Visual Basic 2005 Express Edition

A VISUAL STUDIO HASZNÁLATA

Összeállította: Juhász Tibor (2008)

NEM LEKTORÁLT VÁLTOZAT

A dokumentumban előforduló hibákat és egyéb megjegyzéseket kérjük a juhaszt@zmgzeg.sulinet.hu címen jelezni.

Az alábbiakban összefoglaljuk a Visual Studio használatára vonatkozó legfontosabb tudnivalókat. Nem célunk az integrált fejlesztői környezet részletes ismertetése, nincs is módunk ennek az összetett és bonyolult rendszernek a szisztematikus áttekintésére. Csupán a Visual Basic programok írásához, a projektek kezeléséhez szükséges alapismeretekre térünk ki. Javasoljuk az olvasónak, hogy először nézze át az *Első lépések* című dokumentumot.

A Visual Basic programok szerkezete

A konzolalkalmazások szerkezete

Konzolalkalmazásaink egyetlen modulból állnak. A modult egy kezdő és egy záró utasítás határolja (*Module … End Module*). A kezdő utasítás után írjuk a modul nevét (általában *Module1*):

```
Module modulnév
  …
End Module
```

A modul változódeklarációkat és eljárásokat tartalmazhat. Az eljárásokon kívül deklarált változók globális hatókörűek a modul eljárásaira nézve.

```
Module modulnév
  a modulra nézve globális változók deklarációi
  eljárások definíciói
End Module
```

Az eljárásokat egy kezdő és egy záró utasítás határolja (*Sub … End Sub*)[1]. A kezdő utasítás után áll az eljárás neve, majd zárójelben az eljárás paraméterei:

```
Sub eljárásnév(paraméter, paraméter, …)
  …
End Sub
```

A zárójelpárt akkor is célszerű kitenni, ha az eljárásnak nincsenek paraméterei.

Egy konzolalkalmazás futtatásakor a *Module1* modul *Main* (fő) nevű eljárása kerül végrehajtásra, ennek kötelezően szerepelnie kell a forráskódban.[2] Egy új konzolalkalmazás létrehozásakor ezek az egységek a szükséges kezdő és záró utasításokkal együtt automatikusan bekerülnek a forráskódba:

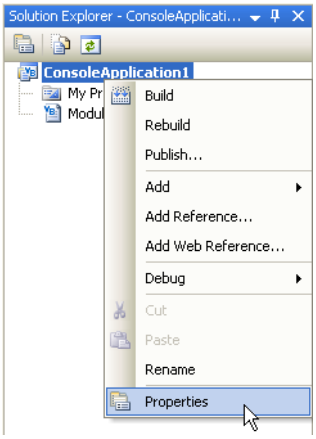
```
Module Module1

  Sub Main()

  End Sub

End Module
```

[1] A *Sub* a szubrutin (eljárás, alprogram) angol nyelvű rövidítése.

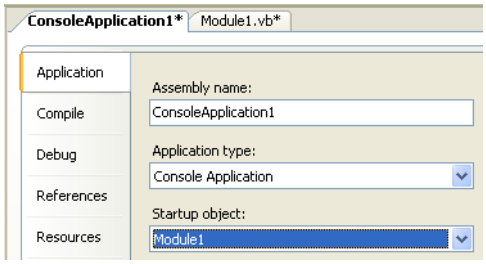
[2] A Windows-alkalmazások is rendelkeznek *Main*() eljárással, de ezt a fordítóprogram automatikusan létrehozza. A *Main*() eljárás nyitja meg például a programablakot.

A modulnév módosítása

A modul nevét a forráskódban módosíthatjuk. Ebben az esetben azonban az alkalmazás tulajdonságainál meg kell adnunk az új azonosítót. A tulajdonságok a megoldástallózóban a jobb egérkattintásra megnyíló helyi menüből érhetők el (*Properties*). A *Resources*/*Startup object* legördülő listájából válasszuk ki az új nevet vagy a *Main*() eljárást![3]



Az alkalmazás tulajdonságainak a megnyitása



A Startup object az alkalmazás tulajdonságai között

A Windows-alkalmazások szerkezete

Windows-alkalmazásaink egyetlen ablak (form, űrlap) osztálydefinícióját tartalmazzák. Az osztálydefiníciót egy kezdő és egy záró utasítás határolja (*Class … End Class*). A *Class* után írjuk az osztály nevét (általában *Form1*):

```
Public Class osztálynév
  …
End Class
```

A *Public* kulcsszó arra utal, hogy az osztálydefinícióra más modulokból is hivatkozhatunk.[4]

Az osztálydefinícióban a változók deklarációit és az eljárások (köztük az eseménykezelő eljárások) definícióit helyezzük el. Az eljárásokon kívül deklarált változók alapértelmezés szerint *Private* hatókörűek, azaz csak az osztályon belül érhetők el.

```
Public Class osztálynév
  az osztályra nézve globális változók deklarációi
  eljárások/eseménykezelő eljárások definíciói
End Class
```

Megjegyezzük, hogy a *Form1* a .NET keretrendszerben definiált *Form* osztály leszármazottja. Rendelkezik a *Form* osztály tulajdonságaival és metódusaival. A tulajdonságok kezdőértékeit például a *Form* osztályból örökli. Eseménykezelő eljárásaink a leszármazott osztály metódusait bővítik. A fordítási, illetve futási időben megadott tulajdonságok a leszármazott osztály tulajdonságait módosítják.

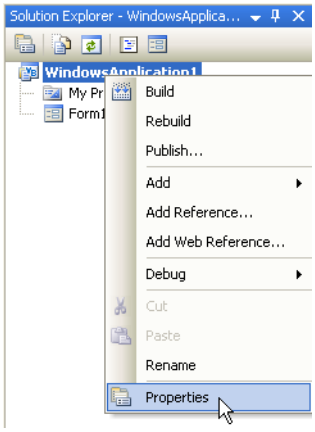
[3] A megoldástallózóban (*Solution Explorer*) a jobb egérkattintásra megnyíló helyi menü *Rename* parancsával a forráskódot tartalmazó fájlt nevezhetjük át.

