.NET

BCL

Albert István
ialbert@aut.bme.hu
QB. 221, 463-1662

BME, Automatizálási és Alkalmazott Informatikai Tanszék

Tartalom

Gyűjtemények

- Tömbök
- Gyűjtemények (IEnumerable)
  - > Objektumok halmazának kezelése
- System.Collections
- System.Collections.Generic
- System.Collections.Specialized
- …

Sorosítás (serialization)

- Objektum-gráf leképezése adatfolyamra
  - > Bináris, XML, SOAP, json, …
- Beépített sorosítók
  - > **BinaryFormatter**, XMLSerializer
- Külső komponensek
  - > JSON.NET
- Általában kiterjeszthetőek, testreszabhatók
  - > Például származtatás az ISerializable interfészből
- Nem végez titkosítást!

Edit and Continue

- Használd!
- Gyorsabb fejlesztési, javítási ciklus!
- Az új Visual Studio sokkal több mindent támogat, mint elődjei

Reguláris kifejezések

- „Csak azt tudom, hogy körülbelül hogy néz ki”
- A reguláris kifejezések segítségével mintákkal dolgozunk, nem pontos értékekkel
- A keretrendszer teljes támogatást ad a reguláris kifejezésekhez
  - > Melyek kompatibilisek a PERL 5-el
  - > A VS is támogatja a .NET-ben implementált kifejezéseket
  - > Letölthető segédprogramok (regex workbench)

Globalizáció, lokalizáció

- Globalizáció
  - > Az alkalmazásban nincs kultúrafüggő rész, tehát használható más földrészen is
- Lokalizáció - honosítás
  - > Egy nyelv (vagy kultúra) sajátjaként használhatja
- System.Globalization névtér

Formázás

- **String interpolation:**

  $"Szia! {nev} éppen {kor} éves"
- String.Format(string formatString, params object [] values)

  .Format("Szia! {0} éppen {1} éves", nev, kor)
- { N [, M][: formázó]}
  - > N: paraméter sorszáma
  - > M: opcionális, kimenet szélességét befolyásolja, balra/jobbra igazított (pozitív/negatív szám)
  - > formázó: opcionális, típusfüggő formázó paraméter

DateTime vs DateTimeOffset

- 0001. január 1 éjfél óta eltelt idő 100 nanoszekundumokban (tick)
- DateTime (62+2 bit)
  - > Időpont, időzona nélkül
  - > Kind: UTC, helyi, időzóna nélküli
- DateTimeOffset (64+16 bit)
  - > Időzóna eltolódás információ
    - Az időzónát nem tárolja el
- TimeSpan: időintervallum

Gyűjtemények

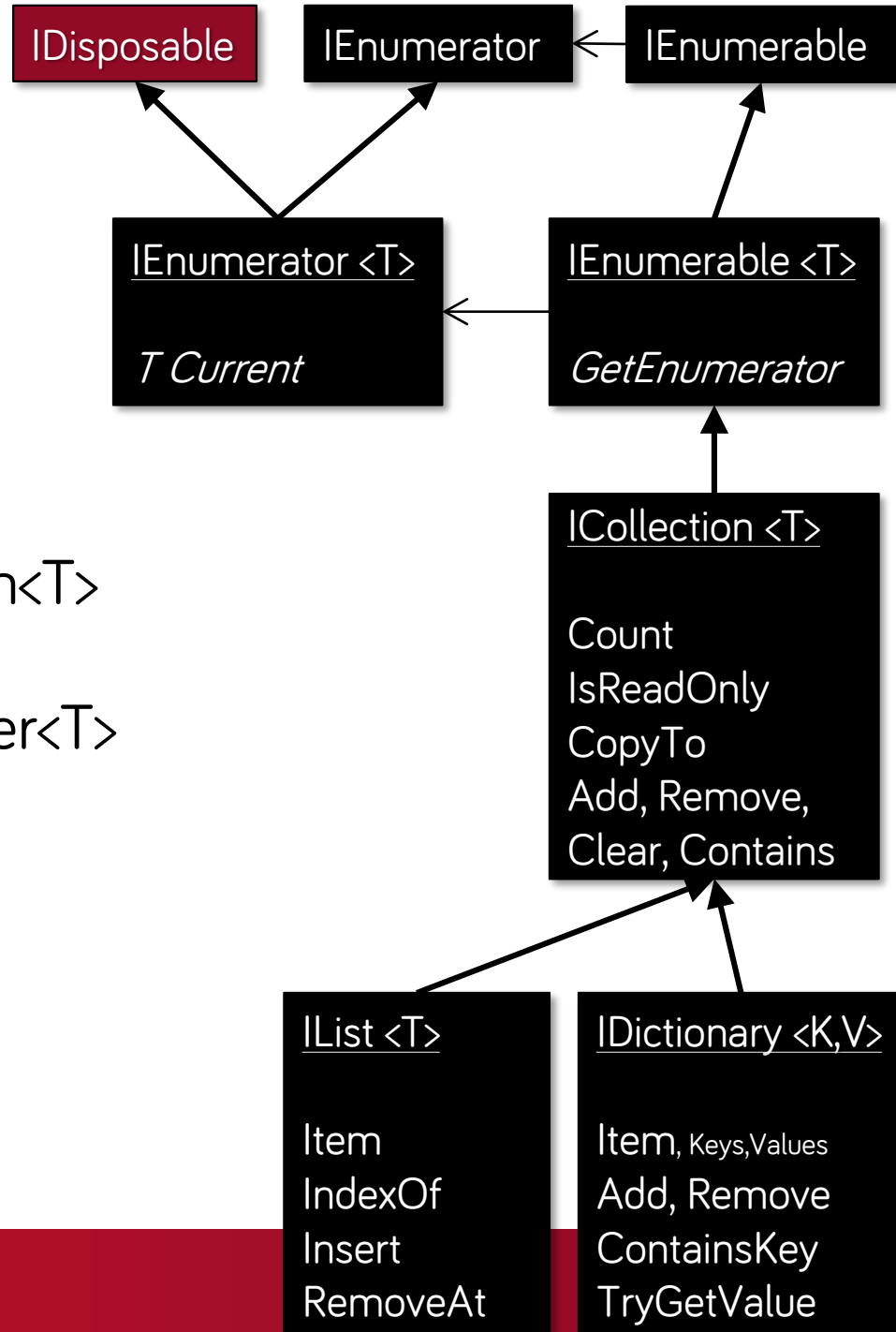
- Tömbök
- Gyűjtemények (IEnumerable)
  - Objektumok halmazának kezelése
- System.Collections
- System.Collections.Generic
- System.Collections.Specialized
- …

