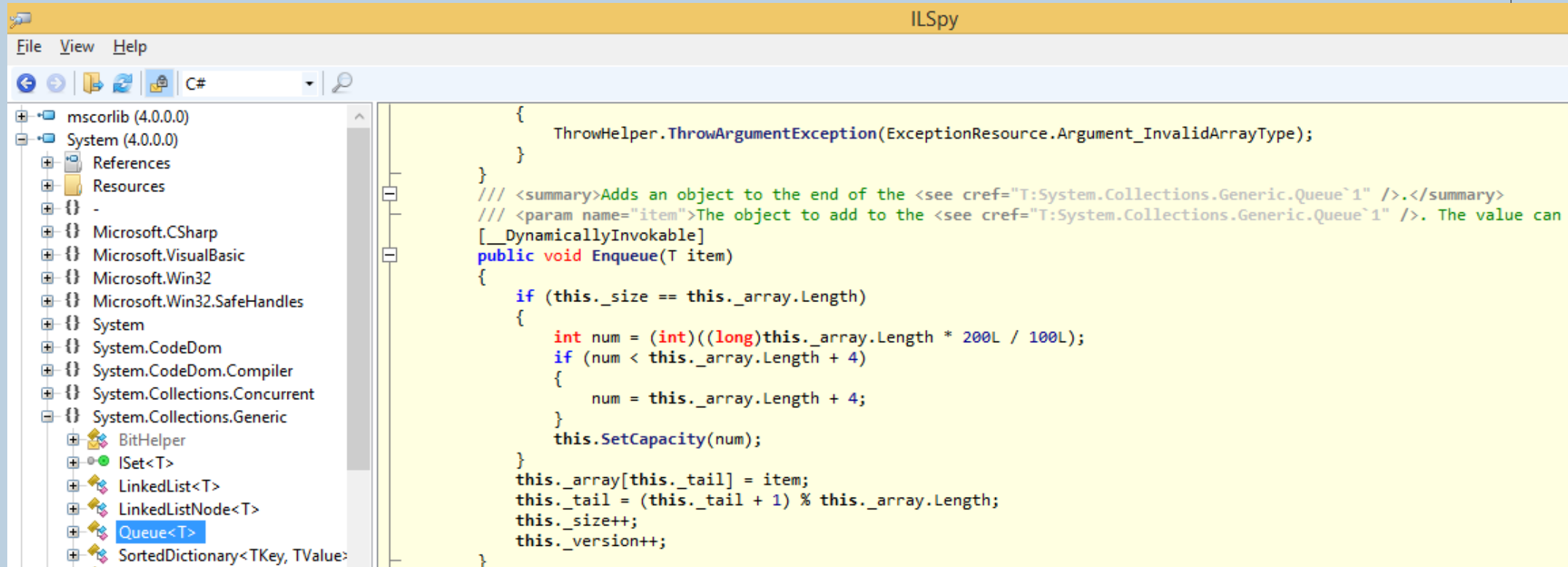
```
TypeDef #5 (02000006)
-----
TypeDefName: WpfApplication1.MainWindow (02000006)
Flags      : [Public] [AutoLayout] [Class] [AnsiClass] [BeforeFieldInit] (00
Extends    : 01000003 [TypeRef] System.Windows.Window
Field #1 (04000004)
-----
Field Name: _contentLoaded (04000004)
Flags      : [Private] (00000001)
CallConvtn: [FIELD]
Field type: Boolean

Method #1 (06000010)
-----
MethodName: .ctor (06000010)
Flags      : [Public] [HideBySig] [ReuseSlot] [SpecialName] [RTSpecialN
RVA        : 0x0000023cf
ImplFlags  : [IL] [Managed] (00000000)
CallConvtn: [DEFAULT]
hasThis
ReturnType: Void
No arguments.

Method #2 (06000011)
-----
MethodName: Button_Click (06000011)
Flags      : [Private] [HideBySig] [ReuseSlot] (00000001)
RVA        : 0x0000023e4
ImplFlags  : [IL] [Managed] (00000000)
CallConvtn: [DEFAULT]
hasThis
ReturnType: Void
2 Arguments
  Argument #1: Object
  Argument #2: Class System.Windows.RoutedEventArgs
3 Parameters
```

