



Aszinkron programozás C#-ban

Aszalos Attila

Tartalom

Bemutakozás

Bevezető

Az aszinkron
programozás alapjai

Könyvtárház
applikáció

Összefoglaló



Bemutatózás (1)



Csapatok

- WPF
- Web
 - MVC
 - React
- Mobil
 - Android – Java, Kotlin
 - iOS – Objective C, Swift
 - ReactNative
- Teszt



Dózsa György utca, 64 – 68 szám
Multinvest Business Center 2, II. emelet
Telefon: +40 265 265487
E-Mail: office@dvse.ro



Bemutatókozás (2)

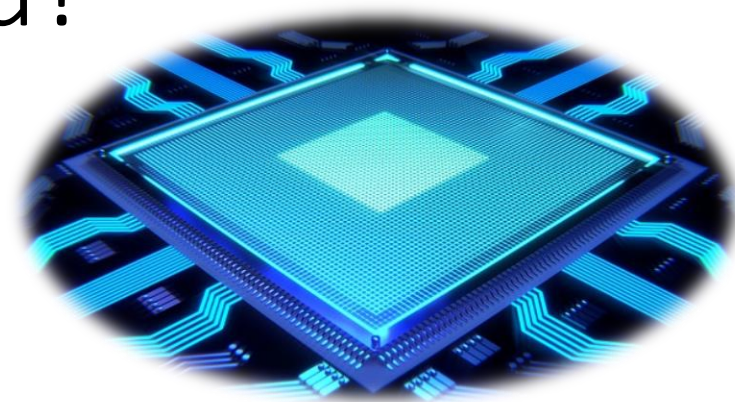
- Tanulmányok:
 - Bolyai Farkas Elméleti Líceum – Matematika Informatika
 - Sapientia EMTE – Számítástechnika
 - Petru Maior Egyetem – IT technológiák



Bevezető



Hogyan hajtódik végre a forráskód?



```
private void ExportButton_Click(object sender, RoutedEventArgs e)
{
    this.ExportProgressBar.Visibility = Visibility.Visible;
    List<BookDetails> allDetails = new List<BookDetails>();
    foreach (Book book in SearchResultsItemsControl.ItemsSource.Cast<Book>().Where(p => p.IsChecked))
    {
        BookDetails bookDetails = this.dataService.GetDetails(book);
        allDetails.Add(bookDetails);
    }

    BookDetailsExporter.Export("my_books", allDetails);
    this.ExportProgressBar.Visibility = Visibility.Collapsed;
}
```

Alapvető művelettípusok

- Rendező algoritmusok
- Kereső algoritmusok
- Kriptográfiai algoritmusok
- Backtrack algoritmusok
- Utazó ügynök probléma

Számítási
művelet

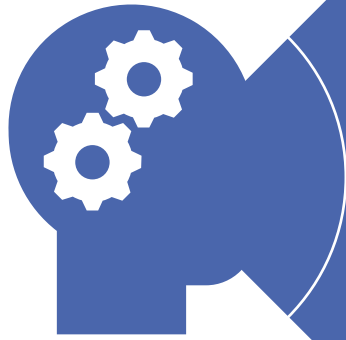
- Hálózati műveletek
 - Letöltés
 - Feltöltés
- Fájlműveletek
 - Írás
 - Olvasás

Erőforrás
hozzáférési
művelet

Mérjük le!



Eredmény



Amit tudunk

- A forráskód szekvenciális
- Egyszerre csak 1 művelet hajtódik végre
- Létezik blokkoló hívás



Amit láttunk

- Megfagy az ablak
- Megfagy a „*progress bar*”
- Megfagy a „*hover*” effektus



Az aszinkron programozás alapjai

Szálak – Amit már tanultunk...

Az operációs rendszer több folyamatot is futtat

- Sorban
- Egyidejűleg

A folyamatok osztoznak az erőforrásokon

- Processzor
- Memória

Egy folyamaton belül létezhet egy vagy több szál

- Minding van egy fő szál („*Main Thread*”)
- „Párhuzamosság illúzió” vs „Valódi párhuzamosság”

A .NET keretrendszer fejlődése

.NET 1.0 13.02.2002

System.Threading

- Thread
- Timer



.NET 2.0 07.11.2005

System.ComponentModel

- BackgroundWorker



.NET 4.0 12.04.2010

System.Threading

- Task



Írjuk át!