Tömbök

- System.Array osztály, polimorfikus
- Korlátlan dimenzió, illetve hossz
- Létrehozáskor kell definiálni
- Sort statikus metódussal támogatja a rendezést
  - IComparable objektumok
  - Külső összehasonlítás IComparer-rel
- BinarySearch: bináris keresés már berendezett tömbökön

Generikus típusok

- Erős típusosság
- A kasztolások idő- és erőforrásigényesek
- Generikus interfészek
  - IEnumerator<T>, ICollection<T>
  - IList<T>, IDictionary<K,V>
  - IComparable<T>, IComparer<T>
- List <T>
- Dictionary <K,V>
- Stack <T>, Queue <T>
- LinkedList <T>



Gyűjtemények – interfészek

- ICollection<T> : IEnumerable<T>
  - Count(), CopyTo()
  - Add(), Remove(), Contains() és Clear() függvények
- IDictionary<T> : ICollection<T>
  - IDictionary<K,V> : IEnumerable<KeyValuePair<K,V>>
  - Alap asszociációs tömb
    - Kulcs-érték párok táblázata
    - Az Item indexer a kulcs szerinti értéket keresi elő
- IList<T> : ICollection<T>
  - Olyan gyűjtemény, melynek elemeit külön is lehet indexelni
    - Az Item indexer az index szerinti értéket keresi elő

Fontosabb gyűjteményosztályok

- SortedList (key-value + index)
- BitArray
- StringCollection (string List)
- StringDictionary (string Hashtable)

Csak olvasható interfészek (4.5)

- IReadOnlyCollection<T> : IEnumerable<T>
  - Count
- IReadOnlyList<T> : IReadOnlyCollection<T>
  - Index [ ]
- IReadOnlyDictionary<K,V>
  : IReadOnlyCollection<KeyValuePair<K, V>>
  - Index [ ]
  - Contains
  - TryGetValue
  - Keys, Values

List<T>

- FindAll(Predicate<T> match)
- ForEach
- TrueForAll
- RemoveAll

- Példa:

```
List<MyItem> evenList =
        myList.FindAll( item => item.Value % 2 = = 0 );
```

További gyűjtemények

- SortedSet<T>
  - Mindig sorrendben tartja az elemeket
  - Duplikált elemeket nem enged meg
    - Ilyenkor false értékkel tér vissza a hozzáadás.
- HashSet<T>
  - Nem tartja sorrenben a benne lévő elemeket
  - Duplikált elemeket nem enged meg
- ISet<T> interfészt implementálnak
  - Add, Remove, …
  - IsSubsetOf, …, UnionWidth, ExceptWith, …

