

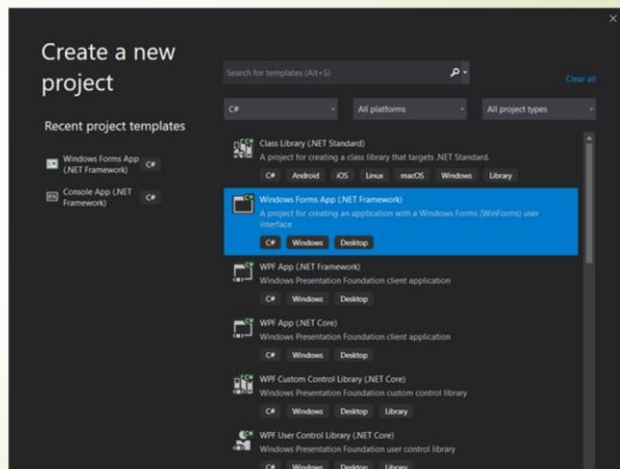


Excel fájlkezelés C#-ban

We

Új projekt létrehozása

- Visual Studio
 - C# .NET Framework létrehozása
 - Console vagy Windows Form

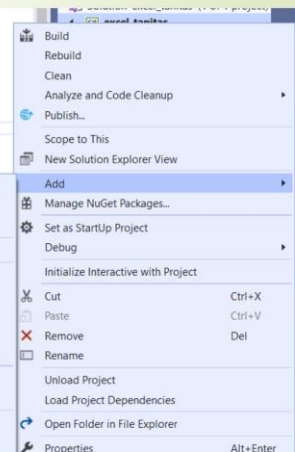
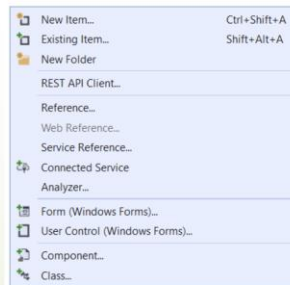


.NET Framework C# projekt létrehozása Visual Studio-ban.

Header és references

- Add -> Reference, Microsoft Excel ... object library pipa -> OK
- using Microsoft.Office.Interop.Excel;
- using System.Text.RegularExpressions;
- using System.IO;

Microsoft DirectX Transform Core Type Library	1.1
Microsoft DirectX Transforms Image Transforms...	1.1
Microsoft Disk Quota 1.0	1.0
<input checked="" type="checkbox"/> Microsoft Excel 15.0 Object Library	1.8
Microsoft External Item Picker	1.0
Microsoft Fax Service Extended COM Type Library	1.0
Microsoft Feeds 2.0 Object Library	2.0
Microsoft File Tracing Automation Library 1.0	1.0



Add jobb klikk, add Reference, Microsoft Excel object ... library pipa, aztán OK
Összes használt namespace:

- using System;
- using System.Collections.Generic;
- using System.Linq;
- using System.Text;
- using System.Threading.Tasks;
- using Microsoft.Office.Interop.Excel;
- using System.IO;
- using System.Text.RegularExpressions;

Alapok – megnyitás, értékadás

```
_Application excel = new Application();  
  
Workbook wb;  
Worksheet ws;  
  
wb = excel.Workbooks.Open(@"C:\mappa\..\excelneve.xlsx");  
ws = wb.Worksheets[1]; //1. worksheet, 1-től számoz
```

Excel file-ok megnyitására példányosítani kell:

- `_Application excel = new Application();`

Létezik workbook, illetve worksheet

- `Workbook wb;`
- `Worksheet ws;`

Egyszerű értékadás:

- `wb = excel.Workbooks.Open(@"C:\mappa\...\excelneve.xlsx");`
- `ws = wb.Worksheets[1];` a wb projekt 1. worksheet-e lesz, 1-től kezdi a számozást, nem 0-tól