[4] Pontosabban szólva a *Public* hatókör korlátozás nélküli elérhetőséget jelent.

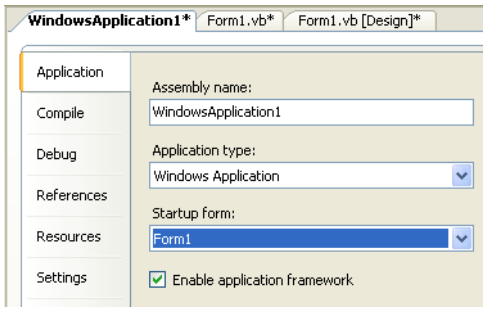
A kezdőablak beállítása

A programablaknál általában meghagyjuk a *Form1* azonosítót, de megadhatunk másik nevet is a forráskódban vagy a tulajdonságablakban (*Properties*).[5]

Ügyeljünk azonban arra, hogy a fordítóprogramnak ismernie kell a programablak (a program futtatásakor megnyíló ablak) azonosítóját. Ezt a Windows-alkalmazás tulajdonságainál állíthatjuk be, melyek a megoldástallózóban a jobb egérkattintásra megnyíló helyi menüből érhetők el (*Properties*). A tulajdonságok között a *Resources*/*Startup form* legördülő listájából választhatjuk ki a program indításakor megnyíló ablakot. Ennek általában csak akkor van szerepe, ha több ablakot is létrehozunk a projektben.



Az alkalmazás tulajdonságainak a megnyitása



A Startup form az alkalmazás tulajdonságai között

Ha a programablak azonosítóját a forráskódban vagy a tulajdonságablakban módosítjuk, akkor automatikusan módosul a *Startup form* értéke. Ha azonban töröljük a *Form1*-et, és egy másik alkalmazásból, esetleg más azonosítóval vesszük át az osztály definícióját, akkor nekünk kell beállítanunk a kezdőablak nevét!

[5] A megoldástallózóban (*Solution Explorer*) a jobb egérkattintásra megnyíló helyi menü *Rename* parancsával az osztálydefiníciót tartalmazó fájlt nevezhetjük át.

A projektek kezelése

Megoldások és projektek

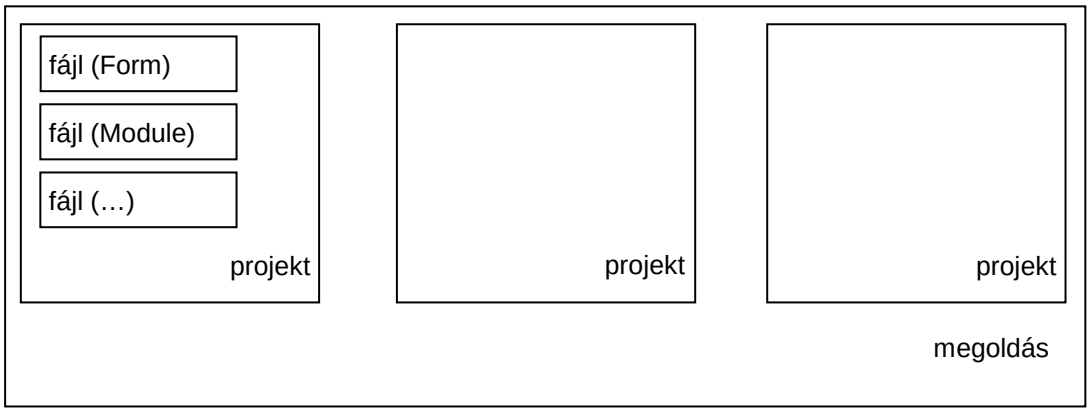
A Visual Studio összetett feladatok, bonyolult programok készítésére alkalmas. Ezeknél a fejlesztéseknél gyakran előfordul, hogy több programozó dolgozik egy-egy részfeladaton. A program sok részből áll, az egyes részeket egységes rendszerré kell összefogni.

A Visual Studio **megoldásnak** (*solution*) nevezi az összetett feladatot. A megoldás magában foglalja a készülő program összetevőit. Egy összetevő hozhatja létre a képernyőn megjelenő ablakot, vezérelheti a felhasználóval történő kommunikációt. Egy másik összetevő tarthatja a kapcsolatot egy adatbázis- vagy webszerverrel stb.

A megoldás összetevőit projekteknek hívjuk. A megoldás egy vagy több projektből áll. A **projekt** az adott alkalmazáshoz szükséges fájlokat tartalmazza. Ilyen fájl írhatja le például a program futtatásakor a képernyőn megjelenő ablak tulajdonságait. Egy másik fájlban helyezhetjük el az eljárások forráskódját vagy a képernyőn megjelenő képeket stb. A megoldáson belül az egyes projektek akár más és más programozási nyelven készülhetnek.

Az ablakok megjelenését és viselkedését leíró fájlokat általában űrlapoknak (*Form*), az eljárásokat tartalmazó fájlokat pedig moduloknak (*Module*) nevezzük.

A Visual Studio adminisztrációs fájlokban tárolja a megoldás és a projektek jellemzőit. Az adminisztrációs fájlok a megoldások és projektek mentésekor jönnek létre.



Egy Visual Studio alkalmazás szerkezete

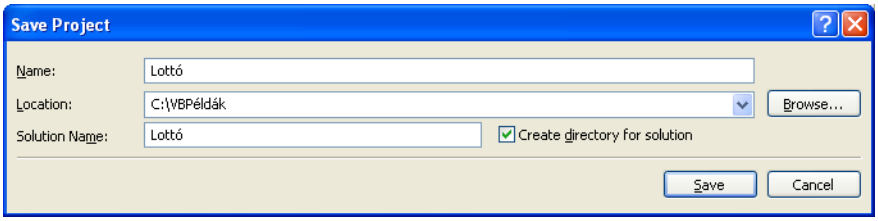
Új projekt létrehozása

Feladataink megoldásához általában elegendő lesz egyetlen projektből álló megoldást készítenünk. Ehhez a *File*/*New Project* parancsot használjuk. Válasszuk ki a megfelelő sablont (*Windows Application*, illetve *Console Application*). A *My Templates* csoportban az általunk előzőleg létrehozott sablonokat láthatjuk (lásd később). A létrehozásnál adjunk beszédes nevet (*Name*) a projektnek!

Amint létrejön a projekt, létrejön a projektet tartalmazó megoldás is. A megoldástallózó azonban csak a projektet mutatja.

A projekt mentése

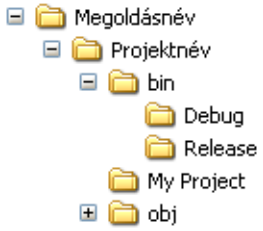
A munka során ne az egyedi fájlokat (modulokat), hanem a teljes projektet mentsük (*File*/*Save all*, , *Ctrl+Shift+S*)! Az első mentésnél adjunk beszédes nevet mind a projektnek (*Name*), mind a megoldásnak (*Solution Name*)! Figyeljünk oda a mentés helyére (*Location*)! Hozzunk létre saját mappát projektjeinknek, és ne fogadjuk el a Visual Studio által felajánlott elhelyezést! A *Create directory for solution* jelölőnégyzet segítségével hozassunk létre saját mappát a projektnek (illetve a megoldásnak)!



A projekt első mentésénél megjelenő ablak

A projektet a *File*/*Close project* paranccsal zárhatjuk be. A szokásokkal ellentétben a *Discard* feliratú parancsgombbal vethetjük el a módosításokat.

A projekt fájljai



A projekt mappaszerkezete

A Visual Studio a projekt mentésénél összetett mappaszerkezetet alakít ki a megadott elérési úton. A megoldással megegyező nevű mappában jönnek létre a megoldáshoz tartozó egyes projektek mappái.

A konzolalkalmazások forráskódja a *megoldásnév\projektnév* mappában helyezkedik el *.vb* kiterjesztéssel (például: Module1.vb). A Windows-alkalmazások forráskódja ugyanezen mappa két fájljában található. Az *osztálynév*.vb fájl az osztály definícióját tartalmazza a változódeklarációkkal és eljárásokkal együtt, míg az *osztálynév*.Designer.vb fájlban az ablak beállításait, szerkezetét tároljuk (például Form1.vb, illetve Form1.Designer.vb).

A *.vb* kiterjesztésű fájlok (a mappákban lévő sok más fájllal együtt) a Jegyzettömbbel olvasható szövegfájlok. Módosításukhoz azonban mindig a Visual Studio fejlesztői környezetét használjuk!

A projekt *bin* mappájában találjuk a futtatható *.exe* fájlokat. A *Debug* mappa *.exe* fájlja akkor jön létre, amikor a fejlesztői környezetben először adjuk ki a *Debug*/*Start Debugging* parancsot. Ez valójában nyomkövető-hibakereső futtatásnak felel meg.

A *Release* mappába kerülő *.exe* fájlt a *Build*/*Build* parancs kiadása hozza létre. Ez a parancs felel meg a klasszikus programozási nyelvek fordítási (*compile*) parancsának. A *Release* mappa *.exe* fájlja képviseli az elkészült programot. Ezt a fájlt futtathatjuk a fejlesztői környezet telepítése nélkül, ezt a fájlt másolhatjuk további számítógépekre stb.

Összetett alkalmazások esetén a program működéséhez további fájlokra van szükség. A *Build* menü *Publish* parancsával telepítőkészletet készíthetünk az alkalmazásunkhoz. Ezt a lehetőséget itt nem tárgyaljuk.

Projekt megnyitása

Meglévő projektet a *File* menü *Open Project* (*Ctrl+O*) parancsával nyithatunk meg. Ha az Intézőben duplán kattintunk a projekt mappájában lévő *.vbproj* fájlra, akkor a projekttel együtt megnyílik a fejlesztői környezet. Ugyanezt érjük el a megoldás mappájában lévő *.sln* fájllal is.

Egy program szerkesztéséhez, módosításához mindig a projektet nyissuk meg, ne pedig a modul forráskódját tartalmazó fájlt! A *File* menü *Open File* (📂) parancsa helyett válasszuk az *Open Project* (*Ctrl+O*) parancsot. A *File* menüben megtaláljuk az utoljára megnyitott projektek listáját (*Recent Projects*). Itt se a *Recent Files* listát használjuk a megnyitáshoz!

A megoldástallózó a rejtett fájlok kivételével megmutatja a megnyitott projekthez tartozó fájlokat. Ha egy projekt megnyitásánál nem látjuk kódszerkesztő vagy a tervezőablakot, akkor a megoldástallózóban a jobb egérgombbal kattintsunk a .vb kiterjesztésű fájlra, és válasszuk a *View Code*, illetve a *View Designer* parancsot. Dupla kattintással konzolalkalmazásnál a kódszerkesztő ablak, Windows-alkalmazásnál pedig a tervezőablak nyílik meg.

Sablonok használata

A Visual Studio projektek számos elemből állnak. Megtaláljuk közöttük az ablakok definícióit tartalmazó fájlokat, a forráskódot és egyéb, a projekt nyilvántartását, adminisztrációját szolgáló fájlokat. Az összetett szerkezet miatt nem létezik a projektre vagy az összetevőkre vonatkozó *Mentés másként* parancs.[6] Ennek hiányát sablonok használatával pótolhatjuk. A projekt egyes összetevőit, vagy magát az egész projektet sablonként menthetjük. A sablonok tartalmazzák a mentésig elvégzett beállításokat, az elkészült forráskódot. Egy új projekt létrehozásánál kiválaszthatjuk az elmentett sablont, így visszakapjuk a sablonban tárolt beállításokat és forráskódot.

Saját sablon készítése

Sablonként a készülő (vagy már kész) projektet, illetve összetevőit menthetjük el. Ehhez válasszuk a *File*/*Export template* parancsot! A sablon létrehozása előtt a Visual Studio szükség esetén rákérdez a projekt mentésére.

A megjelenő varázsló *Choose Template Type* ablakában válasszuk ki a sablon típusát. *Project template* esetén a teljes projektből, *Item template* esetén pedig a következő ablakban kijelölhető elemből készítünk sablont.

Projektsablon készítése

A *Project template* választása estén a következő ablakban megadhatjuk a sablon nevét (*Template name*) és rövid leírását (*Template description*), ami az új projekt létrehozásánál tájékoztatja a felhasználót a sablon tartalmáról. A *Finish* gombbal zárjuk le a sablon létrehozásának folyamatát.

Ablaksablon készítése

Ha az elkészült ablakot és a változódeklarációkat, illetve eljárásokat tartalmazó osztálydefiníciót szeretnék sablonként menteni, akkor a varázsló *Choose Template Type* ablakában jelöljük be az *Item template* választógombot! A megjelenő ablakban válasszuk ki az elemet (például *Form1.vb*). A következő, *Select Item References* ablakban adhatnánk meg a felhasznált hivatkozásokat, de erre nincs szükségünk, így kattintsunk a *Next* gombra.[7] A folytatás innen kezdve megegyezik a projektsablon készítésével.

Megjegyezzük, hogy ha ékezetes karaktereket használunk a sablon nevében, akkor figyelmeztető üzenetet kapunk a korlátozásokról, de nem kell vele foglalkoznunk.[8]

A Visual Studio a sablon mentése után – hacsak nem tiltjuk le – megnyitja a

C:\Documents and Settings*felhasználó*\Dokumentumok\Visual Studio 2005\My Exported Templates

mappát. Itt látjuk az elmentett sablonokat, melyeket *.zip* fájlok tartalmazzák. A fájlokat egy fájlkezelővel (például az Intézővel) a szokásos módon adminisztrálhatjuk (másolás, törlés stb.).

Saját sablon alkalmazása

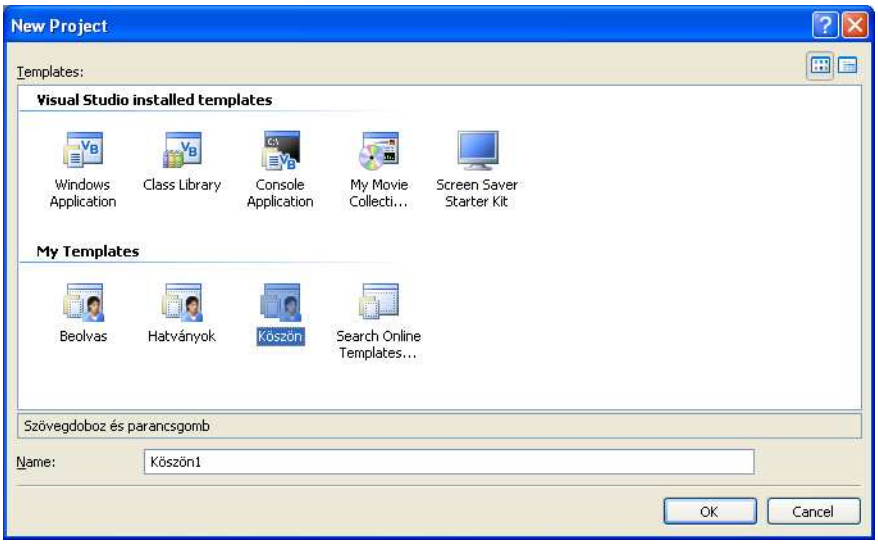
Az elkészült projektsablonokat egy új projekt létrehozásánál használhatjuk fel. A *File*/*New Project* parancsra megnyíló ablakban a *My Templates* csoportból választhatjuk ki sablonjainkat.

---

[6] A *File*/*Save … As* parancs a projekthez tartozó fájlt nevezi át, és a munka az új fájllal folytatódik (például ezen a néven jegyzi a program indításakor megnyíló ablakot).

[7] Ezt az opciót csak akkor kell használnunk, ha a projekt készítése során alkalmaztuk az *Add Reference* parancsot.

[8] Az üzenet arra utal, hogy az operációs rendszer angol nyelvű változatának a fájlnév miatt gondjai lehetnek a *.zip* fájl kezelésével.

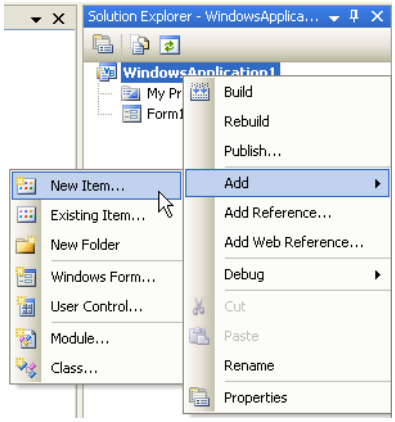


Saját sablon kiválasztása

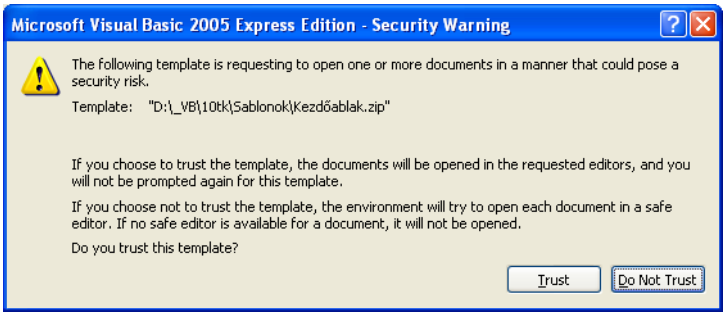
Ablaksablont természetesen csak már létező projektbe illeszthetünk be a megoldástallózóban a jobb egérkattintásra feltűnő helyi menü *Add/New Item* parancsával. A megjelenő ablak *My Templates* csoportjában találjuk saját sablonjainkat.