Összehasonlító interfészek

- IComparer<T>
  - két értéket haszonlít össze
  - int Compare(T x, T y);
- IComparable<T>
  - önmagát hasonlítja egy másikhoz
  - int CompareTo(T obj);
- IEquatable<T>
  - csak egyenlőséget vizsgál
  - bool Equals(T other);

ReferenceEquals, Equals, op ==

- static bool ReferenceEquals(object, object)
  - Referenciát ellenőriz, értéktípusra mindig false
- static bool Equals(object a, object a)
  - True ha azonos a referencia vagy a.Equals(b) true
  - A bal oldali objektumot használja !
- bool Equals(object)
  - Referencia típus (object): ReferenceEquals
  - Érték típus (ValueType override): tartalom szerint
    - MemCmp vagy reflexió (ami NAGYON LASSÚ)
- op ==:
  - Referencia típusoknál: Equals()
  - Értéktípusoknál nincs alapértelmezett, de override-olják

GetHashCode

- Csak azokhoz a típusokhoz, amiket hashtáblában tárolunk
  1. Ha két példány azonos (Equals), legyen a két hash érték is az
  2. Egy példány hash kódja nem változhat meg
     > Így mindig a helyes zsákban van
  3. Használja ki a teljes értéktartományt

GetHashCode

- object.GetHashCode(): minden példányhoz tartozik egy pszeudo random érték
  - 1: Ha átírjuk az Equals-t, akkor ezt is át kell
  - 2: OK
  - 3: OK
- ValueType.GetHashCode(): az első nem null mező GetHashCode-ját használja
  - 1: Csak ha az Equals is az első értéket hasonlítja
  - 2: Csak ha az első mező nem változik
  - 3: ?

Sorosítás (serialization)

- Objektum-gráf leképezése adatfolyamra
  - Bináris, XML, SOAP, json, …
- Beépített sorosítók
  - **BinaryFormatter**, XMLSerializer
- Külső komponensek
  - JSON.NET
- Általában kiterjeszthetőek, testreszabhatók
  - Például származtatás az ISerializable interfészből
- Nem végez titkosítást!

BinaryFormatter, testreszabás

- Mezőket sorosít, nem propertyket!
- Automatikus bináris sorosítás
- A sorosítható osztályt a `[Serializable]` attribútummal kell megjelölni
  - `[NonSerialized]` azokon a mezőkön, amit ki kell hagyni
  - Az opcionális (új) mezőket a `[OptionalField]` attribútummal kell megjelölni
- `IDeserializationCallback`

```
void OnDeserialization( object sender );
```

  - Sorosítás után hívja meg, például hash táblán

A sorosítás testreszabása: ISerializable

```
GetObjectData(
                SerializationInfo info,
                StreamingContext context  )
```

- `info.AddValue( name, value )`
- `info.SetType( Type )`
- `info.GetUInt32(), stb.`

- Kötelező .ctor – azonos paraméterezéssel

Sorosítóhelyettes: ISerializationSurrogate

- Sorosítás más objektumok helyett
  - Azokhoz nem tették hozzá a [Serializable] attriíbútumot
  - Különböző verziók támogatása

```
void GetObjectData(
                    object obj,
                    SerializationInfo info,
                    StreamingContext context );

object SetObjectData( ...
                    ISurrogateSelector selector );
```

Json .NET 1

- Egyszerű, gyors sorosítás, visszaállítás

```
Product product = new Product();
product.Name = "Apple";
product.Expiry = new DateTime(2008, 12, 28);
product.Sizes = new string[] { "Small" };

string json = JsonConvert.SerializeObject(product);
// {
//   "Name": "Apple",
//   "Expiry": "2008-12-28T00:00:00",
//   "Sizes": [
//     "Small"
//   ]
// }
```

```
string json = @"{
  'Name': 'Bad Boys',
  'ReleaseDate': '1995-4-7T00:00:00',
  'Genres': [
    'Action',
    'Comedy'
  ]
}";

Movie m = JsonConvert.DeserializeObject<Movie>(json);

string name = m.Name;
// Bad Boys
```

Json .NET 2

- Nem .NET-es típusok sorosítására készült!
- Nincs típusinformáció
  - Bekapcsolható
- Nem kezel generikus típusokat
  - Kiterjeszthető…
- Körkörös hivatkozás kezelés bekapcsolható

Könyvtár kezelés

- Directory és File: string alapú statikus metódusok
  - Hagyományos műveletek: törlés, másolás, mozgatás
  - Attribútumok, dátumok, fájlméret, Exists stb.
  - ACL alapú hozzáférésszabályozás
- Directory: mappa kezelés
  - Fájlok listája: szinkron és aszinkron módon
  - GetFiles / GetDirectories / GetFileSystemEntries
  - EnumerateFiles / EnumerateDirectories / …
- File: fájl kezelés
  - Egyszerű szöveg kezelés, aszinkron módon is
    - Append, ReadAll, Replace, ReadLine, WriteAllText, …
- Titkosítás, DPAPI