BackgroundWorker egyszerű használat

```
private void FindPrime_Click(object sender, RoutedEventArgs e)
{
    BackgroundWorker backgroundWorker = new BackgroundWorker();
    backgroundWorker.DoWork += BackgroundWorker_FindNthPrime;
    backgroundWorker.RunWorkerCompleted += BackgroundWorker_FindNthPrimeCompleted;
    backgroundWorker.RunWorkerAsync();
}
```

Konfigurálás

Nem blokkoló hívása a DoWork-nek

```
private void BackgroundWorker_FindNthPrime(object sender, DoWorkEventArgs e)
{
    int prime = PrimeTool.FindNthPrime(n);
    e.Result = prime;
}
```

Háttér szál

```
private void BackgroundWorker_FindNthPrimeCompleted(object sender, RunWorkerCompletedEventArgs e)
{
    ResultLabel.Content = $"The {n}th prime is: {e.Result}";
}
```

Fő szál

Vigyázat, nem mindegy...

Aszinkron

- Futhat háttérszálon
- Futhat a fő szálon is
 - Példa:

```
Debug.WriteLine("Before begin invoke");
Dispatcher.BeginInvoke((Action)(() =>
{
    Debug.WriteLine("In begin invoke");
}));
Debug.WriteLine("After begin invoke");
```

Háttér szál

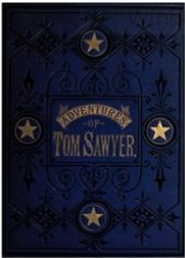
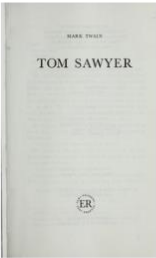
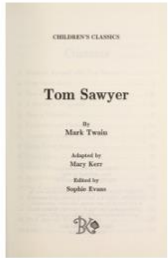

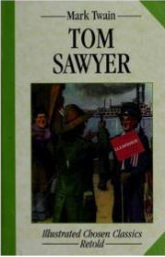
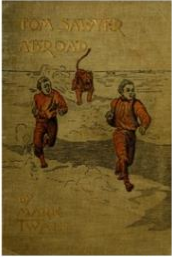
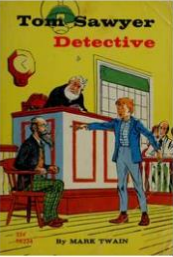

- Mindig aszinkron módon indul el



Könyvtárház applikáció

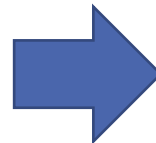
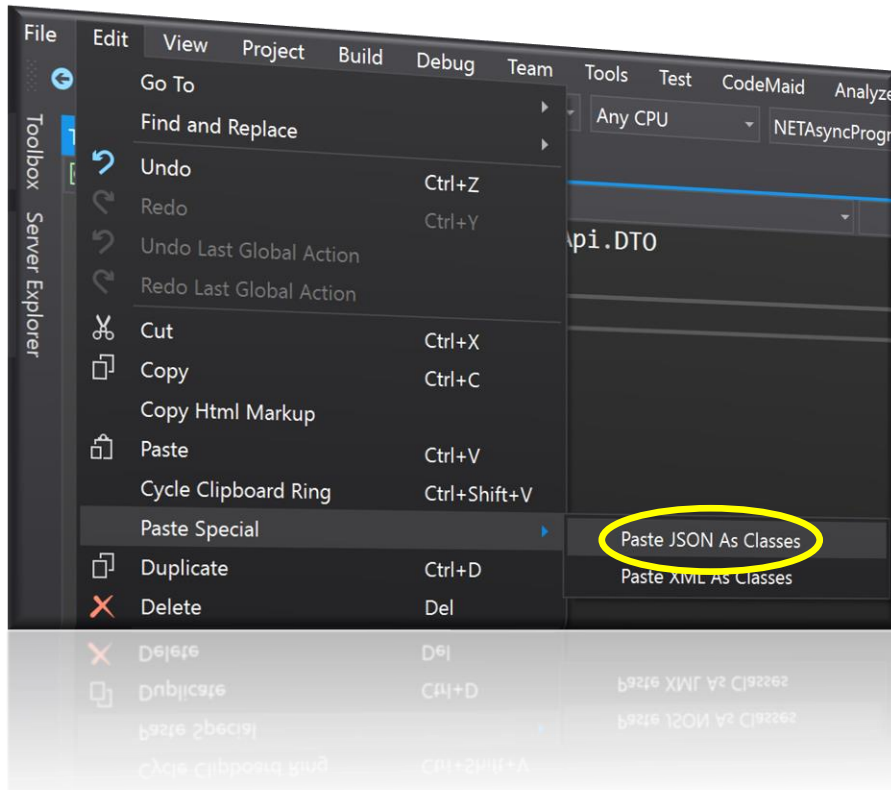
Book Store © DVSE RO - Attila Aszalos