Alapok – bezárás, mentés

```
wb.Close(); //rákérdez, hogy elmentse-e (pop up)
wb.Close(0); //nem ment rá a file-ra
wb.Close(1); //ráment

wb.SaveCopyAs(@"C:\mappa\...\ujexcelneve.xlsx"); //mentés másként
```

Fájl bezárása:

- `wb.Close()`; rákérdez, hogy rámentsen-e, ha egymás után több fájlt kezelünk akkor idegesítő lehet a popup
- `wb.Close(0)`; nem ment rá
- `wb.Close(1)`; rámenti

Ha esetleg crash-elne a program, és nem jut el a bezárásig, akkor az excel a háttérben nyitva marad, ezt le kell kezelni.

readInFiles()

```
public void readInFiles(){
    var allFiles = Directory.EnumerateFiles(@"C:\mappa\...\célmappa");
    Regex fileFilter = new Regex(@"^(.)*?\\[^\~$]+.xlsx$");
    foreach (string file in allFiles){
        if (fileFilter.IsMatch(file)) {
            wb = excel.Workbooks.Open(file);
            mergeSheets();
            ws = wb.Worksheets[1];
            writeOutData();
            wb.Close(0);
        } } }
```

A következőkben végigmegyünk egy mappa tartalmán, megnyitjuk egymás után az excel file-okat, átrakjuk az összes értéket az első worksheet-re, átállítjuk a ws változót az első worksheet-re, hogy ne kelljen mindig kiírni, kiírjuk a kellő adatokat, majd mentés nélkül bezárjuk a file-t.

Ha simán akarnál végigmenni egy mappa tartalmán és az összes excel file-t egyenként megnyitni, csinálni vele valamit, majd bezárni, akkor error-t adna ki, mivel az excel a megnyitás előtt csinál mindegyik excel fájlhoz egy üres temporary verziót, ~\$fájlnév.xlsx névvel, ezért csinálunk egy regex-et, ami kiszűri ezeket:

- `^(.)*?` - a fájl mappaszerkezete
- `\\` - elválasztás
- `[^\~$]+` - ne legyen benne ~, \$
- `.xlsx$` - xlsx legyen a vége

MergeSheets() – hova másolunk?

```
public void mergeSheets(){ //minden sheetet átrakja a legelsőre
    for (int i = 2; i < wb.Worksheets.Count+1; i++){
        int goalSheetRow = getRowNum(wb.Worksheets[1]);
        string goalSheetStart = "A" + (goalSheetRow + 1);

        int sourceSheetRow = getRowNum(wb.Worksheets[i]);
        int sourceSheetColumn = getColNum(wb.Worksheets[i]);
        string sourceSheetEnd=GetExcelColumnName(sourceSheetColumn)
                                + sourceSheetRow;

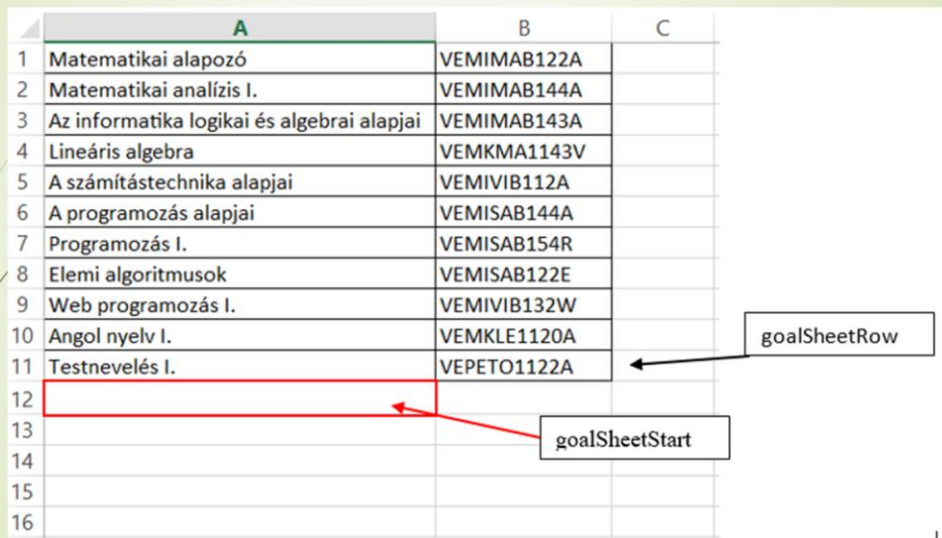
        wb.Worksheets[i].Range["A1", sourceSheetEnd].
            Copy(wb.Worksheets[1].Range[goalSheetStart]);
    }
}
```