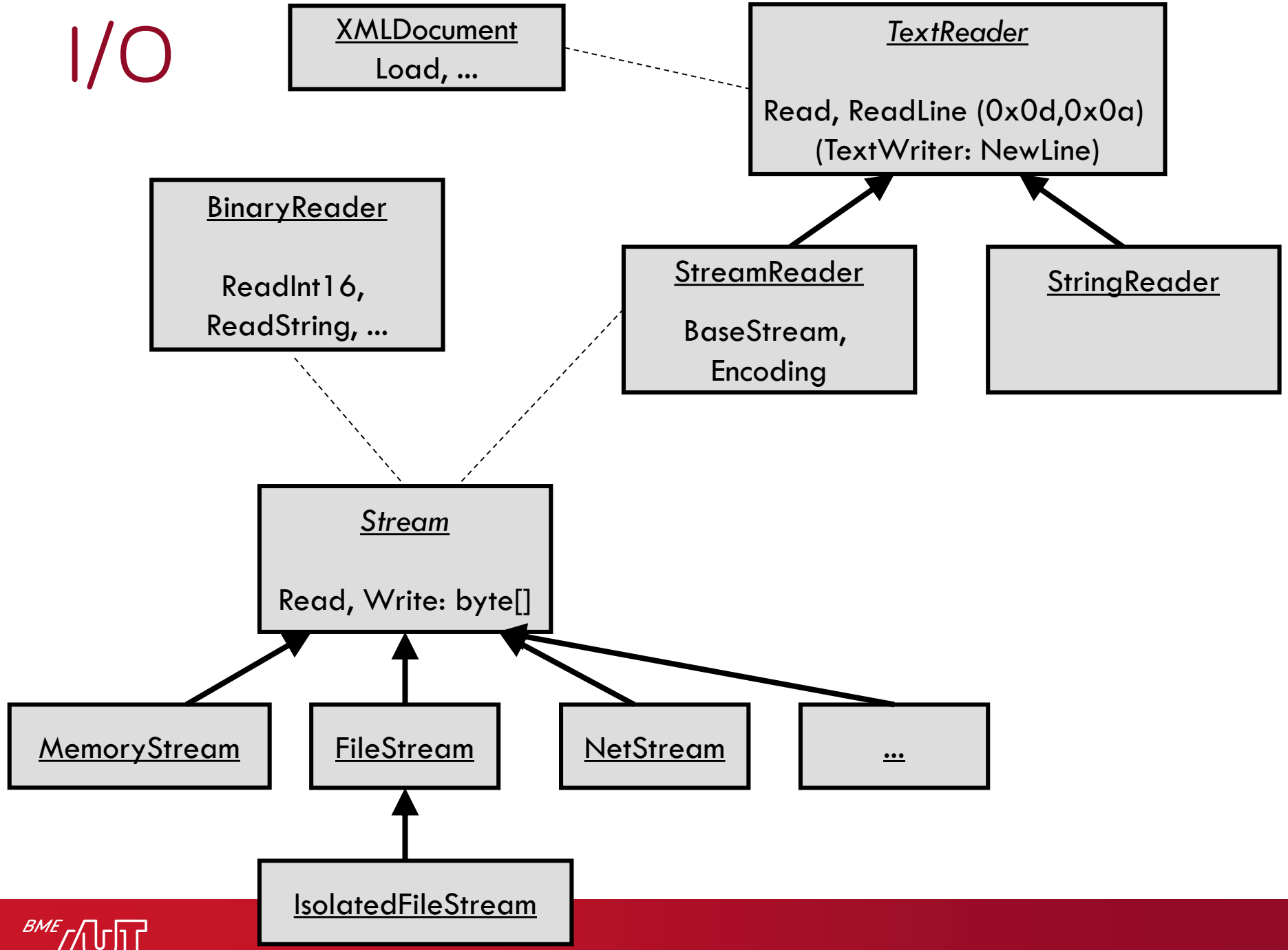
File- és DirectoryInfo alapú megoldás

- DirectoryInfo és FileInfo példány alapú elérés
- Közvetlenül példányosítható az elérési út megadásával, Refresh-sel frissíthető állapot
- A korábbi műveletek mind elérhetőek
- FileInfo példányok a DirectoryInfo.GetFiles / … metódusaival
- FileInfo: fájlmegnyitás Open… hívásokkal
  - Open(), OpenRead(), OpenWrite(), OpenText()
  - Egy Stream-et ad vissza

