

INDULÓ KÉSZLET WEBFEJLESZTŐKNEK

ÚTMUTATÓ AZ AUTOSHOP PÉLDAALKALMAZÁSHOZ

2006. május 10.

Készítette: Gincsei Gábor (MSDN Kompetencia Központ)

Tartalomjegyzék

Tartalomjegyzék.....	2
1. Bevezetés	3
2. A webalkalmazás és az adatbázis létrehozása	4
2.1. A webalkalmazás létrehozása.....	4
2.2. Az Images könyvtár hozzáadása a projekthez.....	5
2.3. Az adatbázis létrehozása Microsoft SQL Server 2005 Express Edition használatával... 5	5
2.4. Az adatbázis létrehozása Microsoft SQL Server 2005 használatával	10
3. Adatkötés.....	14
3.1. Tallózó nézet elkészítése	14
3.1.1. Kategóriák megjelenítése	16
3.1.2. Kiválasztott kategória autóiinak megjelenítése	20
3.2. Új hirdetés feltöltése.....	23
3.3. Keresés	26
4. MasterPage	29
5. Login kontrollok.....	37
6. Témák és bőrök	42
7. WebPartok	43
7.1. Legújabb 5 elemet megjelenítő webkijelző elkészítése	44
7.2. DisplayText.ascx elkészítése.....	45
8. Jogosultságok beállítása.....	47

1. Bevezetés

A 2006. február 9-10-én „Webes megoldások ASP.NET 2.0 alapokon” címen megrendezett előadássorozat gyakorlatain bemutatott példaalkalmazást azzal a céllal készítettük, hogy megmutassuk, miként lehet egy webes alkalmazást gyorsan és hatékonyan elkészíteni ASP.NET 2.0 segítségével. Fontos megjegyezni, hogy annak ellenére, hogy egy teljes rendszer megvalósítására törekedtünk, az elészült alkalmazást (melynek teljes [forráskódja letölthető a devPORTAL-ról](#)) inkább szánjuk példa-, mint mintaalkalmazásnak.

A példaalkalmazás elkészítése során megismerhetjük az ASP.NET 2.0 platform olyan újításait, melyek drasztikusan lerövidítik a webalkalmazások fejlesztésének idejét azáltal, hogy számos olyan megoldást kapunk készen, amit eddig mindig saját kezűleg kellett implementálnunk.

A következőkben a példaalkalmazás elkészítésének részletes ismertetése következik, amely szinte kattintásról kattintásra végigvezet a megvalósítás lépésein. Amennyiben a leírtakkal kapcsolatban bármilyen megjegyzése vagy kérdése van, kérjük, látogasson el a [devPORTAL Webfejlesztés fórumára](#), ahol szívesen várunk minden visszajelzést, és készséggel válaszolunk a felmerülő kérdésekre.

Az MSDN Kompetencia Központ csapata

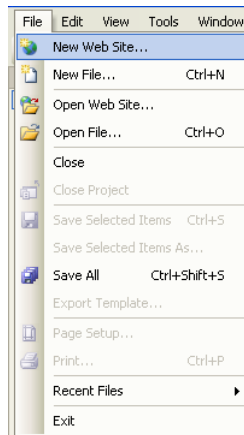
2. A webalkalmazás és az adatbázis létrehozása

Ebben a fejezetben megismerkedünk a legalapvetőbb feladatok megoldásával. Megnézzük, hogy hogyan tudunk webalkalmazást létrehozni, hogyan tudunk az alkalmazásunkhoz már meglévő állományokat hozzáadni (jelen esetben képeket), illetve megnézzük, hogyan tudunk adatbázist létrehozni a portál adatainak a tárolására.

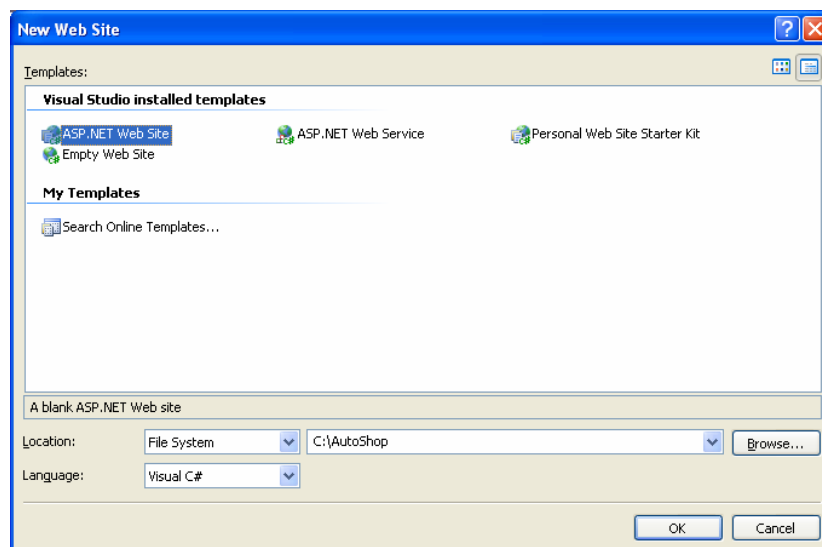
2.1. A webalkalmazás létrehozása

Legelőször nézzük meg, hogyan tudunk egy új weboldalt létrehozni a [Visual Web Developer 2005 Express Edition](#) segítségével. Ez a fejlesztőeszköz a Visual Studio 2005 egy ingyenesen letölthető verziója, mely kizárólag webes fejlesztésre alkalmas.

1. Új website létrehozásához kattintsunk a *File menü* → *New Web Site* menüpontra.



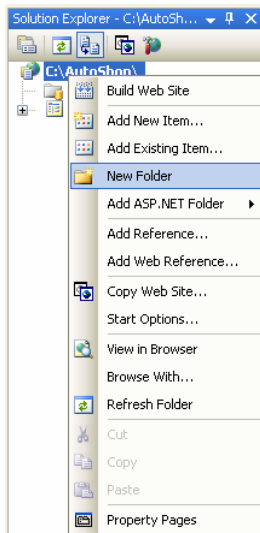
2. A következő képernyőn válasszuk ki, hogy
 - a. Mit szeretnénk készíteni – *ASP.NET Web Site*
 - b. Hogyan kívánjuk elérni az alkalmazás fájljait – *File System*
 - c. Válasszuk ki a programozási nyelvet – *Visual C#*
 - d. És adjuk meg, hogy hol jöjjön létre a webalkalmazás – *C:\AutoShop\Web*



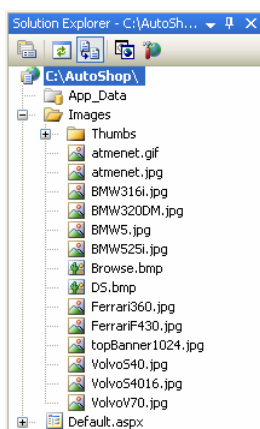
2.2. Az Images könyvtár hozzáadása a projekthez

Ezt követően az Images könyvtárat szeretnénk hozzáadni a meglévő webalkalmazásunkhoz. Ehhez a következőket kell tennünk:

1. Solution Explorer ablakban hozzunk létre egy új könyvtárat Images névvel a C:\AutoShop könyvtár alá.



2. Másoljuk ide a mellékelt zip fájlban található képeket, majd frissítsük a könyvtár tartalmát. Ezek után a következőt kell látnunk a Solution Explorer ablakban.



2.3. Az adatbázis létrehozása Microsoft SQL Server 2005 Express Edition használatával

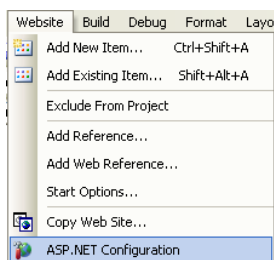
A példaalkalmazáshoz a szükséges adatokat adatbázisban fogjuk tárolni. Ehhez természetesen létre kell hoznunk egy adatbázist, ami két részből áll. Egyrészt szükségünk van a felhasználói adatok eltárolására (felhasználók, profil adatok, jogosultság ...), másrészt az autókra vonatkozó adatokat is el kell tárolnunk (kategória, típus, autó adatai).

Az adatbázis első felének létrehozását a .NET keretrendszer támogatja, hiszen ezekre az információkra majdnem minden webes alkalmazásban szükség van. Azonban az adatbázis másik felét nekünk kell megtervezni és létrehozni. Ez a fejezet ennek az adatbázisnak a létrehozását ismerteti.

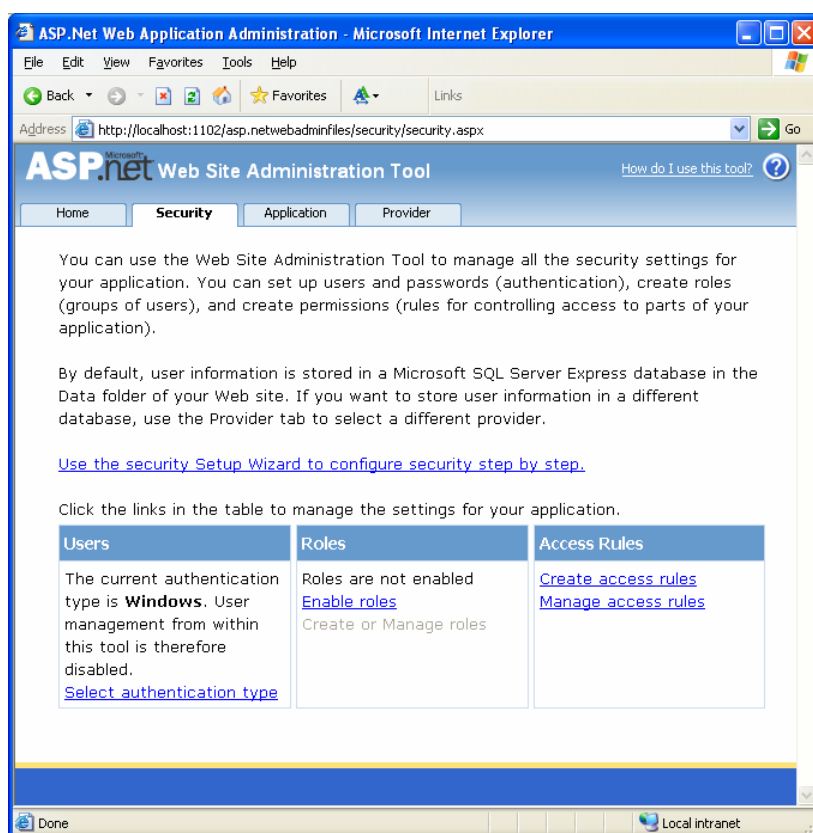
Ahhoz, hogy az ASP.NET 2.0-ba beépített jogosultság- és szerepkörkezelést, illetve a felhasználói profilt használni tudjuk, szükségünk van egy adatbázisra, ahol a keretrendszer eltárolja a fent említett funkcióhoz szükséges információkat. Ezt az adatbázist nagyon

egyszerűen létre tudjuk hozni, abban az esetben, ha Microsoft SQL Server 2005 Express Editiont használunk.

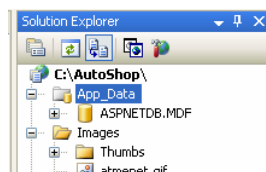
1. Indítsuk el a Web Site Admin Toolt. *WebSite* menü → *ASP.NET Configuration* menüpont.



2. Majd válasszuk ki a *Security* fület.



3. Zárjuk be a Web Site Admin Toolt és frissítsük az *App_Data* könyvtárat a Solution Explorerben. Ekkor meg fog jelenni egy *ASPNETDB.MDF* adatbázis állomány ebben a könyvtárban.

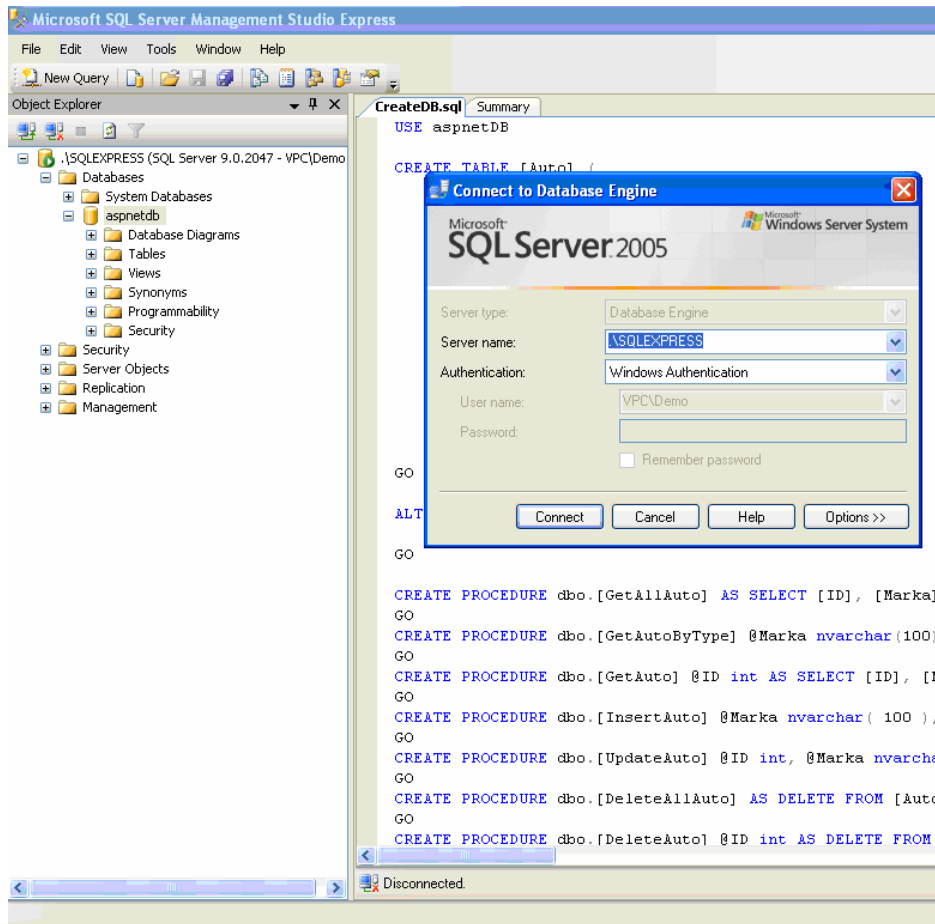


4. Ezzel el is készítettük az adatbázisunknak azt a felét, amiben a keretrendszer eltárolja a legfontosabb adatokat, mint például a felhasználók adatait, a szerepköröket, illetve a profil információkat.

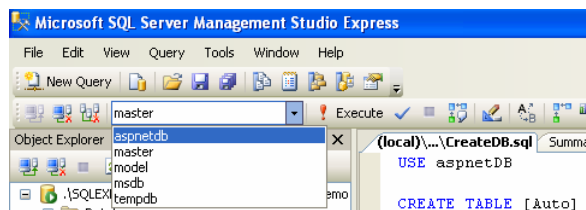
5. Egészítsük ki a létrehozott adatbázis az alkalmazás specifikus táblákkal és tárolt eljárásokkal. Ehhez le kell futtatnunk a zip fájlban található SQL szkripteket. Ezt a következőképpen tehetjük meg:
 - a. Indítsuk el az *SQL Server Management Express*-t, amit a Start menü → All programs → Microsoft SQL Server 2005 menüpont alatt találhatunk meg.
 - b. Csatlakozzunk a felkínált adatbázishoz Windows hitelesítéssel. Az adatbázis szerver neve: \SQLEXPRESS legyen. A pont a lokális gépet jelenti, az SQLEXPRESS pedig az, hogy az ilyen nevű adatbázis szerverhez szeretnénk csatlakozni. Ezt azért így kell megadni, mert az SQL Express alapértelmezés szerint ezzel a névvel kerül telepítésre, hogy ne ütközzön a neve egy esetlegesen feltelepített SQL Server 2000 / 2005-tel.



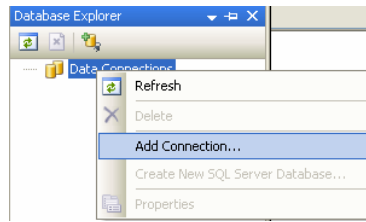
- c. Nyissuk meg a zip állományban található CreateDB.sql fájlt, a File menü → Open parancsával. Ekkor az alábbi képernyővel fogunk találkozni:



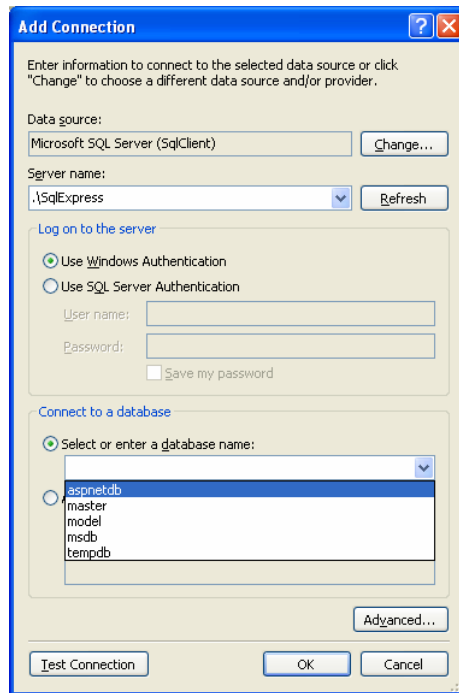
- d. Csatlakozunk ismét az adatbázisszerverhez, a *Connect* segítségével.
- e. Majd fent a toolbaron állítsuk be, hogy az aspnetdb adatbázison szeretnénk lefuttatni a szkriptet, majd kattintsunk az Execute ikonra, ahogy azt az alábbi ábra mutatja.



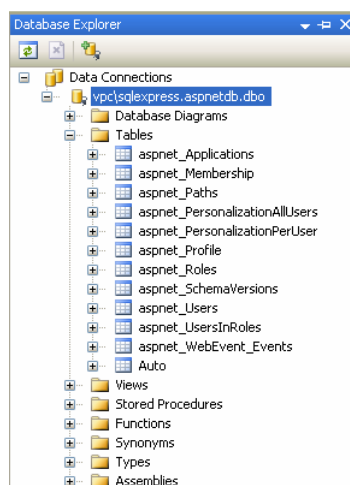
- f. Majd ismételjük meg a fenti lépéseket (c. – e.) az AddAutos.sql állományra is.
6. Zárjuk be az *SQL Server Management Express*-t.
 7. A Visual Studio 2005 Express Edition segítségével ellenőrizzük le, hogy valóban létrejött-e a megfelelő táblák.
 - a. Nyissuk meg a Database Explorer ablakot
 - b. Jobb gomb a DataConnection-ön, és válasszuk az *Add Connection... -t*



- c. Adjuk meg az SQL szerver nevét: `.\SqlExpress`, majd a legördülő menüből válasszuk ki az `aspnetdb` adatbázist, majd kattintsunk az OK gombra.

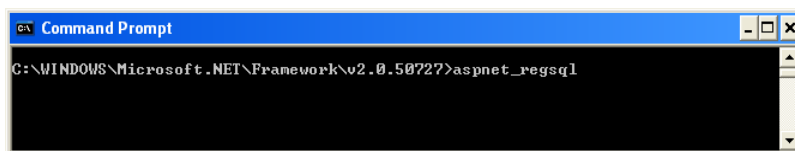


- d. Ezzel sikeresen hozzáadtunk egy új adatbázis kapcsolatot a Database Explorer ablakhoz.
- e. Kattintsunk az adatbázis neve előtti „+” jelre, majd hasonlóan nyissuk meg a Tables-t. A következőt kell látnunk:

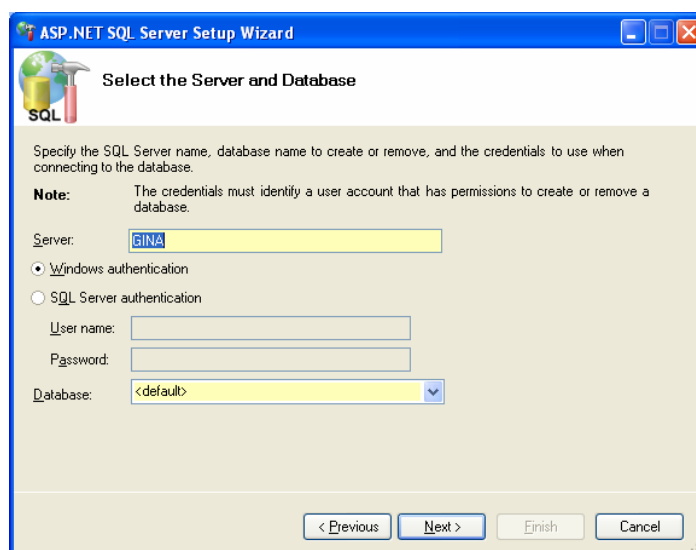


2.4. Az adatbázis létrehozása Microsoft SQL Server 2005 használatával

1. Ahhoz, hogy az ASP.NET 2.0-ba beépített jogosultság-, szerepkörkezelést és a profilt használni tudjuk, szükségünk van egy adatbázisra, ahol a keretrendszer eltárolja a fent említett funkcióhoz szükséges információkat. Ezt az adatbázist nekünk kell létrehozni, azonban van egy nagyon jól használható varázsló, ami ebben segítségünkre lesz.
2. Parancssorból indítsuk el az *aspnet_regsql.exe*-t, amit a `c:\Windows\Microsoft.NET\Framework\v2.0.50727` könyvtár alatt találunk.



3. Az előbukkanó varázslóban kattintsunk a Next gombra.
4. Ezután kiválaszthatjuk, hogy az eszköz új adatbázist hozzon létre, vagy távolítsa el a meglévőt. Hagyjuk kiválasztva a *Configure SQL Server for application services* opciót.
5. A következő lépésben megadhatjuk, hogy
 - a. melyik szerveren hozza létre az adatbázist,
 - b. milyen azonosítást használjon,
 - c. és milyen névvel hozza létre az adatbázist. Alapértelmezés szerint az adatbázis neve aspnetdb lesz.

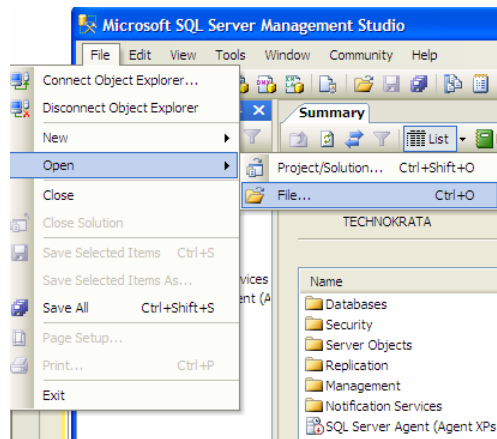


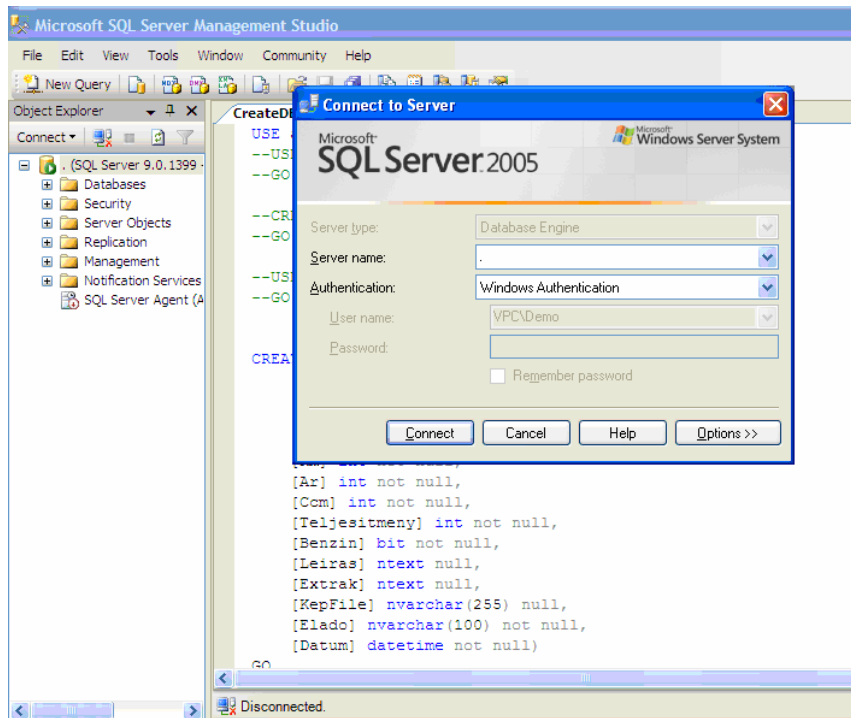
6. Ez követően a varázsló összefoglalja, hogy milyen beállításokkal fogja létrehozni az adatbázist.
7. Majd a végén közli, hogy sikerült létrehozni az adatbázist.
8. Ezzel el is készítettük az adatbázisunknak azt a felét, amiben a keretrendszer eltárolja a legfontosabb adatokat, úgy mint a felhasználók adatait, a szerepköröket, illetve a profil információkat.
9. