

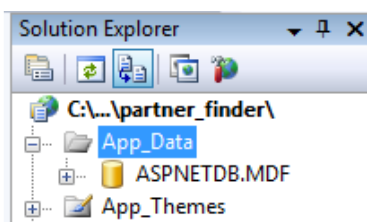
# ASP.NET 3.5 Tutorial II. rész (Adatbázis, Tartalomkialakítás, Adatkötés)

És már vissza is tértünk a rövid reklámszünetből, remélhetőleg mindenki újult erővel, illetve új billentyűzettel vág bele a Tutorial második és egyben befejező részébe. Innentől mindenki csak saját felelősségére olvassa tovább! ☺

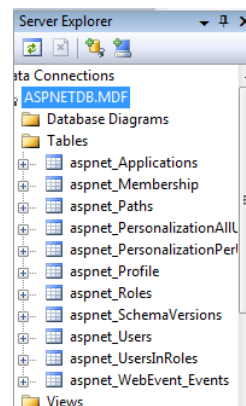
## Adatbázis kibővítése

Az előző tutorialt ott hagytuk abba, hogy elkészültünk a regisztrációs oldal UI-ával (felhasználói felületével), de az újonnan felvett mezők tartalmát még nem mentettük el az adatbázisba. Ebből két dolog következik. Az egyik az, hogy már van egy adatbázisunk, csak még nem tudunk róla, a másik pedig, hogy a CUW nem menti el automatikusan az új adatokat. De mégis hol van az adatbázisunk és milyen néven? A kérdés jogos, a válasz pedig egyszerű. Az **App\_Data** könyvtárban és **ASPNETDB.MDF** néven. Ez az adatbázis akkor jött létre, amikor a WSAT-ban felvettünk egy teszt felhasználót.

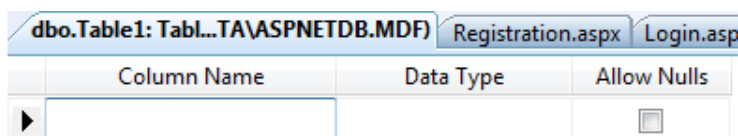
1. Ahhoz, hogy láthassuk is az adatbázisunkat az App\_Data mappát frissítenünk kell. Jobb klikk a könyvtáron, majd **Refresh Folder!** Ezek után a Solution Explorerben emellett a mappa mellett is megjelenik a kis plusz ikon, melyre rákattintva láthatjuk a könyvtár tartalmát.



2. Az adatbázis megnyitásához kattintsunk kétszer az ASPNETDB.MDF-re! Ilyenkor pár másodpercig elszöszmötöl a Visual Studio, majd a **Server Explorer** ablakra vált át a Solution Explorer-ről. Nyissuk ki a **Tables** csomópontot! Itt láthatunk egy jó pár *aspnet\_*-sal kezdődő táblát. Az **aspnet\_Users**-ben találhatóak a felhasználók adatai (*név, jelszó hash, ... , utoljára bejelentkezés dátuma, stb.*). Ha nagyon akarnánk, akkor ezt a táblát is kibővíthetnénk, de nem célszerű az *aspnet\_* szóval kezdődő táblákat módosítani! Ezért egy új adattáblában fogjuk eltárolni a felhasználókról az extraadatokat.

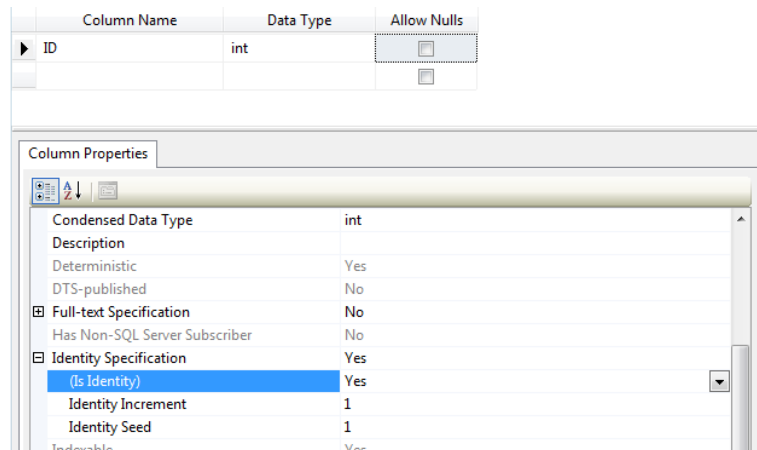


3. Kattintsunk a Tables csomóponton jobb egér gombbal, majd **Add New Table!** Ekkor középen meg fog nyílni egy új ablak, ahol az adattáblánk szerkezetét alakíthatjuk ki, vagyis a mezőket/oszlopokat definiálhatjuk.



Az első oszlopunk azonosítónak fog funkcionálni, ezért az **ID** nevet adjunk neki! A típusa legyen **int**, és ne engedjük meg, hogy **null** értéket felvehessen! Alul a **Column Properties**

ablakban a **Table Designer** csoporton belül az **Identity Specification** csomópontot nyissuk ki! Itt az **(Is Identity)** tulajdonság melletti **No**-ra kattintsunk kétszer!



Most azt állítottuk be, hogy az ID oszlopunk automatikusan kapjon értéket. Minden új felhasználó eggyel nagyobb azonosítót fog kapni, mint az előző. Ez azért jó, mert így ezzel nekünk a jövőben már nem kell foglalkoznunk. ☺

4. Az egyszerűség kedvéért ebben a táblában is el fogjuk tárolni a felhasználó nevét, illetve az e-mail címét. Bár emiatt az adatbázisunk már nem lesz *redundáns*, de így csak egyetlen adattáblából kell majd kinyernünk az adatokat!


5. Tehát vegyünk fel egy **Nickname**, illetve egy **Mail\_Address** nevű oszlopot, majd pedig egy **City** nevűt is! Mindháromnak a típusa legyen **nvarchar(50)**! Az 50-eket írjuk át **100**-ra, és ne engedjük meg a null értékeket!

6. A következő oszlopunkban a születési dátumot fogjuk eltárolni, ezért legyen a neve **Born\_Date**, legyen dátum értékű (**Date**), és ne lehessen null! Vegyünk fel még egy **IsFemale** mezőt is, amely logikai típusú (**bit**) legyen, és szintén ne engedélyezzük a null értéket!

7. Mint minden rendes párkereső oldalnál lehetőséget kell biztosítanunk a felhasználóknak képfeltöltésre, illetve rövid bemutatkozó szöveg írására. A képeknek csak a nevét fogjuk eltárolni az adatbázisunkban. Lesz egy alapértelmezett képünk is, amely addig fog megjelenni, amíg nem tölt fel magáról képet a felhasználó!

8. Adjunk az adattáblánkhoz egy új **Image** nevű oszlopot, a típusa legyen **nvarchar(100)**! Alul a Column Properties ablak (**General**) csoportjában a **Default Value or Binding** tulajdonsághoz írjuk be azt, hogy `'no_image.jpg'`! Az **Allow Null**-t állítsuk **False**-ra!

9. Utolsó mezőként vegyünk fel a **Sort\_Description**-t! A rövid leírás legyen max. 1000 karakter hosszú (**nvarchar(1000)**), és itt kivételesen engedjük meg a null értéket is!

10. Végül állítsuk be az ID mezőt **elsődleges kulcsnak**. (Hogy erre miért van szükségünk, azt most nem részletezném, kell és kész ☺!) Jelöljük ki az első sort, majd az eszköztáron kattintsunk a kulcs ikonra !



11. Mentsük el a munkánkat, nyomjunk le a **Ctrl+S** billentyűkombinációt, majd adjuk azt a nevet az adattáblánknak, hogy *members*!

Column Name	Data Type	Allow Nulls
ID	int	<input type="checkbox"/>
Nickname	nvarchar(100)	<input type="checkbox"/>
Mail_Address	nvarchar(100)	<input type="checkbox"/>
City	nvarchar(100)	<input type="checkbox"/>
Born_Date	datetime	<input type="checkbox"/>
IsFemale	bit	<input type="checkbox"/>
Image	nvarchar(100)	<input type="checkbox"/>
Sort_Description	nvarchar(1000)	<input checked="" type="checkbox"/>
		<input type="checkbox"/>

12. Vegyünk fel egy teszt rekordot az adatbázisunkba, hogy kipróbáljuk, minden tökéletesen működik. Ehhez kattintsunk jobb egér gombbal a Server Explorer-ben a members táblára, majd itt a helyi menüben válasszuk a **Show Table Data** menüpontot. Ilyenkor az alábbi ablak fog megjelenni közepén:

ID	Nickname	Mail_Address	City	Born_Date	IsFemale	Image	Sort_Description
* NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

13. Bár a képen annyira nem látszik, de az ID mező null értéke pirosas színű, ami arra utal, hogy ne nyúlj hozzám☺. Teszt-felhasználónk ki más lehetne, mint Teszt Elek, kinek e-mail címe [teszt@elek.com](mailto:teszt@elek.com), város ahol él az Budapest, és 1950.07.15-én született, illetve természetesen férfi.

ID	Nickname	Mail_Address	City	Born_Date	IsFemale	Image	Sort_D
⚡ NULL	Teszt Elek	teszt@elek.com	Budapest	1950.07.15. 0:00	False	NULL	NULL
* NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

14. Kattintsunk az alatta lévő sorba, majd mentünk! Zárjuk be ezt az ablakot, majd nyissuk meg újból! Így már látni fogjuk, hogy az 1-es azonosítót kapta Elek barátunk, illetve az Image mezője is kitöltődött automatikusan a *no\_image.jpg*-gel. Szuper!