Folyam kezelés

- Absztrakt alaposztály: System.IO.Stream
  - Read(), Write(): alap szinkron elérés
  - Aszinkron támogatás
    - ReadAsync, WriteAsync metódusok
- FileStream
  - Fájlok közvetlen elérését teszi lehetővé
  - File.Open(), ...
- MemoryStream
  - Csak a memóriában létező folyamot állít össze
  - Hasznos bináris adatok kezelésére

I/O


XMLDocument
Load, ...
TextReader
Read, ReadLine (0x0d,0x0a)
(TextWriter: NewLine)
BinaryReader
ReadInt16,
ReadString, ...
StreamReader
BaseStream,
Encoding
StringReader
Stream
Read, Write: byte[]
MemoryStream
FileStream
NetStream
...
IsolatedFileStream

Folyam olvasása

- Magasabb szintű Stream olvasást tesznek lehetővé
- BinaryReader
  - Bináris adatok típusos adatolvasásához
  - Az alaptípusokhoz van Read metódus
    - ReadInt16(), ReadBoolean(), ReadDouble(), stb.
- TextReader
  - Absztrakt alaposztály szöveges állományok olvasására
  - ReadLine() – egész sor olvasása
  - ReadBlock(index, count)
  - ReadToEnd() – az egész állomány beolvasása
- StreamReader : TextReader
- StringReader : TextReader
  - String példány hasonló kezelésére

Folyam írása

- Magasabb szintű Stream írást tesznek lehetővé
- BinaryWriter
  - Bináris adatok típusos kiírására
  - Több mint 15 Write() override a megfelelő típusoknak
- TextWriter
  - Absztrakt alaposztály a String-ek Stream-be írásához
  - Write / WriteLine, … Metódusok
  - Encoding, NewLine, FormatProvider beállítás
- StreamWriter : TextWriter
- StringWriter : TextWriter
  - Hasonló, StringBuilderbe lehet írni a TextWriterrel

Hálózat kezelés

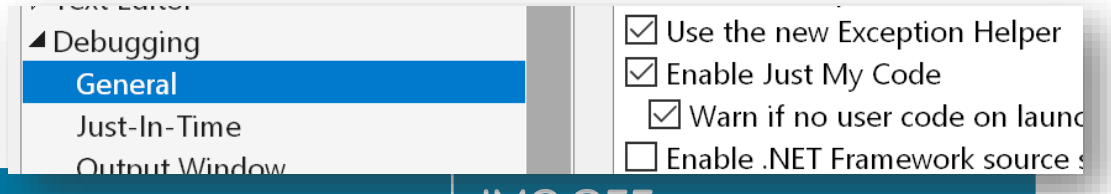
- System.Net tartalmazza az összes hálózati protokollt támogató osztályt
  - TCP, IP, UDP támogatás
- Alkalmazásszintű protokoll támogatása
  - HTTP, HTTPS, FTP
- A HTTP autentikációs metódusok támogatása
  - Basic, Digest, NTLM Challenge/Reponse
- Teljes HTTP cookie támogatás

Edit and Continue

- Használd!
- Gyorsabb fejlesztési, javítási ciklus!
- Az új Visual Studio sokkal több mindent támogat, mint elődjei

Debug attribútumok

- [DebuggerBrowsable]
  - DebuggerBrowsalbeState
    - Collapsed, Never, RootHidden
- [DebuggerDisplay]
  - "literal {tagváltozó}"
- [DebuggerVisualizer] (saját visualizer)
- JMC: JustMyCode Debugging



	JMC ON	JMC OFF
[DebuggerHidden]	Nincs töréspont Nincs step-in (F11) Látszik a call stackben	Töréspont működik Step-in működik (F11) Látszik a call stackben
[DebuggerStepThrough]	Nincs töréspont Nincs step-in (F11) Látszik a call stackben	Töréspont működik Step-in működik (F11) Látszik a call stackben
[DebuggerNonUserCode]	No break / no step in Nem látszik a call stackben	No break / no step in Nem látszik a call stackben

Példa

Watch 1

Name	Value	Type
p	"Krisztina" (42)	ConsoleApplication6.Person
Age	42	int
[0]	"Palkó" (13)	ConsoleApplication6.Person
[1]	"Edina" (11)	ConsoleApplication6.Person
Raw View		
Mother	null	ConsoleApplication6.Person
Name	"Krisztina"	string
Siblings	Count = 1	System.Collections.Generic.List<C
[0]	"Erzsébet" (35)	ConsoleApplication6.Person
Raw View		

```
[DebuggerDisplay("{Name} ({Age})")]
14 references
public class Person
{
    4 references
    public string Name { get; set; }
    4 references
    public int Age { get; set; }
    2 references
    public Person Mother { get; set; }

    [DebuggerBrowsable(DebuggerBrowsableState.RootHidden)]
    3 references
    public List<Person> Children { get; set; }

    2 references
    public List<Person> Siblings { get; set; }

    4 references
    public Person()
    {
        this.Children = new List<Person>();
        this.Siblings = new List<Person>();
    }
}
```

System.Diagnostics.Debugger

- Lehetővé teszi az alacsony szintű kommunikációt a Debuggerrel
  - **Break()**
  - **Log(int level, string category, string message)**
    - category max 256 karakter hosszú!
  - IsAttached
  - IsLogging()
  - Launch()

System.Diagnostics . Debug / Trace

- **Debug**: csak Debug módban fordított kód esetén kerül bele a kódba
- **Trace**: a Debug és Release módban fordított szerelvényekbe is bekerül

- Assert(condition)
  - A kód nélküle is ugyanúgy működjön!
- Fail
  - Hibaüzenet kiírása
- Indent/UnIndent
  - debug üzenetek formázásához
- Print
  - csak csatolt Listenerbe ír
- Write, WriteLine, WriteIf, WriteLineIf
  - Visual Studio Output window
- Close, Flush – bufferelt adat ürítése → Listenereknél látni fogjuk

Trace Listeners

- DefaultTraceListener
  - Visual Studio Output window
- TextWriterTraceListener
  - Szöveges file-ba vagy stream-be ír
  - listener = new TextWriterTraceListener(@"C:\output.txt");
- XmlWriterTraceListener
  - A TextWriterTraceListener-re épül, kiterjesztett funkcionalitással
- EventLogTraceListener, DelimitedListTraceListener

TraceSource, TraceSwitch

- TraceSource .ctor
  - A kiírandó információ forrását reprezentálja, kötelező nevet megadni
- TraceSwitch
  - Megváltoztatja a Trace osztály viselkedését
  - TraceLevel: Off < Error < Warning < Info < Verbose
- TraceEventType
  - Start, Stop, Suspend, Resume
  - Error, Critical
  - Information, Warning, Verbose, Transfer
- TraceSource.Switch = SourceSwitch
  - SourceLevel
    - Off < Critical < Error < Warning < Information < Verbose < All

TraceListeners

- Teljes konfigurációs file támogatás

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
    <system.diagnostics>
        <trace autoflush="true" indentsize="5">
            <listeners>
                <add name="DemoListener" type="System.Diagnostics.XmlWriterTraceListener"
                initializeData="output.xml" />
                <remove name="Default" />
            </listeners>
        </trace>
    </system.diagnostics>
</configuration>
```

Egy „trükkös” segédosztály példa

```
internal class DebugTracer : IDisposable {
    Stopwatch sw; string at;
    public DebugTracer( string at )
    {
            this.at = at;
            Debug.WriteLine( at + " started" );
            sw = Stopwatch.StartNew();
    }
    public void Dispose()
    {
            sw.Stop();
            Debug.WriteLine( at + " finished in " + sw.ElapsedMilliseconds + " ms." );
} }
```