Search text:

<input type="checkbox"/>  Adventures of Tom Sawyer Mark Twain	<input checked="" type="checkbox"/>  Tom Sawyer Mark Twain	<input checked="" type="checkbox"/>  Tom Sawyer Mary Kerr	<input type="checkbox"/>  Tom Sawyer June Edwards
<input type="checkbox"/>  Tom Sawyer Mark Twain	<input type="checkbox"/>  Tom Sawyer abroad Mark Twain	<input type="checkbox"/>  Tom Sawyer, Detective Mark Twain	<input type="checkbox"/>  Tom Sawyer Abroad/Tom Sawyer Detective Mark Twain

OPEN LIBRARY API használata C#-ból (1)

- DTO-k (Data Transfer Object) létrehozása



```
internal class SearchResponse
{
    public int start { get; set; }
    public int num_found { get; set; }
    public int numFound { get; set; }
    public SearchResultItem[] docs { get; set; }
}

internal class SearchResultItem
{
    public string title_suggest { get; set; }
    public string[] publisher { get; set; }
    public string[] isbn { get; set; }
    public string title { get; set; }
    public string[] author_name { get; set; }
    public string key { get; set; }
    public string[] text { get; set; }
    public string[] author_key { get; set; }
    public string type { get; set; }
    public string cover_edition_key { get; set; }
    public string[] subject { get; set; }
    ...
}
```



LIBRARY API használata C#-ból (2)

- GET HTTP kérés küldése:

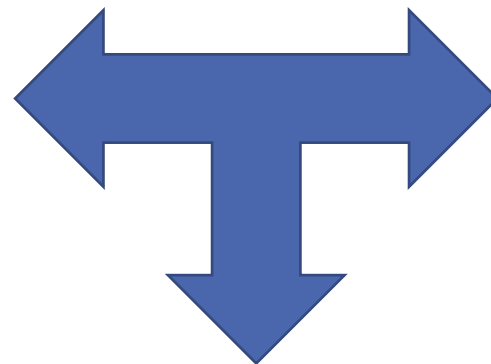
```
string address = "http://openlibrary.org/search.json?q=the+lord+of+the+rings";  
WebClient client = new WebClient();  
string responseText = client.DownloadString(address);
```



```
{  
  "start": 0,  
  "num_found": 86,  
  "numFound": 86,  
  "docs": [  
    {  
      "title_suggest": "The Lord of the Rings",  
      "edition_key": [  
        "OL9177076M",  
        "OL7883890M",  
        "OL10681058M",  
        "OL21217116M",
```