ID	Nickname	Mail_Address	City	Born_Date	IsFemale	Image	Sort_D
▶ 1	Teszt Elek	teszt@elek.com	Budapest	1950.07.15. 0:00...	False	no_image.jpg	NULL
* NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

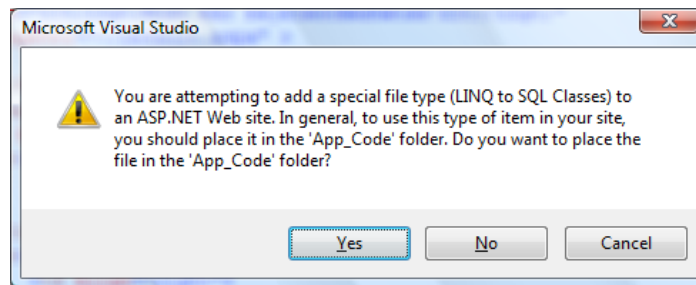
15. Zárjuk be a members adattáblánkhöz kapcsolódó ablakokat, és a Server Explorer-ről váltsuk vissza Solution Explorerre!



## Adatbázisból Entitás osztályok generálása

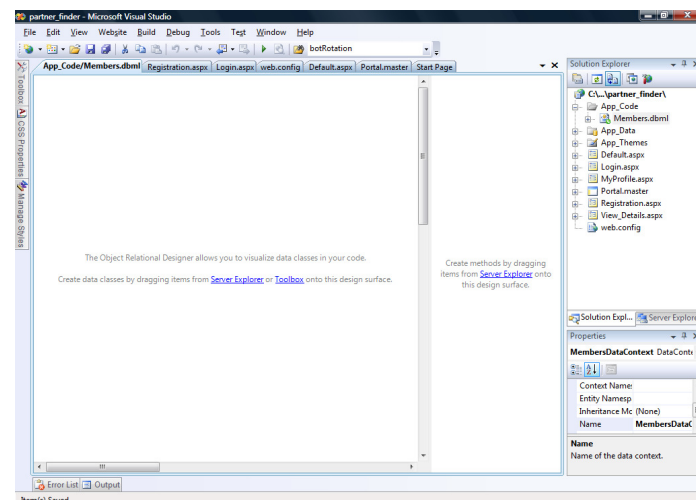
Ahhoz, hogy az adatbázisban lévő adatokat kényelmesen LINQ-val (*nyelvbe ágyazott lekérdezésekkel*) tudjuk menedzselni, szükségünk lesz **entitás osztályokra**. Ha nagyon durván akarnék fogalmazni, akkor azt mondanám, hogy az entitás osztályok az adattáblák, rekordok, stb. *objektum orientált* megfelelői. Ez a definíció jelen esetben megállja a helyét, de akiket pontosabban érdekel a Linq háttere, nekik a [Tanulmányi Hét Linq To Sql leírását](#) ajánlom, a la Árvai Zoli!

1. Legelőször létre kell hoznunk egy \*.dbml fájl, ahol megszabhatjuk, mely adattáblákból generálódjanak entitás osztályok. Magából az adatbázisból is készül egy osztály, de az nem entitás lesz, hanem egy úgynevezett **DataContext** objektum!
2. Kattintsunk a Solution Explorer-en belül a projektünkön jobb egérgombbal, majd *Add New Item...* Itt keressük meg a **LINQ to SQL Classes** ikont, és azt a nevet adjuk az új fájlnak, hogy **Members.dbml**! Majd klikk az *Add* gombra!
3. Ilyenkor az alábbi figyelmeztetés fog felpattanni, amely összesen annyiról tájékoztat minket, hogy még nincs **App\_Code** mappánk, és szeretnénk-e, hogy létrehozza nekünk a Visual Studio:

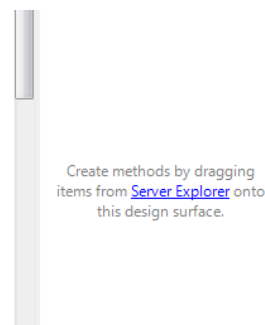
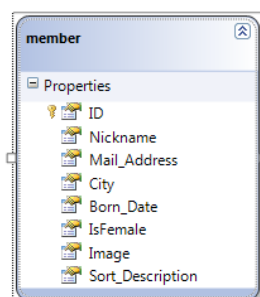


Természetesen szeretnénk, ezért kattintsunk a *Yes* gombra!

4. Ezek után a Visual Studio körülbelül fél percig keményen hegeszt a háttérben, majd az alábbi ablak fog megjelenni:



5. Most váltsunk vissza jobboldalt a Solution Explorer-től Server Explorer-re! Fogjuk meg a members táblánkat és drag&drop módszerrel húzzuk rá a baloldali nagy fehér területre! (Egy kis érdekesség: a members táblánkból member lett! 😊)




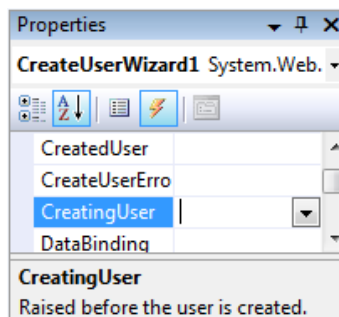
6. Mentsünk, majd pedig zárjuk be a dbml fájlunkat! A mentés szintén el fog tartani egy kis ideig, ugyanis ilyenkor generálódnak le az entitás osztályok.

## Regisztrációs oldal befejezése

Most már minden rendelkezésünkre áll ahhoz, hogy a felhasználókról beszerzett felbecsülhetetlen értékű új információkat eltároljuk az adatbázisunkban. Tárolásra fel!

1. A CUW-unknak van egy jó pár eseménye, amelyekre feliratkozhatunk (pl.: **ContinueButtonClick**, **DataBindig**, **SendingMail**, stb.), de ezek közül csak néhány jó most számunkra. Mi most a **CreatingUser** eseményt választjuk, ami azelőtt hajtódik végre, mielőtt a felhasználóhoz még létrejönne a rekord az adatbázisban. (Természetesen a **CreatedUser** esemény is tökéletesen megfelelő, ha esetleg valakinek ez szimpatikusabb lenne!)

2. A CUW Properties ablakának az eszköztárában kattintsunk a villám ikonra (  ), így nem a tulajdonságait, hanem az *eseményeit* fogjuk látni a CreateUserWizard-nak.



3. Keressük meg a CreatingUser eseményt és a mellette lévő szövegdobozba kattintsunk kétszer. Ilyenkor a Visual Studio meg fogja nyitni a **Registration.aspx.cs** fájlt és létrehozza benne ehhez az eseményhez az eseménykezelő függvényt (**CreateUserWizard1\_CreatingUser** néven).

4. Először is változóba el kell tárolnunk a beviteli mezők szövegeit. A felhasználónév, illetve e-mail cím elkérése eléggé egyszerű, ugyanis direktbe, tulajdonságokon keresztül lekérdezhetőek. Íme a két érték letárolásának kódja:

```
string name = this.CreateUserWizard1.UserName;  
string mail = this.CreateUserWizard1.Email;
```

A **this** objektum magát az oldalt jelöli, amin keresztül elérhetjük a template-sített CUW-unkat. (Ha nem sablonosítjuk, akkor **this** nélkül is simán hivatkozhatunk rá!)

5. A városnév elkérése már kicsit neccesebb, ugyanis erre nincs előredefiniált tulajdonsága a CUW-nak, ezért először el kell érnünk a txtbox\_city beviteli mezőt. Ezt az alábbi módon tehetjük meg:

```
this.CreateUserWizard1.CreateUserStep.ContentTemplateContainer.FindControl(  
"txtbox_city")
```

(Hogy ezt miért így kell elkérni, abba most nem mennék bele, mert messzire vezetne, ezért elégedjünk meg annyival, hogy így kell csinálni!)

6. A **FindControl** metódus egy egyszerű **object** típusú objektumot ad vissza. Mivel mi tudjuk, hogy ez egy TextBox, ezért ezt meg is mondhatjuk a .NET-nek. Ezt úgy tudjuk megtenni, hogy az egész elé zárójelek közé odaírjuk, hogy TextBox (ezt hívjuk **kasztolásnak**). Most tegyünk az egész legelejére és a legvégére is egy-egy zárójelet, így már az IntelliSense is tudni fogja, hogy ez egy beviteli mező, így ha a kódsor legvégére oda írjuk, hogy „.Text”, akkor nem fog kiadni a VS. Tehát a városnév elkérése egyben így néz ki:

```
string city = ((TextBox)
this.CreateUserWizard1.CreateUserStep.ContentTemplateContainer.FindControl(
"txtbox_city")).Text;
```

7. A születési dátum elkérése hasonlóképpen történik, csak ott dátum formátumba még át kell konvertálnunk a szöveget (**string**-et). Ezt a **Convert** objektum **ToDateTime()** metódusával tehetjük meg. Valami hasonló a születési dátumot elkérő kódunk:

```
DateTime born = Convert.ToDateTime(((TextBox)
this.CreateUserWizard1.CreateUserStep.ContentTemplateContainer.FindControl(
"txtbox_born")).Text);
```

8. Már csak a NEM elkérése van hátra. Ez annyiban tér el az előzőektől, hogy itt a **FindControl**-t nem *TextBox*-ra, hanem **DropDownList**-re kell kasztolnunk, illetve nem a **Text** tulajdonságát kérjük el, hanem a kiválasztott elemértékét (**SelectedValue**). Íme a szinte már kisujjból jövő kód:

```
string sex = ((DropDownList)
this.CreateUserWizard1.CreateUserStep.ContentTemplateContainer.FindControl(
"ddl_sex")).SelectedValue;
```

9. A most letárolt NEM értéket át kell alakítanunk logikai értéké attól függően, hogy az új felhasználó hölgy vagy úr. Ezért felveszünk egy *isfemale* nevű **bool** (logikai) változót, és az értékét az alábbi módon állítjuk be:

```
bool isfemale;
if (sex == "Girls") isfemale = true;
else isfemale = false;
```

10. Elérkeztünk a dolog érdemi részéhez az adatbázisba történő mentéshez. Először is létre kell hoznunk egy **DataContext** példányt (ez reprezentálja az adatbázisunkat). A **DataContext** úgy kapja a nevét, hogy a dbml fájl neve + **DataContext**, vagyis jelen esetben a **MembersDataContext** néven tudunk hivatkozni az adatbázist reprezentáló osztályra.

```
MembersDataContext mdc = new MembersDataContext();
```

11. Ezek után létre kell hoznunk egy új *felhasználót/member entitást/ rekordot!* Majd pedig a változóknak eltárolt értékeket be kell állítanunk az új **member** példány tulajdonságain keresztül, valahogy így (Sajnos a képet is kézzel be kell állítani, mivel a default értéket valamiért nem hajlandó átvenni az adatbázistól a .NET):

```
member m = new member();
m.Nickname = name;
m.Mail_Address = mail;
m.City = city;
m.Born_Date = born;
m.IsFemale = isfemale;
m.Image = "no_image.jpg";
```

12. Végül már csak az új rekordot be kell szúrni a **members** táblába, illetve utána érvényre kell juttatni a változásokat:

```
mdc.members.InsertOnSubmit(m);
mdc.SubmitChanges();
```

13. És készen is vagyunk! Akár ki is próbálhatjuk, bár egyelőre még nem fogunk semmit se látni belőle a portálon, de **Server Explorer**-ből megnézhetjük, hogy ténylegesen jól töltődtek-e fel a mezők adatokkal.