Hogy könnyebb legyen a workbook-ból kiolvasni, ne kelljen ha egy worksheet végére értünk átállni a következő worksheet-re, ezért az összes worksheet tartalmát átmásoljuk az elsőre, hogy csak a worksheet[1]-et kelljen kezelni (később ws változó).

Az 1. sheet jelenlegi adatait:

- `int goalSheetRow = getRowNum(wb.Worksheets[1]);`
ez az első sheet utolsó használt sora, ehhez majd hozzá kell adni egyet, és hozzá adni az A betűt, ide fogjuk másolni az i.sheetet
- `string goalSheetStart = "A" + (goalSheetRow + 1);`

MergeSheets() – hova másolunk?



	A	B	C
1	Matematikai alapozó	VEMIMAB122A	
2	Matematikai analízis I.	VEMIMAB144A	
3	Az informatika logikai és algebrai alapjai	VEMIMAB143A	
4	Lineáris algebra	VEMKMA1143V	
5	A számítástechnika alapjai	VEMIVIB112A	
6	A programozás alapjai	VEMISAB144A	
7	Programozás I.	VEMISAB154R	
8	Elemi algoritmusok	VEMISAB122E	
9	Web programozás I.	VEMIVIB132W	
10	Angol nyelv I.	VEMKLE1120A	
11	Testnevelés I.	VEPETO1122A	
12			
13			
14			
15			
16			

Az 1. sheet jelenlegi adatait:

```
int goalSheetRow = getRowNum(wb.Worksheets[1]); // itt ez 11
```

ez első sheet utolsó használt sora, ehhez majd hozzá kell adni egyet, és hozzá adni az A betűt, ide fogjuk másolni az i.sheetet

```
string goalSheetStart = "A" + (goalSheetRow + 1); //itt ez „A”+(11+1)=A12
```


MergeSheets() – honnan másolunk?

```
public void mergeSheets(){ //minden sheetet átrakja a legelsőre
    for (int i = 2; i < wb.Worksheets.Count+1; i++){
        int goalSheetRow = getRowNum(wb.Worksheets[1]);
        string goalSheetStart = "A" + (goalSheetRow + 1);

        int sourceSheetRow = getRowNum(wb.Worksheets[i]);
        int sourceSheetColumn = getColNum(wb.Worksheets[i]);
        string sourceSheetEnd=GetExcelColumnName(sourceSheetColumn)
                                + sourceSheetRow;

        wb.Worksheets[i].Range["A1", sourceSheetEnd].
            Copy(wb.Worksheets[1].Range[goalSheetStart]);
    }
}
```

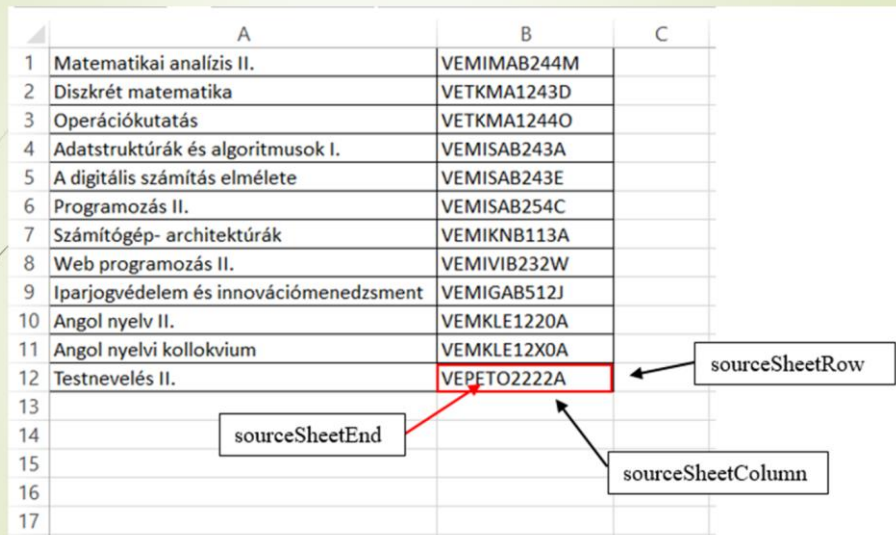
Az i. sheet adatai:

- int sourceSheetRow = getRowNum(wb.Worksheets[i]);
- int sourceSheetColumn = getColNum(wb.Worksheets[i]);

Kombináljuk a fenti 2 adatot, hogy megkapjuk az utolsó használt cella nevét, itt pl B+12=B12

- string sourceSheetEnd = GetExcelColumnName(sourceSheetColumn) + sourceSheetRow;

MergeSheets() – honnan másolunk?



	A	B	C
1	Matematikai analízis II.	VEMIMAB244M	
2	Diszkrét matematika	VETKMA1243D	
3	Operációkutatás	VETKMA1244O	
4	Adatstruktúrák és algoritmusok I.	VEMISAB243A	
5	A digitális számítás elmélete	VEMISAB243E	
6	Programozás II.	VEMISAB254C	
7	Számítógép- architektúrák	VEMIKNB113A	
8	Web programozás II.	VEMIVIB232W	
9	Iparjogvédelem és innovációmenedzsment	VEMIGAB512J	
10	Angol nyelv II.	VEMKLE1220A	
11	Angol nyelvi kollokvium	VEMKLE12X0A	
12	Testnevelés II.	VEPETO2222A	
13			
14			
15			
16			
17			

Az i. sheet adatait:

- `int sourceSheetRow = getRowNum(wb.Worksheets[i]); //itt ez 12`
- `int sourceSheetColumn = getColNum(wb.Worksheets[i]); //itt ez 2`

Kombináljuk a fenti 2 adatot, hogy megkapjuk az utolsó használt cella nevét, itt pl. B+12=B12

- `string sourceSheetEnd = GetExcelColumnName(sourceSheetColumn) + sourceSheetRow;`

MergeSheets() – Copy

```
public void mergeSheets(){ //minden sheetet átrakja a legelsőre
    for (int i = 2; i < wb.Worksheets.Count+1; i++){
        int goalSheetRow = getRowNum(wb.Worksheets[1]);
        string goalSheetStart = "A" + (goalSheetRow + 1);

        int sourceSheetRow = getRowNum(wb.Worksheets[i]);
        int sourceSheetColumn = getColNum(wb.Worksheets[i]);
        string sourceSheetEnd=GetExcelColumnName(sourceSheetColumn)
            + sourceSheetRow;

        wb.Worksheets[i].Range["A1", sourceSheetEnd].
            Copy(wb.Worksheets[1].Range[goalSheetStart]);
    }
}
```

Végül kombináljuk a következő függvény segítségével:

- `wb.Worksheets[i].Range["A1", sourceSheetEnd].Copy(wb.Worksheets[1].Range[goalSheetStart]);`

Ennek a jelentése:

- `melyikmunkalapból.Range[mettől, meddig].Copy(melyikmunkalapba.Range[honnantólkezdve]);`



getRowNum()

```
public int getRowNum(Worksheet sheet){  
    //visszaadja az első teljesen üres sor előtti sor sorszámát  
    int result = sheet.Cells.Find("",  
        System.Reflection.Missing.Value,  
        System.Reflection.Missing.Value,  
        System.Reflection.Missing.Value,  
        XlSearchOrder.xlByRows,  
        XlSearchDirection.xlPrevious, false,  
        System.Reflection.Missing.Value,  
        System.Reflection.Missing.Value).Row;  
    return result;  
}
```

getRowNum() függvény kifejtése, ami csak egy másik beépített függvényt meghív és visszaad egy számot, de kitettük külön függvénybe, hogy átláthatóbb legyen a kész program.

Fontos a végén a .Row, ezért adja vissza az utolsó nem üres sort.

getColNum()

```
public int getColNum(Worksheet sheet){  
    // visszaadja az első teljesen üres oszlop előtti oszlop  
    sorszámát  
    int result = sheet.Cells.Find("",  
        System.Reflection.Missing.Value,  
        System.Reflection.Missing.Value,  
        System.Reflection.Missing.Value,  
        XlSearchOrder.xlByColumns,  
        XlSearchDirection.xlPrevious, false,  
        System.Reflection.Missing.Value,  
        System.Reflection.Missing.Value).Column;  
    return result;  
}
```

Ugyanígy a getColNum() kifejtése, amit ugyanazt csinálja, de a végén egy .Column van.

Ez a két függvény megkeresi az az A1-től kezdve az első olyan sort/oszlopot, ami üres, szóval ha valamiféleképpen a táblázatunkban lenne pl. egy üres sor/oszlop, utána pedig megint adatok, akkor azt a részét levágná, erre figyelni kell. Mégis hasznos függvény, mivel egy excel worksheet-ben rengeteg sor és oszlop van, amin egy mezei for ciklussal sok idő lenne teljesen végigmenni és felesleges is, mivel üres a nagy része.

getExcelColumnName()

```
private string GetExcelColumnName(int columnNumber){ //szám alapján visszaadja a megfelelő betűt
    int dividend = columnNumber;
    string columnName = String.Empty;
    int modulo;

    while (dividend > 0){
        modulo = (dividend - 1) % 26;
        columnName = Convert.ToChar(65 + modulo).ToString() +
                      columnName;
        dividend = (int)((dividend - modulo) / 26);
    }
    return columnName;
}
```

Az GetExcelColumnName egy olyan függvény, amely visszaadja az adott szám alapján az abc megfelelő betűjét, pl. első oszlop, azaz 1 -> A betű, 2 -> B...

Használt forrás: <https://stackoverflow.com/questions/181596/how-to-convert-a-column-number-e-g-127-into-an-excel-column-e-g-aa>

writeOutData()

```
public void writeOutData()
{
    int lastRow = getRowNum(ws);
    for (int currentRow = 1; currentRow < lastRow+1; currentRow++)
    {
        if (ws.Cells[currentRow, 1].Value2 != null &&
            ws.Cells[currentRow, 2].Value2 != null)
        {
            Console.WriteLine(ws.Cells[currentRow, 1].Value2+" "+
                               ws.Cells[currentRow, 2].Value2);
        }
    }
}
```

Lekérjük, hogy az adott excel 1.sheetje hol ér véget, végigmegyünk rajta, megnézzük, hogy az adott sor 1. és 2. értéke nem üres, majd kiíratjuk őket a console-ra.

Az ellenőrzés azért fontos, mert ha pl. meg szeretnénk hívni rá a toString() metódust, hogy belerakjuk egy string változóba, akkor error-t dob ki az üres érték miatt, ezért jó megszokás, ha mindig megnézzük, nem üres-e.

Egy cellára meg tudjuk hívni a Text, Value és Value2 attribútumot is, ezek mind picit máshogy adják vissza a tárolt értéket, általában a Value2-t használják, az a legbiztonságosabb.

Ha egy cellaegyesített cellából szeretnénk értéket kizsedni (pl. A1, B1 és C1 össze van vonva), akkor az az érték az 1. cellaegyesített cellában (A1) található, a többi üres (B1 és C1).



Köszönjük a figyelmet!

Készítette: Lőrincz Beáta, Szabó Alexandra, Bittmann Rebeka