responseText

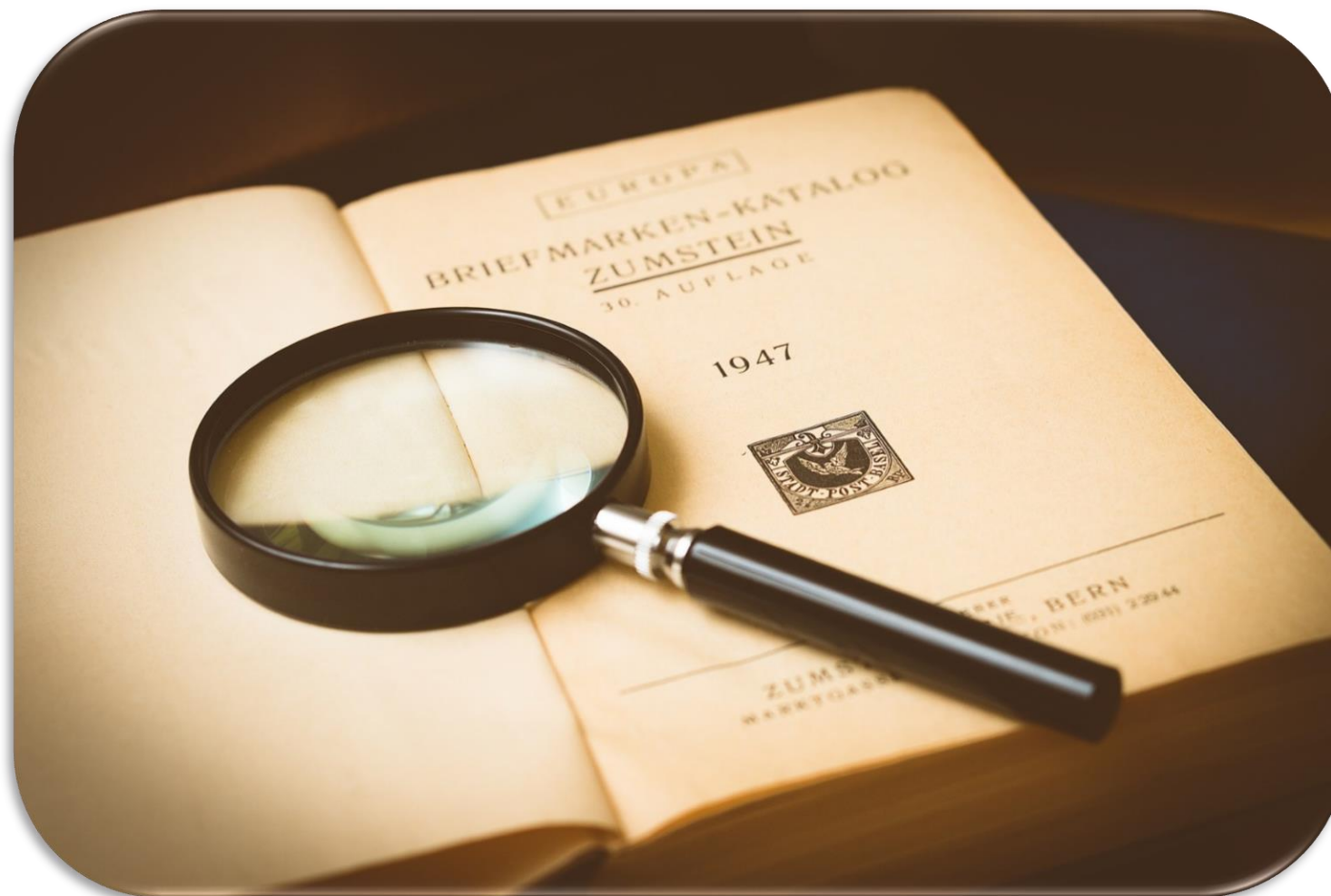
Deszerializálás



```
response  
  docs  
  numFound 86  
  num_found 86  
  start 0  
  response.docs  
    [0]  
      author_alternative_name {string[4]}  
      author_key {string[1]}  
      author_name {string[1]}  
      contributor {string[10]}  
      cover_edition_key "OL1532643M"  
      cover_i 8314541  
      ebook_count_i 7  
      edition_count 113  
      edition_key {string[113]}  
      first_publish_year 1950  
      first_sentence null  
      has_fulltext true  
      ia {string[7]}  
      ia_box_id {string[7]}
```

```
SearchResponse response = JsonConvert.DeserializeObject<SearchResponse>(responseText);
```

Könyvtáruhá – Demo 01



Könyvtárház – Demo 01 – Következtetés

- Hogyan írtuk át a szinkron működést aszinkron működéssé, BackgroundWorker-t használva?
 - **Callback:**
 - Egy olyan metódus (függvény) ami akkor hívódik meg, amikor egy aszinkron művelet befejeződött
 - Visszatérítési érték átalakul **callback** paraméterre
 - Mi is az a „**callback hell**”?

```
public void Search(string searchText, Action<List<Book>> callback)
{
    api.Search(searchText, (SearchResponse response) =>
    {
        SearchResponseMapper.Map(response, (List<Book> models) =>
        {
            callback(models);
        });
    });
}
```