Reflexiót kihasználó eszközök és technológiák

- ILSpy, Reflector, ...
 - .NET dll-ek, exék kódjának tanulmányozását engedik meg
- Intellisense, Properties és más IDE-szolgáltatások
- Több .NET technológia (szerializáció, .NET Remoting, WCF)
- Tesztek



Futásidejű típusanalízis - Assembly

```
Assembly a =  
Assembly.GetExecutingAssembly();
```

```
Assembly a =  
Assembly.LoadFrom(„Path.To.Assembly”);
```

```
Assembly a = Assembly.Load(bytes);
```

```
Assembly a = type.Assembly;
```

```
a.GetTypes() – típusok kinyerése
```

```
a.EntryPoint – belépési pont (metódus,  
exék esetén)
```

Futásidejű típusanalízis - Type

- `Type t = assembly.GetType(„Type.Name.In.Assembly”);`
- `Type t = typeof(int);`
- `Type t = typeof(T);`
- `Type t = obj.GetType();`
- `Type t = Type.GetType(„Type.Name.In.Any.Assembly”);`
 - Ha nem az aktuálisan végrehajtódó szerelvényben vagy az mscorlib.dll-ben van, akkor ún. „assembly-qualified name” megadása szükséges
- `t.FullName`, `t.AssemblyQualifiedName` – nevek különféle formában
- `t.BaseType`, `t.IsSubclassOf(anotherType)`,
`t.IsAssignableFrom(anotherType)` – ős, utód vizsgálat

Futásidejű típusanalízis – MethodInfo, PropertyInfo, FieldInfo

- PropertyInfo pi =
t.GetProperty("PropName");
- PropertyInfo[] pis = t.GetProperties();
- FieldInfo fi = t.GetField("FieldName");
- FieldInfo[] fis = t.GetFields();
- MethodInfo mi =
t.GetMethod("MethodName");
- MethodInfo mis = t.GetMethods();
- Általában átadható BindingFlags
paraméter, amivel a keresés

szűkíthető/konfigurálható

- PropertyInfo pi =
t.GetProperty("PropName",
BindingFlags.Static |
BindingFlags.NonPublic)
 - Nem publikus (privát) tagok is
megkaphatók
 - Ez (elsősorban) nem arra való,
hogy kijátsszuk a láthatóságokat!

Reflektált kódelemek használata futásidőben

- A reflexióval elért típusok példányosíthatók, a metódusok, tulajdonságok, mezők használatba vehetők futásidőben

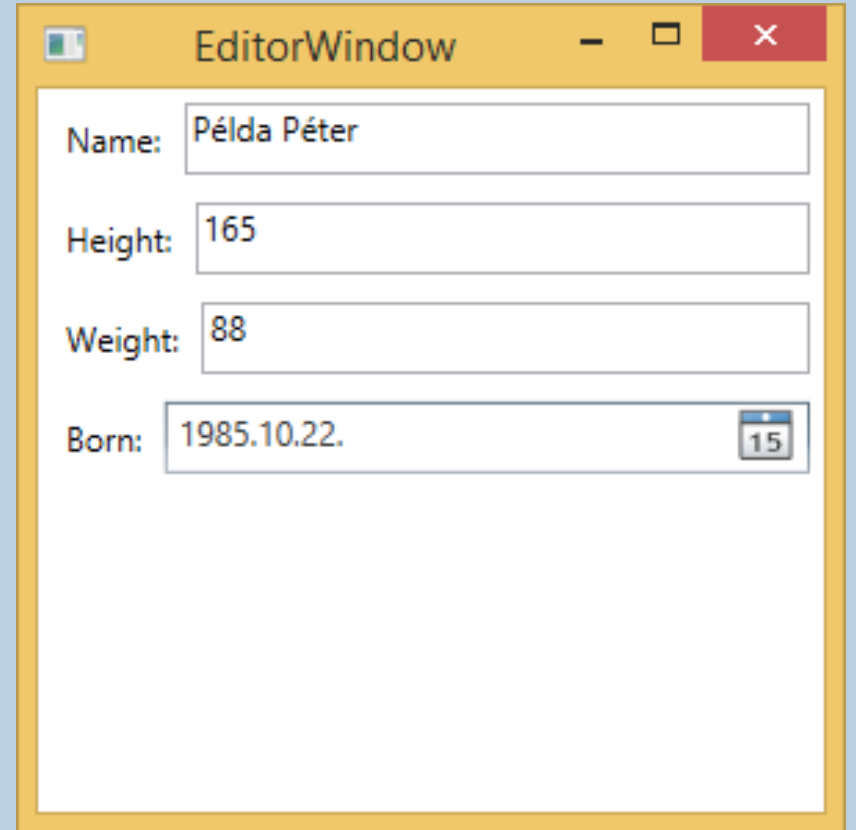
```
Type listType = typeof(List<int>);
MethodInfo addMethod = listType.GetMethod("Add");
PropertyInfo countProperty = listType.GetProperty("Count");

object listInstance = Activator.CreateInstance(listType);

object methodResult = addMethod.Invoke(listInstance,
                                         new object[] { 8 });           // null
object propertyResult = countProperty.GetValue(listInstance); // 1
```

Feladat

- Készítsünk olyan osztályt, amely automatikus szerkesztő GUI-t képes készíteni tetszőleges típusú példány számára
- A szerkesztő felületet a beadott példány tulajdonságainak vizsgálatával készítjük el



The screenshot shows a window titled "EditorWindow" with a yellow title bar. Inside the window, there is a form with four labeled input fields:

- Name:** Példa Péter
- Height:** 165
- Weight:** 88
- Born:** 1985.10.22. (with a calendar icon on the right showing the date 15)

Attributum

- A C# attribútumai hatékony mechanizmust biztosítanak a **kódelemek metaadatainak és futásidejű viselkedésének kiterjesztésére**.
- Azáltal, hogy egyéni attribútumokat adnak hozzá az osztályokhoz, metódusokhoz, tulajdonságokhoz és mezőkhöz, a fejlesztők javíthatják az ezekhez az elemekhez kapcsolódó **információkat**, és befolyásolhatják viselkedésüket futás közben.

Attribútumok

- A fordító által generált metaadat mellé saját metaadat is felvehető
 - Szerelvény, típus vagy tagok esetében is
- System.Attribute osztály utódai
 - Léteznek beépített attribútum típusok különféle célokra, vagy saját Attribute utód is létrehozható
- Használata speciális formában történik
 - Névtér, osztály, metódus, tulajdonság, mező stb. fölött – attribútumtól függően – [XXX], ha az Attribute utódosztály neve XXXAttribute

```
[Obsolete("Do not use this method, use New() instead.")]
static void OldMethod()
{ }

static void NewMethod() { }

static void Main(string[] args)
{
    OldMethod();    //Warningot eredményez
}
```

Attribútumok

- CallerMemberName
 - Ha a paraméter nem kap értéket, akkor a hívó neve kerül bele

```
protected void OnPropertyChanged([CallerMemberName] string propertyName = null)
{
    if (PropertyChanged != null)
    {
        PropertyChanged(this, new PropertyChangedEventArgs(propertyName));
    }
}
```

```
class Settings : Bindable
{
    private string setting1;
    public string Setting1
    {
        get { return setting1; }
        set { setting1 = value; OnPropertyChanged(); }
    }
}
```

Attribútumok felhasználása példa

- Szerializáció
 - adat tárolható formába alakítása, a példában bináris

Kódpélda-szerializáció

- `[Serializable]`
- `class Settings`
- `{ public string Setting1 { get; set; }`
- `public int Setting2 { get; set; }`
- `[NonSerialized]`
- `private int temp;}`
- `class Program`
- `{ static void Main(string[] args)`
- `{ Settings settings = new Settings();`
- `//...`
- `BinaryFormatter formatter = new BinaryFormatter();`
- `using (FileStream stream=new FileStream("settings.dat", FileMode.Create))`
- `{ formatter.Serialize(stream, settings); }`
- `} //mentett információ visszaolvasása: FileMode.Open, Deserialize()`

Saját attributum létrehozása

- Az egyéni attribútumok a C#-ban osztályokként valósulnak meg,
- amelyek az **Attribute** alaposztályból származnak.
- Ezek az attribútumosztályok ezután különböző kódelemekre alkalmazhatók további metaadatok biztosítására vagy viselkedések konfigurálására.