```
using(new DebugTracer("szamitas")
{
            // szamitas
}
```

Reguláris kifejezések

- „Csak azt tudom, hogy körülbelül hogy néz ki”
- A reguláris kifejezések segítségével mintákkal dolgozunk, nem pontos értékekkel
- A keretrendszer teljes támogatást ad a reguláris kifejezésekhez
  - Melyek kompatibilisek a PERL 5-el
  - A VS is támogatja a .NET-ben implementált kifejezéseket
  - Letölthető segédprogramok (regex workbench)

Meta karakterek

- Egy reguláris kifejezés egy egyszerű string
- Egy reguláris kifejezés kétféle karakterből áll
  - Normál karakter
  - Metakarakter
- A metakarakterekkel definiálhatjuk magát a mintát: speciális jelentéssel bírnak
  - Például copy *.txt

Mintaillesztés

- A mintaillesztés egy illesztési fa építésével történik
- Egy ág meghiúsulása esetén a rendszer képes visszalépni, és másik ágon folytatni a keresést
- k(e|é)k illesztése a „kék” szövegre
  - k, stimmel
  - Elágazás
    - e, rossz, visszalépés
    - é stimmel
  - k, stimmel

Speciális karakterek

- A .NET reguláris kifejezése számtalan metakarakterrel dolgozik, néhány közülük
  - \w = [a-zA-Z0-9]
  - \W = [^a-zA-Z0-9]
  - \s = whitespace [\f\n\r\t\v]
  - \d = szám
  - ^kif = a kifejezés a sor elején található
  - kif$ = sor végén

Mennyiséget kifejező karakterek

- Mikor kell ismétlés számmal dolgozni, például
  - Négyjegyű számok - irányítószám
  - Dupla karakterek
  - Telefonszámok
  - …
- Legfontosabbak
  - * - 0 vagy többször
  - + - 1 vagy többször
  - ? – 0 vagy 1-szer
  - {n} – pontosan n-szer
  - {n,m} – n és m között
  - {n,} – legalább n-szer

Csoportok definiálása

- Csoportokat egyszerűen zárójelezéssel lehet létrehozni
  - Kif1(csoportKif)Kif2
- Újabb találati egységnek, al-reguláris kifejezésnek felelnek meg
- Nevesíthetők, így később hivatkozhatunk rá
  - (?<csoportnév>kif)
  - Utalás vagy a nevével, vagy a találat számával
    - „(k(é|e)k)\1 – „kékkék” vagy „kekkek”
    - (?<csop>k(é|e)k)\k<csop> - „kékkék” vagy „kekkek”
- A teljes reguláris kifejezés maga is egy csoport

Regex osztály

- Ez képviseli a teljes reguláris kifejezés támogatást
- Főbb műveletek
  - Match: keres
  - Replace: cserél
  - Split: darabol
- Többszálú
- Előre fordítható
- Timeout

Globalizáció, lokalizáció

- Globalizáció
  - Az alkalmazásban nincs kultúrafüggő rész, tehát használható más földrészen is
- Lokalizáció - honosítás
  - Egy nyelv (vagy kultúra) sajátjaként használhatja
- System.Globalization névtér

Kultúra információ

- CultureInfo osztály (több száz kultúra támogatása)
  - **Invariáns**: kultúrafüggetlen, nincs tekintettel semelyik ország területi beállításaira (angol ☺)
  - **Semleges**: Csak nyelvhez tartozó beállítások, országhoz nem (pl: „en”)
  - **Specifikus**: Nyelv és országfüggő kultúra (pl: „en-GB”, „en-US”)
- Minden kultúrához tarozik LCID is
  - általában a különböző rendezéseket szokták így megadni
  - pl: technikai (betűszerint) vagy alapértelmezett (hangszerint) – „cs”.CompareTo(„cx”)
  - Az LCID Windows szinten támogatott finomabb hangolást tesz lehetővé

Kultúrainformációk

- Lekérdezhetőek
  - Dátum formátum (DateTimeFormat)
    - DateTime.Now.ToLongDateString()
  - Szám formátum (NumberFormat)
    - Pénz formátum is
  - Használt fizetőeszköz, mértékrendszer, ...

Szál információ

- Minden szálhoz két kultúrainformáció tartozik
- CurrentCulture
  - kultúra beállítás
  - elsősorban a formázásra van hatással
- CurrentUICulture
  - erőforrás menedzsernek szól
  - különböző nyelvű erőforrásokat olvas be
- A formázáshoz nem kell mindig átállítani a szál beállításait
  - ToString(..., CultureInfo)

Formázás

- **String interpolation:**

  $"Szia! {nev} éppen {kor} éves"

- String.Format(string formatString, params object [] values)

  .Format("Szia! {0} éppen {1} éves", nev, kor)

- { N [, M][: formázó]}
  - N: paraméter sorszáma
  - M: opcionális, kimenet szélességét befolyásolja, balra/jobbra igazított (pozitív/negatív szám)
  - formázó: opcionális, típusfüggő formázó paraméter

A formázás két összetevője

- Format Specifier
  - a kimenet alakját határozza meg, kultúrafüggetlen
  - szám, dátum (idő), enumeráció

- Format Provider
  - segédinformáció (például a tizedes pont formája)
  - hónapok nevei, sorrend, stb.
  - általában kultúrafüggő

Néhány Format Specifier

- C – pénz
- D – decimális egész szám megjelenítés
- E – exponenciális alak
- F – fixpontos (megadható a tizedesek száma)
- N – szám (ezres vesszők)
- G – általános (ToString();)
- R – roundtrip –információvesztés nélkül
- X – hexadecimális

Néhány példa

Karakter	Leírás	Példa	Kimenet
C or c	Currency	Console.Write("{0:C}", 2.5); (-2.5).ToString("C");	$2.50 ($2.50)
D or d	Decimal	Console.Write("{0:D5}", 25);	00025
E or e	Scientific	Console.Write("{0:E}", 250000);	2.500000E+005
F or f	Fixed-point	Console.Write("{0:F2}", 25); Console.Write("{0:F0}", 25);	25.00 25
G or g	General	Console.Write("{0:G}", 2.5);	2.5
N or n	Number	Console.Write("{0:N}", 2500000);	2,500,000.00
X or x	Hexadecimal	Console.Write("{0:X}", 250); Console.Write("{0:X}", 0xffff);	FA FFFF

Számok testreszabása

- 0 – számjegy (0-t ír ha nincs szám)
- # - számjegy (semmit sem ír ha nincs szám)
- . – tizedes pont
- , - ezres elválasztó
- % - százalékot ír ki (fel is szorozza a számot)
- ” és „ – tetszőleges szöveg kerülhet oda

Néhány példa

Érték	Write("{0:...") 1234.5678.ToString("...")	Kimenet
1234.5678	"00000"	01235
0.45678	"0.00", en-US	0.46
0.45678	"#.##", fr-FR	,46
2147483647	"##,#", en-US	2,147,483,647
2147483647	"#,#„", es-ES	2.147
0.3697	"%#0.00", en-US	%36.97
987654	"\###00\#"	#987654#
68	"# ' degrees'"	68 degrees
12.345	"#0.0#;(#0.0#);-\0-"	12.35
0	"#0.0#;(#0.0#);-\0-"	-0-
-12.345	"#0.0#;(#0.0#);-\0-"	(12.35)

Dátumformátumok

- d – rövid dátum
- D – hosszú dátum
- t, T – rövid idő-, hosszú idő formátum
- f, F – teljes dátum, idővel együtt
- g, G – általános

- itt is van testreszabás
  - yyyy, MM, HH, HHH, stb…

Néhány dátum példa

Érték	Write("{0:...") ToString("...")	Kimenet
6/1/2009 1:45:30 PM	"dd"	01
6/1/2009 1:45:30 PM	"ddd"	Mon (en-US) Пн (ru-RU)
6/1/2009 1:45:30 PM	"MMMM"	June (en-US)
6/1/2009 1:45:30 PM	"tt"	午後 (ja-JP)
6/1/2009 1:45:30.617 PM	"ff"	61
6/1/2009 1:45:30.0005 PM	"FFF"	(üres)
6/1/2009 1:45:30 PM	"h"	1
6/1/2009 1:45:30 PM	"hh"	01
6/1/2009 1:45:30 PM	"H"	13

Kultúrafüggő információk (provider)

- pénz esetén a tizedesjegyek száma
- pénz formázási minta
- negatív előjel mintája
- számok esetén az ezres elválasztó
- százalékjel
- napok, hónapok nevei
- …
- és használhatunk más naptárat is (8 db)

DateTime vs DateTimeOffset

- 0001. január 1 éjfél óta eltelt idő 100 nanoszekundumokban (tick)
- DateTime (62+2 bit)
  - Időpont, időzona nélkül
  - Kind: UTC, helyi, időzóna nélküli
- DateTimeOffset (64+16 bit)
  - Időzóna eltolódás információ
    - Az időzónát nem tárolja el
- TimeSpan: időintervallum
- TimeZoneInfo: időzóna információk

BigIntegers struct

- Nagyon nagy egész számok, nincs felső korlát
- Parse + matematikai műveletek
  - Kasztolás más típusok felé
  - Egyéb: IsNumberOfTwo, IsZero, IsOne, …
- Gyors: Optima Team “Microsoft Solver Foundation”

Complex struct

- Komplex szám támogatás
  - Imaginary, Magnitude, Phase, Real
  - Double alapú
- Math osztály metódusaihoz hasonló műveletek, operátorok, constansok

Késői inicializálás

- Időigényes konstruktorral rendelkező objektumok létrehozásának elhalasztása későbbre
- Lazy<T> osztály: .Value, . IsValueCreated
  - Factory metódus átadható a .ctornak
  - Párhuzamos létrehozás szabályozása
  - Létrehozás kivétel tárolása és újradobása
- ThreadLocal<T>: + IDisposable
  - Szálanként külön inicializált példány jön létre
  - Az összes szál példánya elérhető: Values
- LazyInitializer: statikus segédosztály

MemoryMapped fájlok

- Két gyakori forgatókönyv támogatására:
  1. Nagy fájlok hatékony elérése, operációs rendszer támogatás
  2. Folyamatok közötti hatékony kommunikáció (IPC) közös nevesített memóriával

Tuple támogatás

- Referencia típus (class), nem érték típus (struct)
  - Immutable
  - Kevés interfész implementáció: lightweight
- Típusos gyűjtemény fix számú elemhez
  - Minden elem más típusú lehet, mindegyik generikus típusparaméter, 8 felett összefűzhető…
  - Factory metódussal könnyebb létrehozni
  - Item1..7 propertyvel elérhetőek az értékek
- Cél: különböző nyelvek együttműködésének támogatása, sok kompromisszumot kíván