## Oldalak feltöltése tartalommal

Az oldalainknak nem csak a kinézete, de a tartalma is függeni fog az adott felhasználó nemétől. Ezért érdemes lenne a bejelentkezett felhasználóról eltárolni egy globális változóban a nemét, hogy ne kelljen emiatt mindig egy plusz lekérdezéssel zavarni az amúgy is elfoglalt adatbázisszervert. Ahhoz, hogy ez az információ bárholnan elérhető legyen az egyik legegyszerűbb megoldás, ha a **Profile**-ban tároljuk el ezt az adatot.

## Profil(e) használata

A Profil nem más, mint a felhasználói adatok, beállítások **perzisztens** módon történő tárolása. A perzisztens szó arra utal, hogy ezek az adatok is valójában az adatbázisban vannak eltárolva, de ezek lekérdezése és karbantartása számunkra láthatatlan módon, a háttérben történik, automatikusan (**aspnet\_Profile** tábla). Magyarul a profil egy eléggé kényes módja a gyakran használt (illetve gyakran változó) felhasználóra vonatkozó adatok kezelésének. A Profile-ban eltárolni kívánt adatokat a web.config-ban kell beállítanunk (definiálnunk).

1. Tehát nyissuk meg a web.config fájlt, majd keressük meg a **system.web** tag-et! Ezen belül kezdjük el írni, hogy **<prof**, majd az IntelliSense felajánlja a **profile**-t, válasszuk ki, majd nyomjunk egy *Tab*-ot! Nyomjunk le *space*-t és utána írjuk be **enabled="true"**. Ezzel bekapcsoltuk a profil szolgáltatást. Zárjuk a be profile tag-ünket (>). Most itt tartunk:

```
<system.web>
  <profile enabled="true">
  </profile>
```

2. Annyi dolgunk van még itt összesen, hogy megadjuk, milyen típusú adatot akarunk eltárolni és milyen néven. Ehhez a profile tag-eken belül létre kell hoznunk a **properties** tag-et. A properties-en belül pedig egy **add** tag segítségével tudunk új profil tulajdonságot felvenni. (Azért tulajdonság, mert a **Profile** objektumon keresztül, mint tulajdonság tudjuk majd elérni!) Az add tagnek állítsuk be a **name** attribútumát „**Theme**”-re, a **type**-jét „**string**”-re és a **defaultValue**-t pedig „**Girls**”-re! Mentsünk, majd pedig zárjuk be a web.config fájlt!

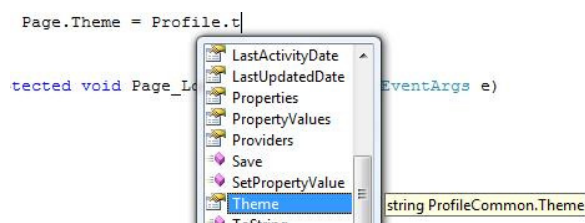
```
<profile enabled="true">
  <properties>
    <add name="Theme" type="string" defaultValue="Girls"/>
  </properties>
</profile>
```

3. A **MyProfile** oldalon lehet majd beállítani a Profil Theme tulajdonságát, de ahhoz, hogy az ott beállított téma érvényre is jusson mindenhol, meg kell mondanunk az oldalaink, hogy a profile-tól kérjék el a témát. Ahhoz, hogy mindig a megfelelő témát töltsék be, az oldalaink **Page\_PreInit** eseményeinél kell ezt beállítanunk. Vagyis mindegyik oldalnál, kivéve a Login-nál, illetve Registration-nél az alábbi kódot szúrjuk be a Page\_Load esemény fölé közvetlen:

```
protected void Page_PreInit(object sender, EventArgs e)
{
    Page.Theme = Profile.Theme;
}
```

```
protected void Page_Load(object sender, EventArgs e) ...
```

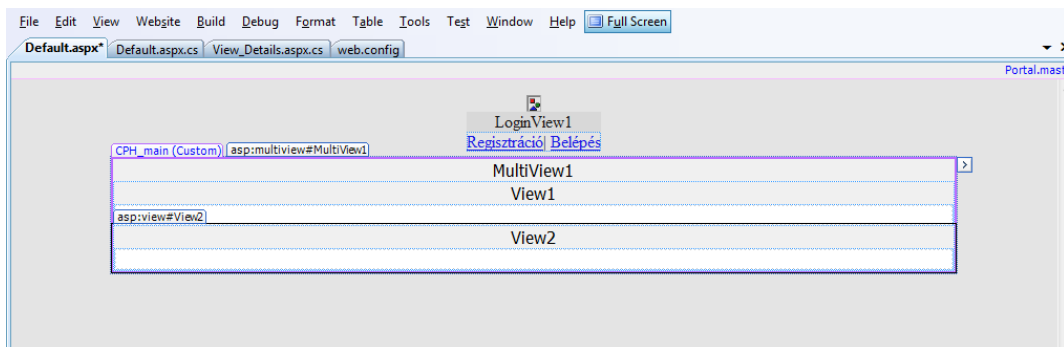
4. Amint azt kódíráskor láthattuk is, ténylegesen tulajdonságokként érhetjük el a Theme profil adatot.



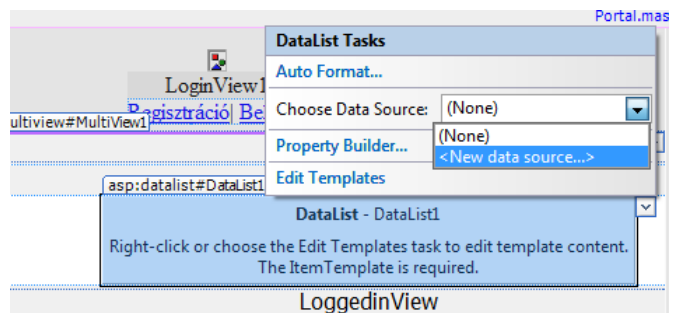
## - Default.aspx

A főoldalunknak hasonlóan a menühöz két arca lesz. Az egyik arcát a vendég felhasználók felé fogja mutatni, a másikat pedig a bejelentkezettek láthatják majd. Ezt is megoldhatnánk *LoginView*-val, de mivel ez egy DEMO alkalmazás, ezért most egy másik módszerrel fogjuk ugyanezt elvégezni. Segítségünkre a **MultiView** vezérlő lesz és az ő gyermek kontrolljai a **View**-k.

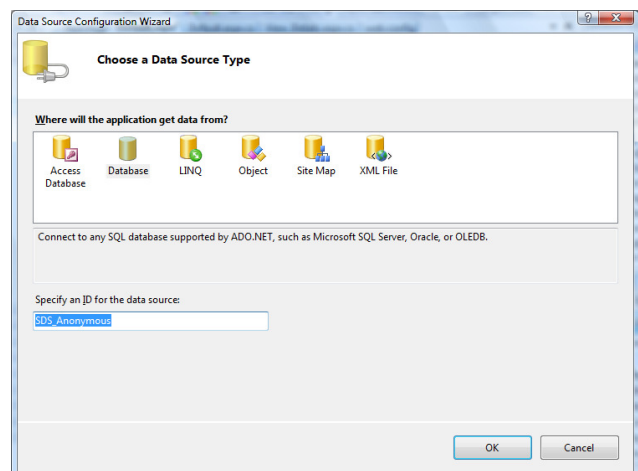
1. Húzzunk a Default.aspx oldalunk **Content**-jébe egy **MultiView** vezérlőt, majd pedig ebbe két **View** kontrollt! Alacsony felbontás esetén nyomjuk meg az **Alt+Shift+Enter** billentyűkombinációt ahhoz, hogy az egész oldalunk kiferjjen a Visual Studio-ban, és ne lógjon bele egyetlen ablak se. Ezzel a billentyűkombinációval tudunk *teljes képernyős nézetbe* váltani. Kódolás esetén is sokszor hasznos tud lenni, ezért használjuk előszeretettel!



2. A View1-et nevezzük át **AnonymousView**-ra, a View2-t pedig **LoggedInView**-ra! Először a vendég nézetet készítjük el. Húzzunk a felső View-ba egy **DataList** vezérlőt a **Toolbox Data** Tab-jéről! Ilyenkor automatikusan meg fog jelenni a **Smart Tag**, ahol a **Choose Data Source** mellett legördülő listából válasszuk a **<New data source...>** lehetőséget!

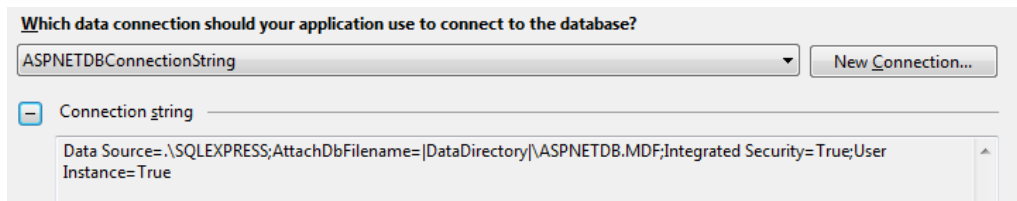


3. Ilyenkor el fog indulni a **Data Source Configuration Wizard**, amelynek az első oldalán válasszuk felül a **DataBase**-t adatforrásnak, majd adjuk azt a nevet az új **SqlDataSource** vezérlőnek, hogy **SDS\_Anonymous**. Kattintsunk az **OK** gombra!