Az ablaksablon beillesztésekor biztonsági figyelmeztetést kapunk. Megbízható forrásból származó sablon esetén kattintsunk a *Trust* gombra.

Ablaksablon beillesztésekor legyünk figyelemmel a kezdőablak beállítására (lásd fent). Célszerű először törölni az eredeti *Form1* ablakot, majd átnevezni *Form1*-re a sablonból beillesztett ablakot. Így nincs szükség az alkalmazás *Startup form* tulajdonságának a módosítására.



Ablaksablon beillesztése

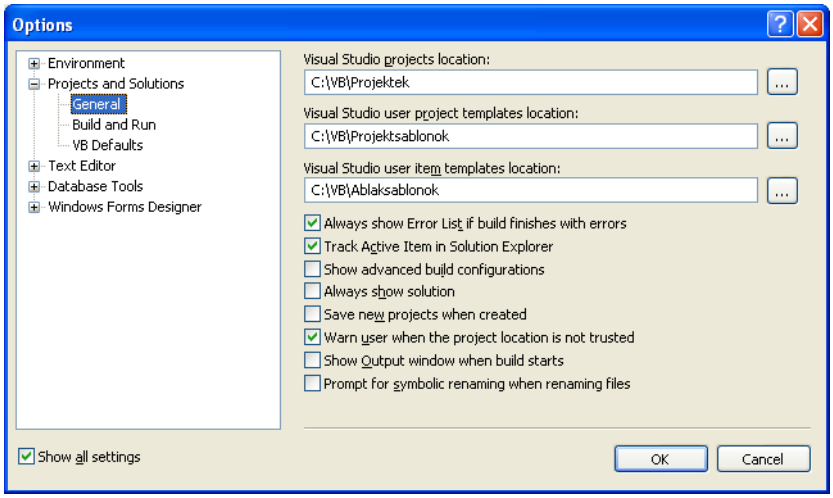


Biztonsági figyelmeztetés az ablaksablon beillesztésekor

Megjegyezzük, hogy az ablaksablon tartalmazza az ablak tulajdonságait és az osztálydefiníciót a változókkal és eljárásokkal együtt (továbbá az itt nem tárgyalt beágyazott erőforrásokat is).

A sablonfájlok helyének módosítása

Ha gyakran van szükségünk a sablonok kezelésére, más sablonok átvételére, törlésére, akkor célszerű egy könnyebben elérhető mappát kijelölni a sablonok tárolására. Ezt a *Tools*/*Options* menü *Project and Solutions*/*General* paneljén tehetjük meg. A *Visual Studio user project templates location* a projektsablonok, a *Visual Studio user item templates location* pedig az ablaksablonok helyét adja meg. Az itt elhelyezett sablonfájlok (*.zip* fájlok) listája jelenik meg egy új projekt létrehozásánál a *My Templates* csoportban.



Saját sablonok helyének a módosítása

Ugyanitt módosíthatjuk a projektek mentésének alapértelmezett mappáját (*Visual Studio projects location*).

Sajátos módon, az új helytől függetlenül a fenti felhasználói mappába is bekerülnek a sablonok, és egy sablon mentésénél ez az eredeti mappa nyílik meg. A mappa megnyitását letilthatjuk a sablonkészítő varázsló *Select Template Options* ablakában a *Display an explorer window on the output files folder* jelölőnégyzetének a kikapcsolásával.

Fájlok felvétele a projektbe

Egyszerűbb esetben nincs szükségünk sablon készítésére. Az ablak tulajdonságait és az osztálydefiníciót tartalmazó fájlokat egyszerűbben is felvehetjük a projektbe.

Készítsünk egy új projektet, mentsük, majd zárjuk be. Töröljük a szükségtelen *Form1.vb* és *Form1.Designer.vb* fájlt a projekt mappájából. Másoljuk át a szükséges *Form1.vb* és *Form1.Designer.vb* fájlt a projekt mappájába, a törölt fájlok helyére. A megnyitás után a szükséges beállításokat és forráskódot tartalmazó ablakot találjuk a projektben.

Ha az új fájlnevek nem egyeznek meg az eredeti (például *Form1*…) fájlnevekkel, akkor a fájlokat a megoldástallózóban a jobb egérkattintásra feltűnő helyi menü *Add*/*New Item* parancsával vehetjük fel a projektbe. Mindig az osztálydefiníciót tartalmazó *.vb* fájlt illesszük be, ne a tulajdonságokat tartalmazó *Designer* fájlt! A *Designer* fájlt fejlesztői környezet automatikusan átveszi.

Új fájlok beillesztése esetén szükség lehet az alkalmazás *Startup form* tulajdonságának a módosítására (lásd fent).

Megoldások (Solutions) alkalmazása

A Visual Studio az összetett alkalmazások projektjeit megoldásokba (*Solutions*) rendezi. A projektek mappái a megoldás mappájában helyezkednek el.

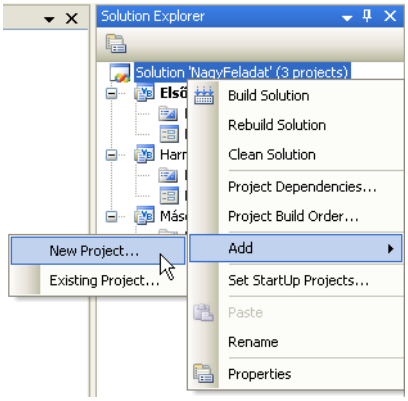
Ha létrehozunk egy új projektet, az önmagában is egy megoldást alkot. A különbség azonban akkor válik láthatóvá, amikor újabb projekteket veszünk fel a megoldásba.

> Bár a megoldás a nagyobb alkalmazások létrehozását segíti elő, az oktatás folyamatában felhasználhatjuk egy program különböző változatainak, módosításainak a tárolására, egyszerű áttekintésére.

Több projektből álló megoldás készítése

Az első projekt létrehozásakor létrejön egy megoldás is, amit a projekt mappaszerkezete jelez. A Visual Studio a projekt mentésénél a megoldás nevére szintén rákérdez (*Solution Name*).

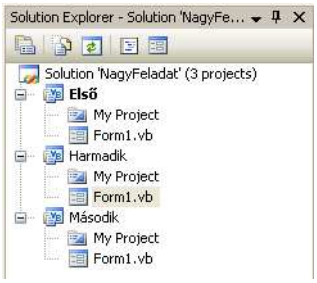
A második projektet a *File*/*Add*/*New project* paranccsal vehetjük fel a megoldásba. Ennek hatására kissé átalakul a megoldástallózó munkaablak. A lista gyökereként megjelenik a megoldás (*Solution*), amelyből kiágaznak az egyes projektek (a nevek ábécé sorrendjében). A továbbiakban a megoldástallózóban a jobb egérkattintásra előtűnő helyi menü *Add*/*New project* (illetve *Existing project*) parancsával is hozzáadhatunk projekteket a megoldáshoz.



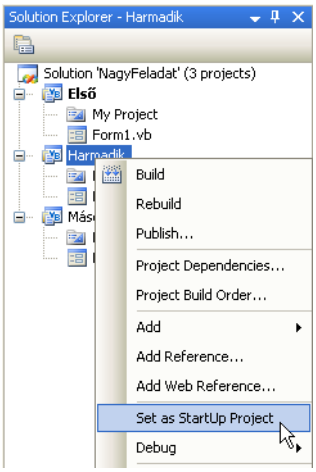
Újabb projekt felvétele a megoldásba

A kezdőprojekt kijelölése

A megoldásnak mindig van egy kezdőprojektje, amelyik elindul, ha a *Debug*/*Start Debugging* parancsot választjuk (▶, F5). A kezdőprojekt neve félkövér betűkkel jelenik meg a megoldástallózóban. Alapértelmezés szerint az elsőként létrehozott projekt lesz a kezdőprojekt.



A kezdőprojekt félkövér névvel látható



A kezdőprojekt beállítása

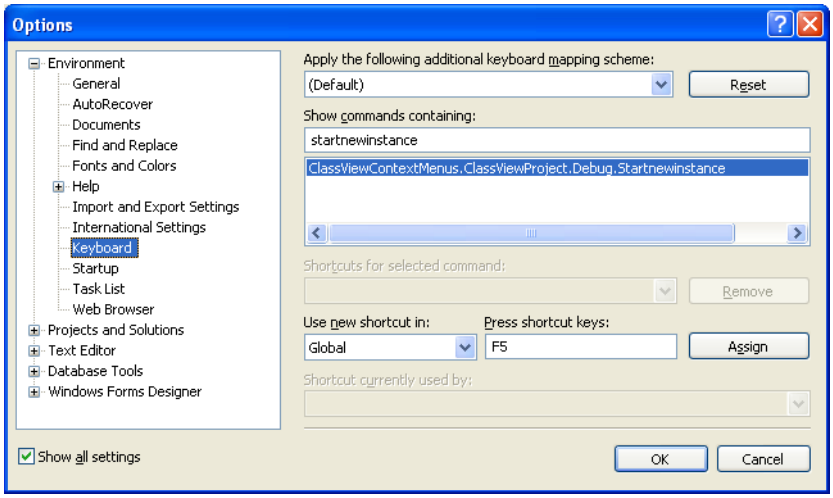
A kezdőprojektet a megoldástallózóban a jobb egérkattintásra előtűnő helyi menü *Set as Startup Project* parancsával állíthatjuk be.

Tetszőleges projekt futtatása

Ha nem akarjuk módosítani a kezdőprojektet, akkor a megoldás egy másik projektjét a megoldástallózóban a jobb egérkattintásra előtűnő helyi menü *Debug*/*Start New Instance* parancsával futtathatjuk.

Egyszerűbben érhetjük el ezt a célt, ha módosítjuk az *F5* funkcióbillentyűhöz rendelt parancsot. Nyissuk meg a *Tools*/*Options* ablakot, majd kapcsoljuk be a *Show all settings* jelölőnégyzetet.

Válasszuk ki a listából az *Environment*/*Keyboard* panelt. Gépeljük be a *Show commands containing* szövegdobozba a *startnewinstance* karaktersorozatot (szóközök nélkül!). Kattintsunk a *Press shortcut keys* szövegdobozra, és nyomjuk le az *F5* funkcióbillentyűt. Az *Assign* gombra kattintással végezzük el a hozzárendelést, majd az OK gombbal zárjuk be az *Options* ablakot.



Az F5 hozzárendelése a Start new instance parancshoz

A hozzárendelés után az *F5* billentyű hatására mindig a kiválasztott projekt indul el.

Megoldás felhasználása a programváltozatok tárolására

Mint említettük, a megoldás felhasználható az oktatás során egy program különböző változatainak, módosításainak a tárolására. Ehhez hozzunk létre egy projektet, majd készítsük el programunk kezdeti, első változatát.

Adjunk hozzá a megoldáshoz egy újabb projektet, és a megoldástallózóban tegyük át az első projekt *Form1* fájlját az újabb projektbe. Ha nem módosítottuk a fájlneveket, akkor megjelenik egy figyelmeztetés a létező fájlok (*Form1.vb*, illetve *Form1.Designer.vb*) felülírása miatt. Kapcsoljuk be az *Apply to all items* jelölőnégyzetet, majd kattintsunk a *Yes* gombra. Egy újabb figyelmeztetés arra utal, hogy a fájlokat egy másik projektből vettük át. Kattintsunk a *Yes to All* gombra.

A program fejlesztését az újabb projektben folytassuk, amely már tartalmazza eddigi fejlesztéseinket. Ezt a folyamatot ismételve tetszőleges számú változatot tárolhatunk.

Mivel a kódszerkesztő ablak tetején lévő fülek a megnyitott fájlok nevét mutatják, könnyebben tájékozódunk, ha a megoldástallózó munkaablakban módosítjuk az azonos, *Form1* fájlneveket. A fájlnév átírása azonban maga után vonja az osztálynév módosítását, ami miatt kézzel kell beállítanunk az újabb projekt *Startup form* tulajdonságát (lásd fent)!

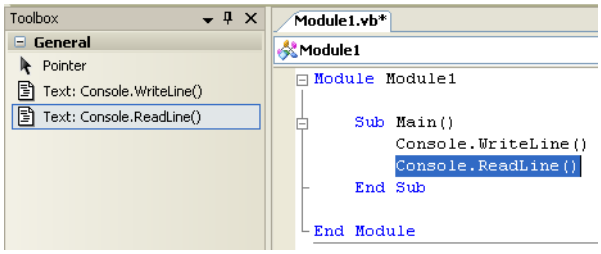
Ha nem egy *Form1* nevű fájlt teszünk át az új projektbe, akkor megmarad az eredeti fájl (nem írjuk felül). Ekkor kézzel kell törölnünk (például kijelölés után a *Delete* billentyűvel a *Solution Explorer* munkaablakban).

Az eszközkészlet használata

Kódrészletek tárolása az eszközkészletben

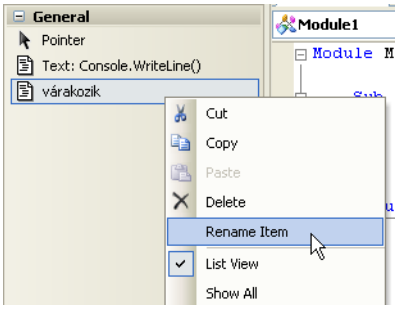
A forráskód elkészítése során sokszor be kell írnunk a *Console.WriteLine*, *Console.Readline* utasításokat. Bár az intelligens súgó segíti a kód begépelését, a munkát egyszerűsíthetjük az eszközkészletben tárolható kódrészletek segítségével.

Jelöljük ki a kód egy részletét (például a *Console.Readline* utasítást), majd az egér segítségével húzzuk rá az eszközkészlet (*Toolbox*) általános (*General*) paneljére. Ezzel tároltuk a kódrészletet. A továbbiakban a panelről szintén az egér segítségével a forráskód tetszőleges helyére beilleszthetjük. (A beillesztést dupla kattintással is elvégezhetjük.)



Kódrészlet elhelyezése az eszközkészletben

Az eszközkészletben elhelyezett kódrészletek listáját áttekinthetőbbé tehetjük, ha funkciójukra utaló nevet adunk az egyes elemeknek. Ezt a jobb egérkattintásra megjelenő helyi menü átnevezés (*Rename Item*) parancsával tehetjük meg.



Kódrészlet átnevezése az eszközkészletben

A súgó használata

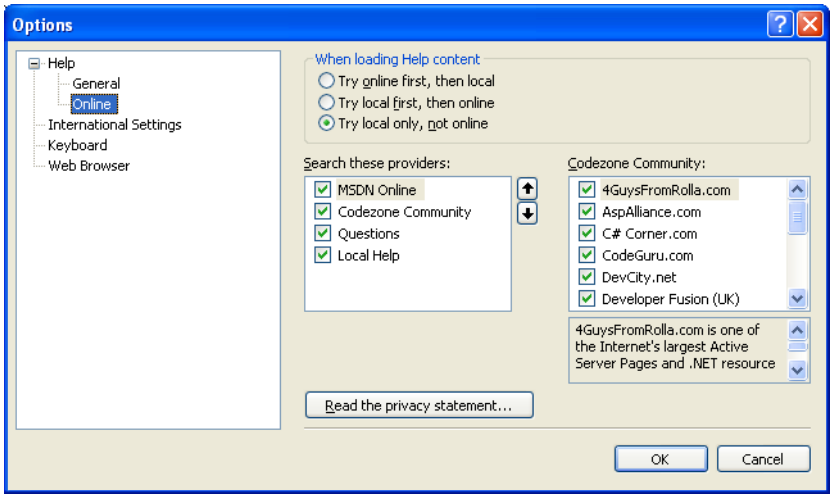
A Visual Studio részletes súgóval rendelkezik. A súgódokumentáció két részből áll. Az *MSDN Express Library* telepítéskor kerül fel a számítógépre. Ebben többek között olvashatunk egy angol nyelvű bevezetést a Visual Basic használatába (Visual Basic Guided Tour), egy programozói kézikönyvet (Visual Basic Programming Guide), illetve szerepel benne a nyelv teljes leírása (Reference).

A dokumentáció online része a Microsoft webhelyén érhető el (MSDN Online). Az online súgóban frissítéseket, aktuális információkat és további kiegészítéseket találunk a Visual Basichez.

A súgót a szokásos módon, az F1 funkcióbillentyűvel vagy a *Help*/*Contents* parancs segítségével érhetjük el.

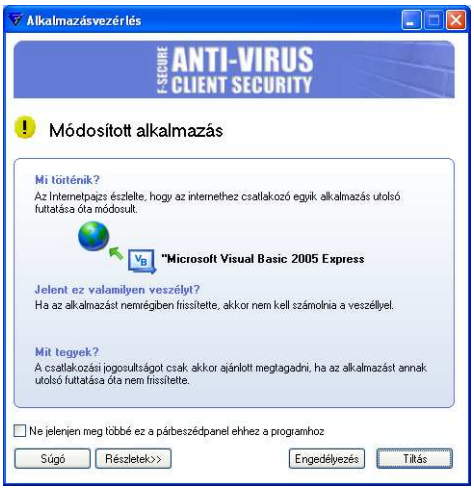
Az online súgó letiltása

A programozás során célszerű letiltani az online súgót, mert a gyakorlati érettségin nem használhatjuk az Internetet! A súgó első megnyitásánál a fejlesztői rendszer rákérdez az online *Help* engedélyezésére. Utólag a súgóablak *Tools*/*Options* parancsával módosíthatjuk a beállítást. Válasszuk a *Help*/*Online* listaelemet, majd jelöljük be a *Try local only, not online* választógombot!



Az online súgó letiltása

Ne felejtsük el a fejlesztői környezet Internet-elérését is letiltani. Ezt általában a számítógépre telepített tűzfal segítségével tehetjük meg. Megfelelő beállítás esetén a tűzfal rákérdez az Internet-elérés engedélyezésére. Válasszuk a tiltás funkciót!



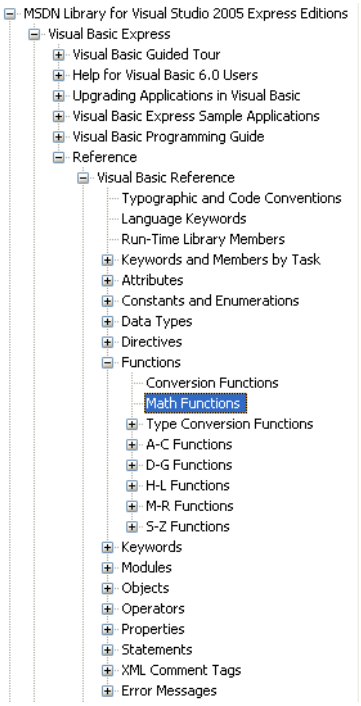
A fejlesztői környezet Internet-elérésének letiltása az F-Secure használata esetén

A Visual Basic referencia

A súgó egyik leggyakrabban használt része a nyelv leírása, melynek megértése csak elemi angoltudást igényel. A referenciát a súgó tartalomjegyzékében a Visual Basic *Express*/*Reference*/*Visual Basic Reference* címszó alatt találjuk. Legfontosabb elemei:

Constants and Enumerations	előre definiált konstansok és felsorolások
Data Types	adattípusok
Functions	függvények (A matematikai függvények ismertetését ne az ábécérend szerint, hanem a *Math Functions* csoportban keressük!)
Operators	operátorok, műveleti jelek
Statements	utasítások

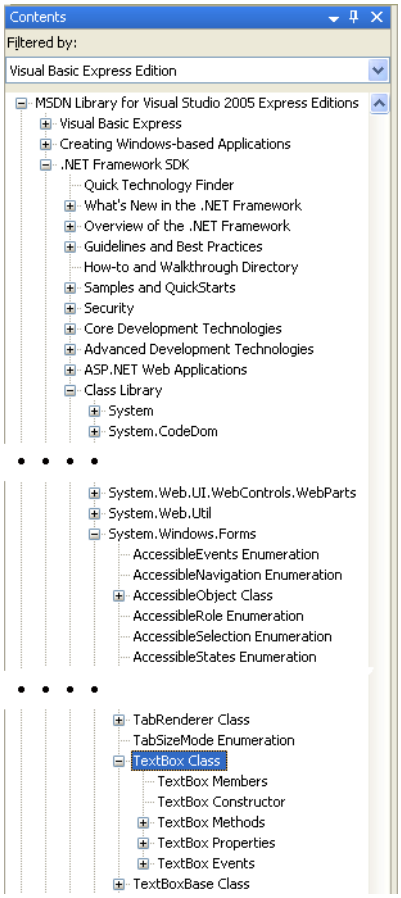
A leírások végén egy vagy több forráskódpéldát is láthatunk, amely bemutatja a megfelelő nyelvi elem használatát.



A súgó tartalomjegyzéke

A grafikus felhasználói felület objektumai a súgóban

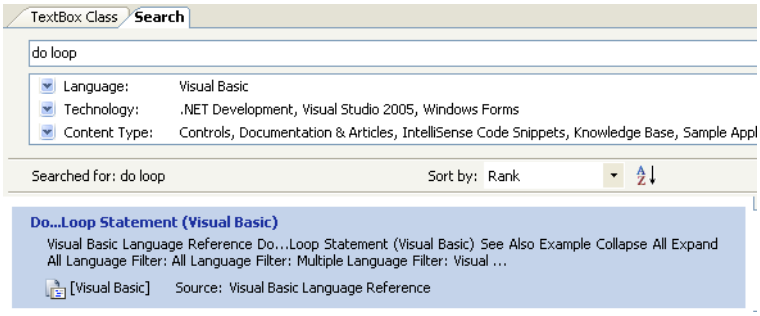
A grafikus felület objektumait (illetve a nekik megfelelő osztályokat) a .NET Framework definiálja, így a súgó tartalomjegyzékének *.NET Framework SDK*/*Class Library* címszavát kell kibontatnunk. Egy-egy osztály kiválasztásakor a rövid leírás mellett megtaláljuk a tulajdonságok és metódusok (members) ismertetését is. A szövegdoboz (textbox) osztály leírásának helye például:



A szövegdoboz a súgó tartalomjegyzékében

Keresés a súgóban

Szükség esetén meg is kereshetjük a kérdéses fogalmat, kulcsszót a súgóban. Ehhez válasszuk a *Help*/*Search* parancsot vagy a súgóablakban a *Search* fület! Írjuk be a keresett szót a szövegdobozba, majd nyomjuk le az Entert (vagy kattintsunk a *Search* gombra). A megjelenő listában a súgó kék háttérrel jelöli a kereséshez legjobban illeszkedő találatot. A sötétkék címszóra kattintva megjelenik a keresett elem leírása.



A Do…Loop utasítás keresése és a találati lista első eleme