Attribute osztály fontos elemei

- AttributeUsage tulajdonság
 - Paraméterei:
 - AttributeTarget – ki használhatja
 - AllowMultiple- többször is lehet használni?
 - Inherited – átadódik a leszármazott osztálynak

```
[AttributeUsage(AttributeTargets.All, AllowMultiple = true)]
```

Saját attribútum osztály definíció

- Saját attribútum létrehozása
 - Konstruktor
 - Tulajdonságok
 - metódusok
- Attribútum elérése reflexióval
 - Az eddig említett módszerek is ezt használják

```
[AttributeUsage(AttributeTargets.Property)]  
class HelpAttribute : Attribute  
{  
    public string HelpURL { get; private set; }  
  
    public HelpAttribute( string helpURL)  
    {  
        this.HelpURL = helpURL;  
    }  
}
```

Saját attributum használata

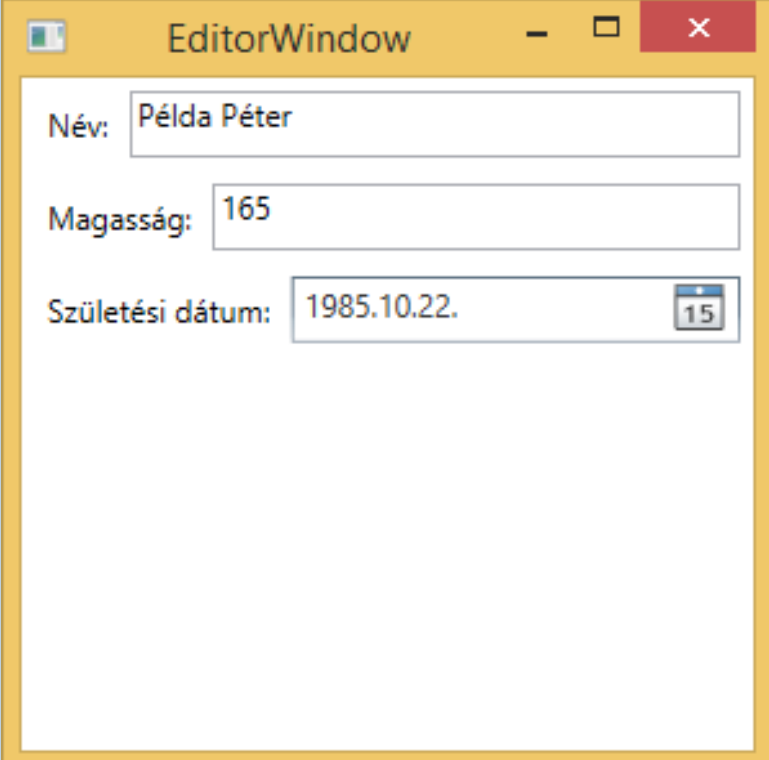
1. Saját osztály kidekorálása

2. Reflexión keresztül meghívni az attributum tulajdonságait vagy metódusait

- `[Help("http://path.to.my.help.for.setting1.html")]`
- `public string Setting1 { get; set; }`
- `//PropertyInfo propertyInfo =
typeof(Settings).GetProperty("Setting1");`
- `HelpAttribute helpAttribute =
propertyInfo.GetCustomAttribute<HelpAttribute>();`
- `Console.WriteLine(helpAttribute.HelpURL);`

Feladat

- Egészítsük ki az előzőleg létrehozott GUI-szerkesztőt a következő funkcionálitással:
- Lehesse tetszőleges „barátságos” nevet megadni a tulajdonságoknak – a tulajdonság eredeti neve helyett ez legyen kiírva
- Lehesse megtiltani, hogy a tulajdonság számára létrejöjjön szerkesztő



The screenshot shows a window titled "EditorWindow" with a yellow title bar. Inside the window, there are three input fields arranged vertically. The first field is labeled "Név:" and contains the text "Példa Péter". The second field is labeled "Magasság:" and contains the number "165". The third field is labeled "Születési dátum:" and contains the date "1985.10.22.". To the right of the date field is a small calendar icon with the number "15" on it.

Köszönöm a figyelmet!