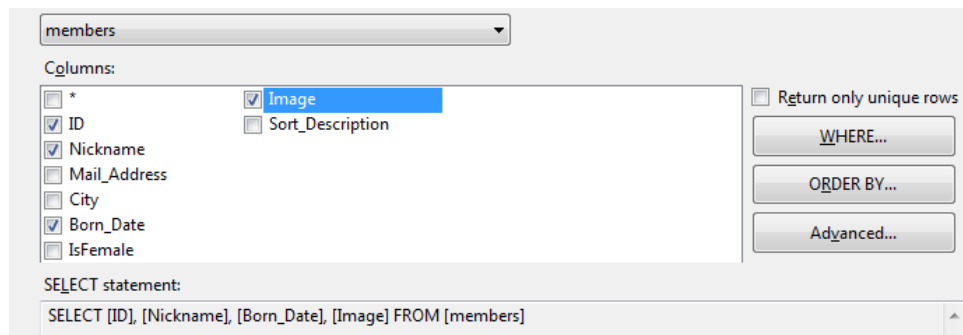


4. A Varázsló következő oldalán az adatbázishoz történő kapcsolódáshoz szükséges adatokat leíró **ConnectionString**-et kell kiválasztanunk. Mivel WSAT-on keresztül csatlakoztunk az ASPNETDB.mdf-hez, ezért a web.config-ban a Visual Studio egyszer már eltárolta a hozzáféréshez szükséges információkat! Ezért a legördülő-listából válasszuk ki az **ASPNETDBConnectionString**-et! Ha alatta a + gombra kattintunk, akkor meg is nézhetjük, hogy mi is ez konkrétan. Akiket

érdekelnek a részletek, azoknak ajánlom, hogy látogassanak el a [connectionstrings.com](http://connectionstrings.com) oldalra!



5. Kattintsunk alul a *Next* gombra, majd a kövi oldalon klikkelgessük be az *ID*, a *NickName*, a *Born\_Date*, illetve az *Image* mezőket! Alul láthatjuk a generált **SQL** lekérdezésünket (ezen majd kicsit később még módosítunk). Katt ismét a *Next*-re!

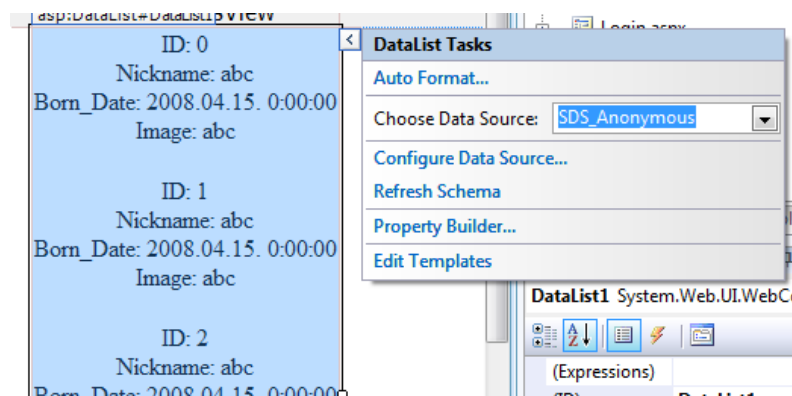


6. A Varázsló utolsó oldalán lehetőségünk van arra, hogy leteszteljük a lekérdezésünket, azáltal, hogy a *Test Query* gombra kattintunk!

ID	Nickname	Born_Date	Image
1	Teszt Elek	1950.07.15.	no_image.jpg
3	Teszt Eszter	1950.07.15.	no_image.jpg

[Test Query](#)

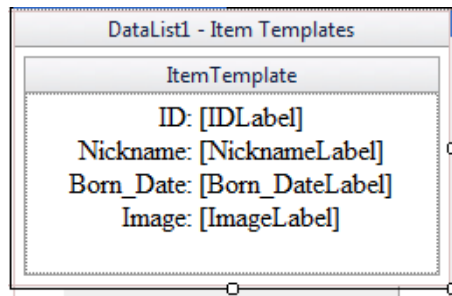
7. Miután ráklikkeltünk a *Finish* gombra, az alábbi látvány fog fogadni minket:



Ez már majdnem jó, ugyanis a DataList lényege, hogy nem táblázatos formában jeleníti meg az adatokat, hanem az egyes rekordok mezőit bármilyen kontrollhoz hozzákötve egyedi megjelenítést alakíthatunk ki! (A Visual Studio hamis adatokkal feltölti az adatlista vezérlőnkét annak érdekében, hogy le tudjuk tesztelni az elrendezés megfelelő-e!) Mi az egyes sorokat külön-külön egy kis táblázatban fogjuk reprezentálni. Egy adott táblának

két oszlopa lesz, az elsőben egy kép fog megjelenni az adott felhasználóról, a másodikban pedig három sorba bontva megjelenítjük az alábbi információkat: Nicknév, Születési dátum és egy bővebb infó-s link.

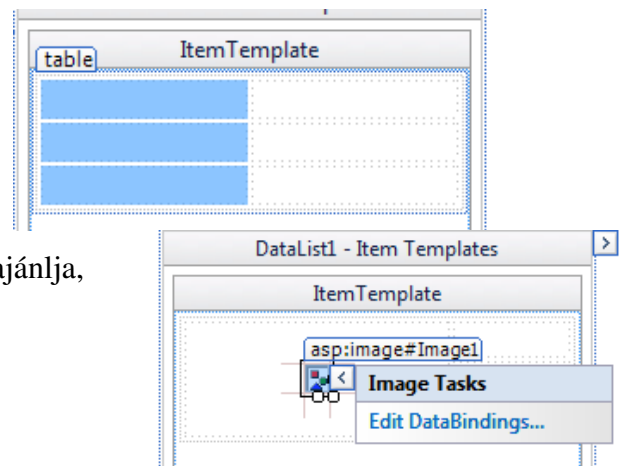
8. A jelenlegi *elemsablon*-t (a rekordok kinézetét leíró **item template**-t) úgy tudjuk módosítani, hogy a DataList Smart Tag-jében az **Edit Templates** linkre kattintunk. Ilyenkor az alábbi szerkesztő nézetbe fog átváltani az adatlista vezérlő:



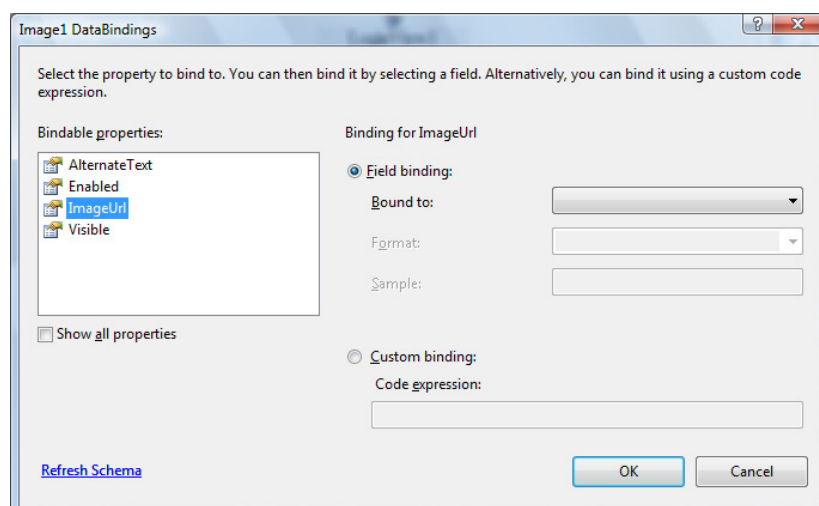
Töröljük ki az összes benne lévő adatot, majd szúrjuk be a helyére egy 3\*2-as táblázatot! **Table** menü belül kattintsunk az **Insert Table**-re, majd itt állítsuk be, hogy 3 soros és 2 oszlopos legyen a beszúrandó táblázatunk!

9. Jelöljük ki az első oszlopot, majd jobb egérgomb, **Modify** almenü, **Merge Cells**!

Az első oszlopba húzzunk be egy Image vezérlőt! Miután megjelent a kontroll a táblázatban automatikusan megjelenik a Smart Tag-je is és felajánlja, hogy **adatkössük**, ezért kattintsunk az **Edit DataBindings...** linkre!

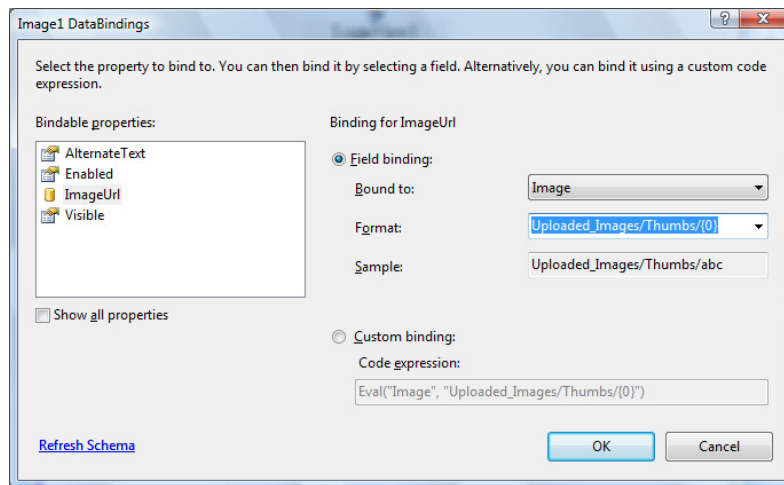


10. Ilyenkor az alábbi ablak fog felpattanni:



Bal oldalt a vezérlő azon tulajdonságait láthatjuk, amelyek *adatköthetőek*. Mi az **ImageUrl**-t szeretnénk összekapcsolni az adatbázisunk Image celláival. Ezért jobb oldalt a **Bound to** mellett legördülő-listából válasszuk az Image-et! Az alatta lévő **Format** mellett szövegdoboz értékét írjuk át a következőre: „**Uploaded\_Images/Thumbs/{0}**”. Az

*Uploaded\_Images* mappába (amelyet majd hamarosan elkészítünk) fogjuk eltárolni a felhasználókról a fényképeket. A feltöltött képekből dinamikusan generáltatni fogunk *bélyeg méretű képeket* is (**thumbnails**), melyek a *Thumbs* mappába lesznek elmentve. A format végén lévő *{0}* egy helyőrző, aminek a helyére be fog helyettesítődni az Image cellák értéke.

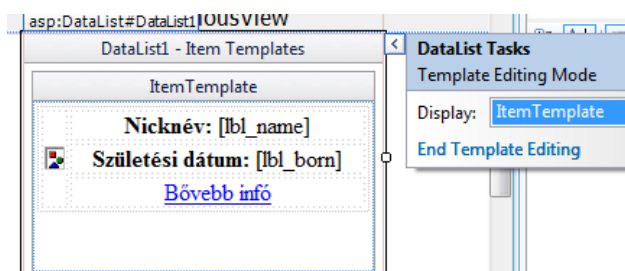


11. Szuper, a kép kötésével már meg is vagyunk! Most a táblázatunk második oszlopát fogjuk feltölteni adatokkal! Az első sorába írjuk be, hogy „**Nicknév:**”, majd vegyük félkövérre, és húzzunk be mellé egy **Label** kontrollt! A címkének töröljük ki a Text tulajdonságát, és azt az azonosítót adjuk neki, hogy *lbl\_name*! A properties ablakban a **Font** tulajdonságcsoportját nyissuk ki, majd állítsuk az **Italic**-ot *True*-ra! A címke Smart Tag-ből válasszuk ki az Edit DataBinings linket! Itt a Text tulajdonságot kössük össze a NickName-mel! A Format-nál a legördülő listából válasszuk a **General – {0}** lehetőséget!

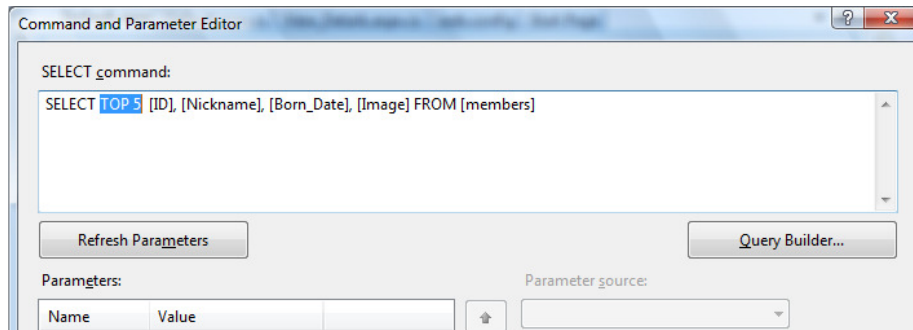
12. Hasonlóan az előzőhöz, a második sorba a „**Születési dátum:**” szöveget írjuk, és húzzunk be mellé egy címkét, amelynek az ID-ja legyen *lbl\_born*! A formázások legyenek ugyanazok, mint az előző sorban! Szokásos Smart Tag >> Edit DataBindings, majd kössük össze a Text tulajdonságot a Born\_Date mezővel! A Formátumot pedig itt **Short date –{0:d}**-re állítsuk! Alatta láthatjuk is, hogy csak a dátum jelenik meg, ha *Long date*-t választanánk, akkor az idő is megjelenne, de nekünk az nem kell!

13. Az utolsó sorba húzzunk be egy **HyperLink** vezérlőt, a **Text**-jét állítsuk be „**Bővebb infó**”-ra, a **NavigationUrl** tulajdonságát pedig a *Login.aspx* oldalra! (Így sarkaljuk a vendéget a regisztrációra!) Ha az utóbbit a properties ablakon keresztül állítjuk be, akkor a ... gombra kattintva kézzel is kiválaszthatjuk az oldalt, így nem kell bepötyögni a teljes oldalnevét!

14. Elkészültünk a sablonnal, ezért a DataList Smart Tag-jében kattintsunk az **End Template Editing** linkre! Váltunk át Source nézetre és a DataList **ItemTemplate**-jén belül a táblázat szélességét írjuk át *100%*-ról **350px**-re! Végül pedig az első oszlop szélességét állítsuk be *150px*-re! Így szebben fog kinézni az egész.



15. Azért, hogy valóságosabbnak tűnjön a rendszerünk, nem az összes felhasználónkat fogjuk kilistázni a vendégeknek, hanem csak az első ötöt. Jelöljük ki az **SDS\_Anonymous** SqlDataSource vezérlőnkét és keressük meg a properties ablakban a **SelectQuery** tulajdonságát, majd kattintsunk a mellette lévő ... gombra! Itt annyit módosítunk a lekérdezésen, hogy a **Select** kulcsszó után legyen egy **TOP 5** utasítás is!



(Értelemszerűen valós rendszer esetén 5db véletlen számot generálnánk, és azon felhasználókat listáznánk ki, akiknek az ID-ja megegyezik valamelyik random számmal.)

16. Az AnonymousView-val elkészültünk, ezért most térjünk át a *LoggedInView*-ra! Itt az ellenkező nem tagjait fogjuk felsorolni. Természetesen nem minden információt fogunk itt megjeleníteni róluk, hanem csak a nevüket, születési dátumukat, városukat, illetve egy-egy linket, amely a **View\_Details.aspx** oldalra fog mutatni. A linken keresztül majd át fogjuk passzolni a kiválasztott hölgy/úr azonosítóját, hogy ott a megfelelő felhasználóról minden információt megjeleníthessünk.

17. Ellentétben az AnonymousView-val itt nem *DataList*-et fogunk használni, hanem **ListView**-t + **DataPager** kontrollt, ugyanis így a felsorolandó rekordok **lapozhatóak** lesznek! (Egyszerre mindig csak x db felhasználó fog megjelenni, és alul lehet majd lapozgatni az x db-os csoportok között! pl.: ha x = 5, akkor változathatunk aközött, hogy az első öt jelenjen meg, vagy a 6.-tól a 10.-ig jelenítsük meg a felhasználókat, stb.) Emellett az adatok lekérését nem varázslókkal (**deklaratív módon**) fogjuk megoldani, hanem kódból (**imperatív módon**)! Na, akkor ugorjunk is neki!

18. Húzzunk a LoggedinView kontrollunkba először egy ListView vezérlőt, majd váltsunk át kódnézetbe! Azért fogjuk Source nézetben szerkeszteni a sablonokat, mivel itt nem áll rendelkezésünkre olyan vizuális szerkesztő, mint amilyen az adatlistánál volt! A ListView vezérlőnél már két sablont kötelező megadni (ellentétben a datalist-tel, ahol csak egyet kellett). Itt az **ItemTemplate**-n kívül még a **LayoutTemplate**-t is definiálnunk kell. A kötelezőkön kívül mi még az **ItemSeparatorTemplate**-t is meg fogjuk adni, amellyel az egyes rekordok közötti elválasztó html tageket fogjuk majd definiálni.

19. Először a LayoutTemplate-t kell megírunk. Itt a fejléctet, illetve a rekordokat közrefogó kereteket szokás definiálni. Keretre nincs szükségünk, de fejlécre igen. A fejléc egy h2-es méretű szöveg lesz, még pedig a következő: „**Jelöltjeid listája**”. Alá húzzunk be a Toolbox-ról egy **PlaceHolder** vezérlőt! Ez is egy ugyanolyan **helyőrző**, mint a mesteroldalnál lévő *ContentPlaceHolder*. Ide fog majd behelyettesítődni az elemsablon. Adjuk azt az azonosítót a PH-nak, hogy **itemsPH**, majd a ListView-nál állítsuk be, hogy ez az a helyőrző, ahová be kell tölteni az ItemTemplate-ben definiált adatokat. Ezt az **ItemPlaceholderID** tulajdonság segítségével szabhatjuk meg. Most itt tartunk:

```

<asp:ListView ID="ListView1" runat="server" ItemPlaceholderID="itemsPH">
  <LayoutTemplate>
    <h2>Jelöltjeid listája</h2>
    <asp:PlaceHolder ID="itemsPH" runat="server"></asp:PlaceHolder>
  </LayoutTemplate>
</asp:ListView>

```

20. Folytassuk az ItemTemplate definiálásával! Az AnonymousView-ban lévő DataList ItemTemplate-jét jelöljük ki, és másoljuk be a ListView LayoutTemplate-je alá. A képet tartalmazó cella **rowspan** tulajdonságát írjuk át 3-ról 4-re, majd a teljes Születési dátumos sorból készítsünk egy másolatot önmaga alá! Itt a *Születési dátum*: helyére írjuk azt, hogy „**Város**”, majd a címke Text tulajdonságát erről: '`<%# Eval("Born_Date", "{0:d}") %>`' módosítsuk erre: '`<%# Eval("City", "{0}") %>`'! A label azonosítója legyen *lbl\_city*! Végül pedig a *Bővebben* HyperLink-nél a megjelenő szöveget írjuk át „**Részletek**”-re és a **NavigationUrl** tulajdonságát pedig adatkössük az alábbi módon: `NavigateUrl='<%# Eval("ID", "~/View_Details.aspx?ID={0}") %>'`! Mint ahogyan azt már korábban is említettem, illetve most már a kódból is látszik, az URL-n keresztül fogjuk átpasszolni a kiválasztott felhasználó azonosítóját. Innen ezt az információt egy úgynevezett **QueryString** objektum segítségével tudjuk majd kinyerni!

21. Már csak az ItemSeparatorTemplate definiálása van hátra a ListView kontrollunkból! Itt összesen egy 300 pixel széles **hr** html taget kell csak elhelyezünk, illetve egy sortörést (br-t). Íme a sablonunk kódja:

```

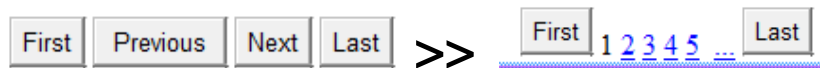
<ItemSeparatorTemplate>
  <hr style="width:300px"><br/>
</ItemSeparatorTemplate>

```

22. Elkészültünk a ListView-val, ezért váltsunk vissza Design vagy Split nézetbe, ahol az alábbi sablont kéne látnunk! Senki ne ijedjen meg, hogy ha nagyon sok sort lát, ez csak az elrendezés leellenőrzéséhez akar segítség lenni!



23. A Default.aspx kinézetét leíró oldalon még egy dolgot kell beállítanunk, még pedig a ListView lapozását! Ehhez a **DataPager** segédvezérlőt fogjuk használni. A Toolbox Data Tab-jéről húzzunk be egyet a ListView alá, és **PagedControlID** tulajdonságát állítsuk be *ListView1*-re! Egyszerre csak 4 felhasználót szeretnénk majd megjeleníteni, ezért a **PageSize**-t vegyük le 10-ről 4-re! Végül a Smart Tag-jében a **Choose Pager Style**-t állítsuk át **Numeric Pager**-re! És készen is vagyunk a főoldal kinézetével, már csak le kell kérni az adatbázisból a megfelelő adatokat.



24. Nyissuk meg a **Default.aspx.cs** fájlunkat és a Page\_Load eseménykezelő függvényéhez az alábbi kódot gépeljük be:

```
if (!User.Identity.IsAuthenticated)
    MultiView1.SetActiveView(MultiView1.Views[0]);
else
{
    MultiView1.SetActiveView(MultiView1.Views[1]);
    ...
}
```

Ez a kódrészlet valósítja meg a LoginView-féle működést, vagyis ha nem vagyunk bejelentkezve, akkor a MultiView vezérlő nulladik View-ja (AnonymousView) lesz az aktív nézet, különben pedig az első.

25. Az adatbázisból azokat a **member**-öket fogjuk lekérdezni, akiknek az **IsFemale** tulajdonsága nem egyezik meg a Profile.Theme-ben beállítottal. Majd pedig a lekért adatokat összekötjük a ListView-val. Tehát így néz ki a teljes else águnk:

```
else
{
    MultiView1.SetActiveView(MultiView1.Views[1]);

    bool isfemale;
    if (Profile.Theme == "Boys") isfemale = false;
    else isfemale = true;

    MembersDataContext mdc = new MembersDataContext();
    var query = from m in mdc.members
                where m.IsFemale != isfemale
                select new { m.ID, m.Nickname, m.Born_Date, m.City, m.Image
    };

    ListView1.DataSource = query;
    ListView1.DataBind();
}
```

25 lépésben, 7 oldalon keresztül, de elkészítettük a főoldalt! Már csak két oldal van hátra ☺, de természetesen ezek jóval könnyebbek lesznek!

## - View\_Details.aspx

Ez az oldal egy kicsit kilóg a többi közül, ugyanis ide közvetlenül menüből nem tudunk eljutni. Ennek az az oka, hogy minden egyes oldalkérésnél szükségünk van egy azonosítóra ahhoz, hogy egyáltalán bármiféle tartalmat is meg tudjunk jeleníteni. Ezért erre az oldalra csak úgy lehet eljutni, ha az alábbi módon hivatkozunk rá: View\_Details.aspx?ID=2.

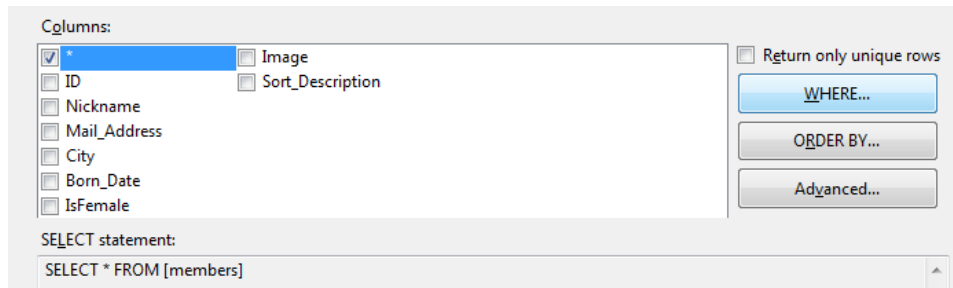
Értelemszerűen a „2” az egyetlen dolog, ami változhat az URL-ben. Az itt megadott számot egy úgynevezett *QueryString* objektumon keresztül tudjuk majd elérni.

Mint ahogy az oldal nevéből is következik, itt minden adatot meg fogunk jeleníteni az adott felhasználóról. Ehhez egy **FormView** vezérlőt fogunk használni az egyszerűség kedvéért. Azért válasszuk ezt a kontrollt, mivel a FormView egyszerre csak egyetlen rekordot tud megjeleníteni (ellentétben a ListView-val, illetve a DataList-tel, stb.).

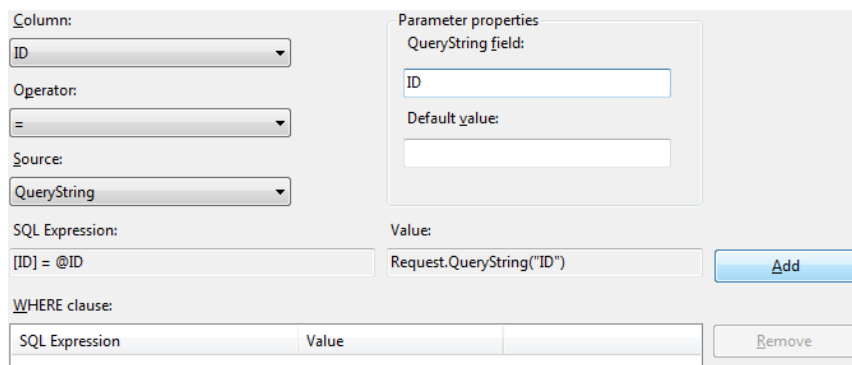
1. Húzzunk a formunkra egy FormView vezérlőt a Toolbox Data Tab-jéről! A Smart tag-jéből a **Choose Data Source**-nál válasszuk a *New data source* lehetőséget! Itt is szintén

egy SqlDataSource-t hozunk létre (felül válasszuk a DataBase-t), és azt a nevet adjuk neki, hogy **SDS\_Details**! A kövi oldalon **ASPNETDBConnectionString**-et válasszuk, majd **Next**!

2. A lekérdezést összeállító oldalon kattintsuk be a \* jelölőnégyzetet, ugyanis az összes adatra szükségünk lesz. Ezek után klikkeljük a **WHERE** gombra, hogy beállítsuk, minket csak az URL-ben átpasszolt azonosítójú felhasználó érdekel!



Az **Add Where Clause** felpattanó ablakban a **Column** legördülő-listából válasszuk az ID-t! Az alatta lévő = operátor nekünk megfelel, ezért a **Source**-nál a **QueryString** lehetőséget válasszuk ki! Jobb oldalt ilyenkor a **Parameter properties** doboz kicsit átalakul. A **QueryString field**-hez írjuk be, hogy ID! (Erre azért van szükség, mert akár több kulcs-érték párt is átpasszolhatunk az URL-ben, ilyenkor & jellel kell elválasztani őket egymástól pl.: Who.aspx?Name=Peet&Age=20). Kattintsunk az **Add** gombra! Ezzel hozzáadtuk a lekérdezésünk WHERE részéhez az új szűrő feltételt. Végül klikk az **OK**-ra!



3. Kattintsunk a **Next**-re, így a lekérdezés-tesztelő oldalra jutunk. Ha most rá klikkelünk a **Test Query**-re, akkor egy ablakban meg kell adnunk a lekérdezendő felhasználó azonosítóját. Próbaként írjuk be az 1-et! Ha megjelent Teszt Elek barátunk, akkor kattintsunk a **Finish**-re!

4. Hasonlóképpen, mint a **DataList**-nél, itt is létrejött az alapsablon. Sőt nem csak egy **Template** keletkezett automatikusan, hanem egyből 3 (**EditItemTemplate**, **InsertItemTemplate**, **ItemTemplate**)! Mivel csak a saját adatainkat szerkeszthetjük majd, illetve van egy külön regisztrációs oldalunk, ezért felesleges az első két sablon. Váltunk át **Source** nézetbe és töröljük ki mindkettőt! Az alapból generált **ItemTemplate** se megfelelő most nekünk, ezért a tag-en belüli részt töröljük is ki! A **Default.aspx** oldalról a **ListView ItemTemplate**-jét másoljuk át egy az egyben ide!

5. Ez azért volt jó húzás, mivel így csak minimális módosításokat kell elvégeznünk. Először is a táblázat szélességét vegyük fel 350-ről 550-re! Majd a képet tartalmazó cella szélességét is írjuk át 150-ről 300-ra! Az **Image ImageUrl** tulajdonságának az értékében a **Thumbs**-t írjuk át **Normals**-ra!

6. Hozzuk végre létre az **Uploaded\_Images** mappát (jobb klikk a projekten, majd **New Folder**) és benne egy **Thumbs**, illetve egy **Normals** könyvtárat! Az egyszerűség kedvéért négyzetes képet fogunk csak használni a demo-ban. A Normals mappába ezért a 300\*300-as képek kerülnek majd, a Thumbs-ba pedig a generált 150\*150-esek. Készítsünk egy normál, illetve egy bélyeg méretű képet azoknak, akik nem akarnak magukról képet feltölteni, és nevezzük el mindkettőt **no\_image.jpg**-nek! Importáljuk be a képeket az **Add Existing Item...** ablak segítségével!

7. Most a Részletek HyperLink-et alakítsuk egy kicsit át! A megjelenő szöveg legyen: „**E-maill küldök neki**”, majd az adatkötést az alábbira módosítsuk: `NavigateUrl= '<%# Eval("Mail_Address", "mailto:{0}") %>'!`

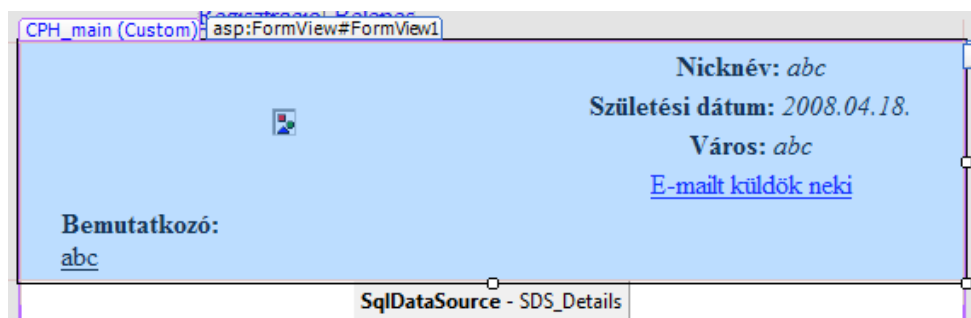
8. Végül adjunk egy teljesen új sort a táblázat legvégéhez! Hozzuk létre benne egy cellát, majd állítsuk be a **colspan** tulajdonságát **2**-re! Erre azért van szükség, mivel alapból két oszlopunk van, de mi ebben a sorban csak egyetlen cellát akarunk, ezért úgymond összevonjuk az oszlopokat. A cella szélessége legyen 500, legyen továbbá balra igazított, és legyen a FormView-nak a bal szélétől 20 pixellel eltolva!

Ebben a pontban az alábbi markup-ot raktuk össze:

```
<tr>
  <td colspan="2" style="width:500; text-align:left; padding-left:20px">
    </td>
</tr>
</table>
```

9. Írjuk bele a cellába, hogy „**Bemutakozó:**”, állítsuk félkövérre, majd törjük meg a sort (br)! Adjunk egy címkét is a cellához, legyen az azonosítója **lbl\_description**, majd a Text tulajdonságát kössük össze a Sort\_Description mezővel: `Text= '<%# Eval("Sort_Description") %>'!` (Ha az Eval-nak nem adunk meg második paramétert, akkor az alapértelmezetten a {0} lesz!)

Végül utolsó simításként a címke **Font-Underline** tulajdonságát állítsuk **True**-ra! Így olyan hatást fog kelteni, mintha egy füzet lapjára lett volna írva a bemutatkozó szöveg.



10. És el is készültünk a View\_Details-zel, így már nézegethetjük más felhasználók adatait, de még nem tudjuk módosítani a sajátjainkat, illetve még képet se tudunk feltölteni. Nos, örömmel jelentem ennyit kell már csak megvalósítanunk a MyProfile oldalon és el is készült a portálunk! Kitartás, már tényleg nincs sok hátra! ☺

## - MyProfile.aspx

A MyProfile oldal tartalma három részből fog felépülni. Az egyik részében a regisztrációnál kitöltött (kötelezően megadott) adatokat tudjuk majd módosítani. Egy másik részében az

opcionális adatok megadásához fogunk egy kényelmes felhasználói felületet biztosítani (bemutakozó írása, képfeltöltés). Végül pedig lesz egy szekciónk, ahol a Témák között tudunk majd váltogatni. Értelemszerűen valós alkalmazás esetén az utolsót nem kell megvalósítanunk, mivel a DEMO alkalmazásba is csak azért kerül bele, hogy ki tudjuk próbálni a különböző témákat ki-, illetve bejelentkezések nélkül!

1. Először készítsük el a témaváltó részét az oldalunknak! Ezt egy egyszerű legördülő-lista segítségével fogjuk megvalósítani. Húzzunk egy DropDownList-et, illetve egy Button-t a Content-ünkbe, majd az egész elé írjuk oda félkövéren, hogy „**Választott téma:**”! A legördülő-listánkat a regisztrációs oldalnál láttak alapján töltsük fel az alábbi két lehetőséggel: **Boys, Girls!** (Items tulajdonság >> **ListItem Collection Editor** ablak >> *Add* gomb ...). A gombunkon megjelenő szöveg legyen az, hogy „**Beállít**”!

Ezek után valahogy így néz ki a markup kódunk:

```
<b>Választott téma: </b>
<asp:DropDownList ID="ddl_theme" runat="server">
  <asp:ListItem>Girls</asp:ListItem>
  <asp:ListItem>Boys</asp:ListItem>
</asp:DropDownList>
<asp:Button ID="btn_save_theme" runat="server" Text="Beállít" />
```

2. Most a gombra való kattintáshoz rendeljünk kódot! Kattintsunk kétszer a Button vezérlőnkre, ilyenkor a **DefaultEvent** eseményére iratkozunk fel, vagyis létrejön a **btn\_save\_theme\_Click** eseménykezelő függvény. Ide az alábbi kódot csattogjuk be:

```
protected void btn_save_theme_Click(object sender, EventArgs e)
{
    Profile.Theme = ddl_theme.SelectedValues;
    Response.Redirect("~/Default.aspx");
}
```

Először elmentjük a Profil Theme tulajdonságába a legördülő-listából kiválasztott témát. Utána átirányítjuk a felhasználót a főoldalra. Utóbbira azért van szükség, mivel az oldal eseményeihez rendelt kódok (pl.: **Page\_Load**, **Page\_PreInit**) mindig hamarabb hajtódnak végre, mint a **PostBack**-et kiváltó esemény eseménykezelője (jelen esetben ez a gomb Click eseménye)! A **PostBack** egy olyan akció, amely a teljes oldal tartalmát visszaküldi a szervernek feldolgozásra. Tehát ahhoz, hogy érvényre jusson az új téma, át kell irányítanunk a felhasználót, hogy ismét végrehajtsódjon a **Page\_PreInit**, de már az új **Profile.Theme** értékkel!

3. A témaválasztóval készen is vagyunk, folytassuk a kötelezően kitöltendő adatok módosításával! Azért, hogy ez a két rész ne csak logikailag, hanem fizikailag is külön legyen választva, helyezünk egy 500 széles **hr** tag-et, illetve egy **br**-t a gomb kontroll alá!

4. Ezek után szúrjunk be egy 4\*2-es (sor\*oszlop) 350 széles táblázatot! Az első oszlop legyen jobbra igazított és 120 széles, a második pedig balra igazított! Az elsőbe az alábbi szövegeket írjuk fentről lefelé haladva: „**Nicknév:**”, „**Város:**”, „**E-mail cím:**”, „**Születési dátum:**”! A jobb oldali oszlop mindegyik cellájába húzzunk be egy-egy **TextBox**-ot, vegyük fel a szélességüket 200-ra, és az alábbi azonosítókat adjuk nekik (fentről-lefelé haladva): **txtbox\_name**, **txtbox\_city**, **txtbox\_mail**, **txtbox\_date**! Ha ezzel megvagyunk, akkor most valahogy így kell kinéznie a táblázatunknak:

Nicknév:	<input type="text" value="asp:TextBox#txtbox_city"/>
Város:	<input type="text"/>
E-mail cím:	<input type="text"/>
Születési dátum:	<input type="text"/>

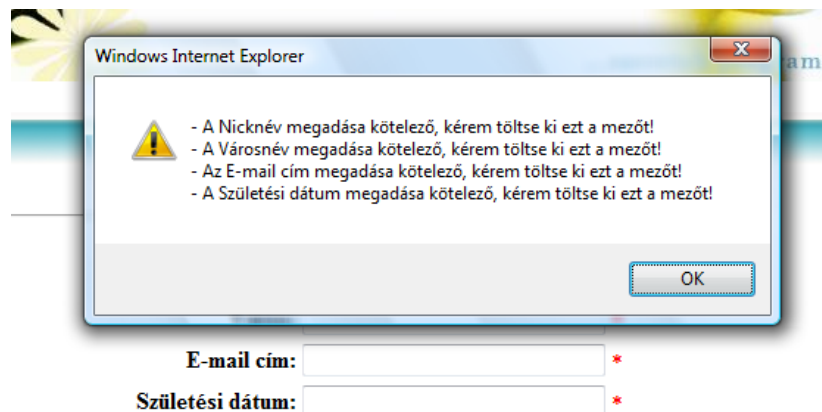
5. Mindegyik textbox mellé tegyünk egy-egy **RequiredFieldValidator**-t, és a **ControlToValidate** tulajdonságokon keresztül kössük össze őket a mellettük lévő szövegdobozokkal! A **Text** tulajdonságokat állítsuk \*-ra, az **ErrorMessage**-eket pedig valami hasonlóra: „**A Nicknév megadása kötelező, kérem töltsse ki ezt a mezőt!**” (Értelemszerűen a Nicknevet mindig helyettesítsük be az aktuális sornak megfelelőre!)

6. Mivel a Validátor vezérlők ellenőrző rutinja mindenPostBack-et kiváltó esemény előtt automatikusan lefut, ezért szükségünk van arra, hogy beállítsuk a **ValidationGroup** tulajdonságukat. Ugyanis, ha ezt beállítjuk, akkor csak az azonos ValidationGroup-ban lévő validátorok fognak kiértékelődni akkor és csak akkor, amikor a szintén azonos érvényesítési csoportban lévő gombra kattintunk. Magyarul, ha ezt beállítjuk, akkor a témaválasztónál lévő btn\_save\_theme gombra való klikkeléskor nem fognak feleslegesen validálódni a szövegdobozok. Ezért állítsuk be az összes validátor ValidationGroup tulajdonságát **requireddatas**-ra!

7. A táblázat alatt hagyjunk ki pár sort, majd helyezzünk el ide egy gombot, illetve egy **ValidationSummary** vezérlőt a Toolbox **Validation** Tab-jéről! A gomb Text-je legyen „**Mentés**”! A ValidationSummary kontroll arra hivatott, hogy a hibát észlelt validátor vezérlők hibaüzeneteit összegyűjtse és egy adott helyen felsorolva megjelenítse őket. Ez a vezérlő azért fontos most a számunkra, mivel a hosszú hibaüzenetek miatt szétszúszhatna a táblázatunk, ha ott helyben jelenítenénk meg őket! Ezt mi nem szeretnénk, ezért használjuk a ValidationSummary-t. A hibaüzeneteket egy felugró böngészőablakban szeretnénk megjeleníteni, ezért a **ShowMessageBox** tulajdonságát állítsuk *True*-ra, a **ShowSummary**-t pedig *False*-ra!

Végül pedig ne felejtsük el beállítani a Button és a ValidationSummary ValidationGroup tulajdonságát **requireddatas**-ra!

8. Ha most elindítjuk a MyProfile oldalt, és egyből a *Mentés* gombra kattintunk, akkor valami hasonlót kell látnunk:



Természetesen a születési dátum, illetve az e-mail cím szabályosságát is illene ellenőrizni, de mivel ezek nem hoznak szinte semmi újdonságot, ezért most eltekintünk a DEMO-ban az implementálásuktól!

9. Az oldalunk utolsó részével az opcionálisan kitölthető adatok megadásával folytassuk tovább a fejlesztést! Először is töröljük ki a táblázat és a gomb közötti felesleges üres sorokat úgy, hogy legyen egy br a táblázat után, és másik kettő pedig a Mentés gomb előtt! Közéjük írjuk be h3-as betűmérettel: „**Opcionális adatok**”, majd húzzunk be egy TextBox vezérlőt! Állítsuk be a szélességét 350pixelre, a magasságát pedig 100-ra! Azt szeretnénk, hogy a


szövegdobozunk többsoros legyen, ezért a **TextMode** tulajdonságát állítsuk át **MultiLine**-ra! Adjuk azt az azonosítót neki, hogy *txtbox\_description*, majd közvetlen a szövegdoboz fölé írjuk: „**Bemutakozó (max. 1000 karakter!)**”. A zárójeles rész legyen egy betűmérettel kisebb, mint alából lenne!

### Opcinális adatok

**Bemutakozó**  
*(max 1000 karakter!)*

10. A szövegdoboz alatt hagyjunk ki egy sort, majd írjuk félkövérrel, hogy „**Jelenlegi kép**” és alá húzzunk be egy Image vezérlőt! Mivel kódból fogjuk majd beállítani a képet, ezért adjuk neki valami értelmesebb nevet az *Image1*-nél, mondjuk *img\_now*! Ismét hagyjunk ki egy sort és írjuk azt, hogy „**Új kép feltöltése:** ”, majd húzzunk ide a **ToolBox Standard** Tag-jéről egy **FileUpload**-ot és adjuk azt az azonosítót neki, hogy *fileup\_img*!

**Jelenlegi kép**



Új kép feltöltése:

- **Error message 1.**
  - **Error message 2.**
- 

11. Elkészültünk a MyProfile felhasználói felületével, már csak a Mentés, illetve a Page\_Load-hoz tartozó kódokat kell megírunk, kezdjük az utóbbival! Azt szeretnénk, hogy az oldal első betöltődésekor minden TextBox töltődjön fel az adatbázisban lévő aktuális adatokkal. Azért írtam, hogy csak az első alkalommal, mivel ha PostBack történik, akkor utazik az összes adat a szerverre, így az ASP.NET az oldal újragenerálásakor automatikusan már az új adatokkal fogja feltölteni a megváltozott szövegdobozokat. Tehát azt szeretnénk, hogy a Page\_Load-ban lévő kód csak egyszer fusson le, a legelső alkalomkor. Ehhez egy egyszerű vizsgálatra van szükségünk, meg kell határoznunk, hogy az oldalra PostBack-kel jutottunk-e, vagy ez még csak az első oldalkérésünk volt. Ezt a Page **IsPostBack** tulajdonságával nagyon egyszerűen le tudjuk kérdezni.

12. Tehát az első oldalkéréskor beállítjuk a témaválasztó legördülő-listáját a profilban lévő éppen aktuális témára. Ezek után a felhasználónév alapján lekérünk minden információt az adatbázisból, majd egyesével összekötjük a megfelelő adatokat a megfelelő szövegdobozokkal. Íme ennek a kódja:

```
if (!IsPostBack)
{
    ddl_theme.SelectedValue = Profile.Theme;
    MembersDataContext mdc = new MembersDataContext();
    IQueryable<member> mems;
    member mem = new member();
```

```

    mems = from m in mdc.members
           where m.Nickname == User.Identity.Name
           select m;
    mem = mems.First();
    txtbox_name.Text = mem.Nickname;
    txtbox_name.DataBind();
    txtbox_city.Text = mem.City;
    txtbox_city.DataBind();
    txtbox_mail.Text = mem.Mail_Address;
    txtbox_mail.DataBind();
    txtbox_date.Text = mem.Born_Date.ToShortDateString();
    txtbox_date.DataBind();
    txtbox_description.Text = mem.Sort_Description;
    txtbox_description.DataBind();
    img_now.ImageUrl = "~/Uploaded_Images/Thumbs/" + mem.Image;
    img_now.DataBind();
}

```

Mint minden normális adatbázis lekérdezés, ez is egy *rekordgyűjteménnyel* tér vissza, csak itt a rekordgyűjteményünk egy **IQueryable generikus tömb**, aminek a típusa *member*. Mivel tudjuk, hogy csak egyetlen ilyen felhasználó létezik, ezért elkérjük a gyűjteménytől az első és egyben egyetlen elemét. Majd ezek után beállítjuk a TextBox-ok (illetve az egyetlen Image vezérlő) értékeit és a **DataBind** metódusokkal pedig adatkötjük őket.

(Amennyiben **RangeValidator**-t is használunk a dátumellenőrzéséhez, akkor a **ShortDate** formátum végén lévő pontot le kell vágnunk, hogy elfogadja validnak a validátor. Ezt az alábbi módon tehetjük meg: `txtbox_date.Text = txtbox_date.Text.Remove(10);`)

13. Legvégül a Mentés gomb Click eseménykezelőjének a kódját kell implementálnunk! Tároljuk el az adatokat változóba, a txtbox\_date szövegét ne felejtjük el átkonvertálni **DateTime** típusúra! Az opcionális adatok közül csak a képfeltöltésnél kell ellenőrzést végeznünk. Ha a feltöltendő fájl mérete nagyobb, mint 0, akkor a felhasználó adott meg új képet. Kérjük le az adatbázisból a saját rekordunkat, amelyet egy member fog majd reprezentálni, és állítsuk be a tulajdonságait! Íme a kódunk:

```

string name = txtbox_name.Text;
string city = txtbox_city.Text;
string mail = txtbox_mail.Text;
DateTime date = Convert.ToDateTime(txtbox_date.Text);
string description = txtbox_description.Text;

if (fileup_img.PostedFile.ContentLength > 0)
{
    string path = Server.MapPath("~/Uploaded_Images/Normals/" +
User.Identity.Name + "_" + fileup_img.FileName);
    fileup_img.PostedFile.SaveAs(path);
    this.CreateThumbnail(path, "~/Uploaded_Images/Thumbs/" +
User.Identity.Name + "_" + fileup_img.FileName);
}

MembersDataContext mdc = new MembersDataContext();
IQueryable<member> mems;
member mem = new member();
mems = from m in mdc.members
       where m.Nickname == User.Identity.Name.ToString()
       select m;
mem = mems.First();

mem.Nickname = name;
mem.City = city;

```

```

mem.Mail_Address = mail;
mem.Born_Date = date;
if(fileup_img.PostedFile.ContentLength >0) mem.Image = User.Identity.Name +
"_" + fileup_img.FileName;
mem.Sort_Description = description;

mdc.SubmitChanges();

img_now.ImageUrl = "~/Uploaded_Images/Thumbs/" + User.Identity.Name + "_" +
fileup_img.FileName;
img_now.DataBind();

```

Egyedül a képfeltöltés igényel némi magyarázatot szerintem. Először egy string-ben összerakjuk a menteni kívánt kép elérési útját, illetve a nevét. A **Server.MapPath** segítségével meghatározzuk a szerveren az alkalmazásunk valós könyvtári elhelyezkedését, majd ezután hozzácsapjuk a paraméterben átadott relatív fájllelési utat. Azért, hogy a felhasználók ne írassák felül más userok képeit, ezért a feltöltött képeket ellátjuk egy prefixszel, ami a felhasználónév lesz! Végül meghívjuk a **CreateThumbnail** sajátfüggvényüket, hogy generáljon a feltöltött képből egy bélyeg méretű változatot!

14. A CreateThumbnail metódusunk 2 paramétert vár, az elsőben a kicsinyítendő kép elérési útját, a másodikban pedig, hogy hova mentse az új képet. A függvény működésének részleteibe nem mennék bele, szerintem eléggé egyértelmű, hogy mit és hogyan csinál. Íme a függvényünk kódja:

```

protected virtual void CreateThumbnail(String srcpath, String destpath)
{
    System.Drawing.Image img = System.Drawing.Image.FromFile(srcpath);
    System.Drawing.Image imgthumb = img.GetThumbnailImage(150, 150, null,
new System.IntPtr(0));
    imgthumb.Save(Server.MapPath(destpath),
System.Drawing.Imaging.ImageFormat.Jpeg);
    img.Dispose();
    imgthumb.Dispose();
}

```

15. **Háromszoros hip hip hurrá!** Ugyanis elkészültünk a portállal!!!! Tudom, hogy hihetetlennek tűnik, de mégis igaz! ☺ Hozzunk létre új felhasználókat, tesztelgessük, nyúzzuk a rendszert, közben pedig örüljünk!

## Továbbfejlesztési lehetőségek

Egyetlen fejlesztői dokumentáció vagy tutorial sem teljes anélkül, hogy nem válaszolunk benne a „merre tovább?” című kérdésre. Ezért íme, néhány továbbfejlesztési javaslat:

- A content oldalakra ráférne egy kis design
- Mindenféle validátor használata mindenhol
- A menünek nevezett hyperlink park lecserélése igazi menu kontrollra, sok új oldal esetén
- Adminisztrációs felület készítése
- Hibakezelések megírása
- Portál AJAX-osítása
- Belső levelező rendszer megvalósítása
- Képgaléria kialakítási lehetőség
- Kommentezési lehetőség
- Fórum, Chat
- ...