Könyvárúház – Demo 01



Könyvárúház – Demo 01 – Következtetés

- Hogyan írjuk át a szinkron működést aszinkron működéssé, Task-ot használva?
 - A metódust async-nak jelöltük
 - Átalakult a visszatérítési érték Task-á
 - Az async metódust await-el hívtuk meg
 - Ami nem támogatja a Task API-t Task.Run(...)-al futtattuk



Task – Egyszerűsített használati útmutató

```
public async Task<SearchResponse> SearchAsync(string searchText)
{
    string address = ConstructSearchUrl(searchText);

    HttpClient client = new HttpClient();

    string responseText = await client.GetStringAsync(address);
    SearchResponse response = await Task.Run(() =>
    JsonConvert.DeserializeObject<SearchResponse>(responseText));

    return response;
}
```


Task – Vigyázat, nem mindegy...



- Await nélkül:

```
Debug.WriteLine("Before Task.Run");  
Task.Run(() => Debug.WriteLine("Task.Run action"));  
Debug.WriteLine("After Task.Run");
```



Before Task.Run
After Task.Run
Task.Run action

- Await-el:

```
Debug.WriteLine("Before Task.Run");  
await Task.Run(() => Debug.WriteLine("Task.Run action"));  
Debug.WriteLine("After Task.Run");
```



Before Task.Run
Task.Run action
After Task.Run

- Task.Run(...)

- Számítási műveletekre ajánlott használni

Fontosabb Task metódusok és property-k

Használatosak

ContinueWith()

Exception

IsCanceled

IsCompleted

IsFaulted

Jó tudni róluk

Result

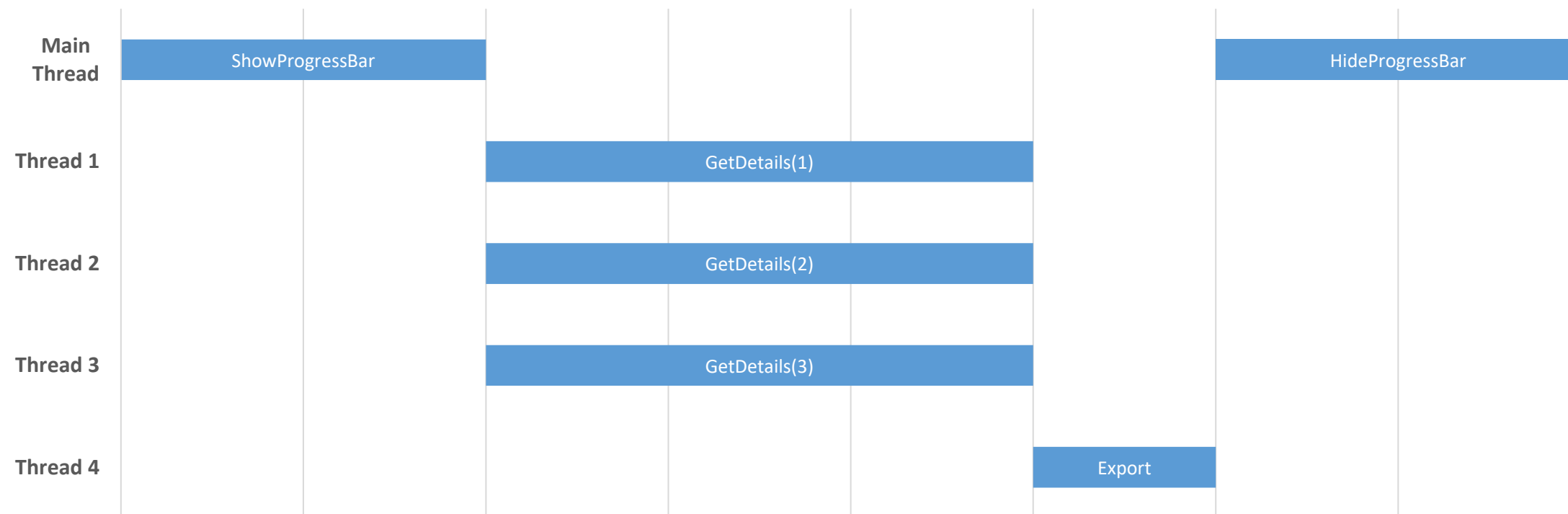
Wait()

Start()

RunSynchronously()

Könyvtárház applikáció

- Export műveletsorrendiség:



Könyvárúház – Demo 02



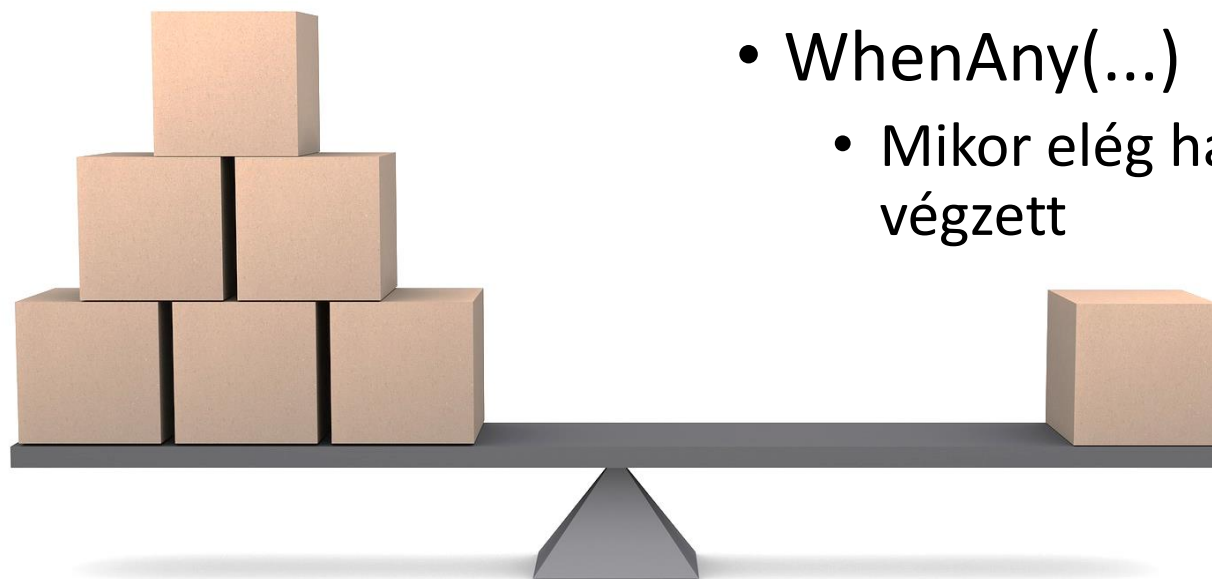
Könyvtárház – Demo 02 – Következtetés

BackgroundWorker

- Számláló bevezetése, hogy tudjuk mindenki végzett

Task

- WhenAll(...) metódus használata
- WhenAny(...)
 - Mikor elég ha csak egy Task végzett



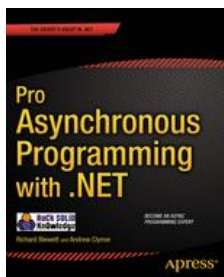
Egyéb információk



- CancellationTokenSource
 - <https://docs.microsoft.com/en-us/dotnet/standard/parallel-programming/task-cancellation>



- TaskCompletionSource
 - <https://docs.microsoft.com/en-us/dotnet/api/system.threading.tasks.taskcompletionsource-1?view=netframework-4.7.2>



- Könyv
 - <https://www.apress.com/gp/book/9781430259206>

Összefoglaló



Szinkron / aszinkron metódus hívás

Számítási művelet / Erőforrás hozzáférési művelet

UI elemekhez csak UI szál férhet hozzá

Szinkron forráskód átalakítása aszinkron forráskóddá

- BackgroundWorker használat
- Callback fogalma
- Task használat

Párhuzamos műveletek végrehajtása

Task megszakítása



Kérdések...



Köszönöm a figyelmet!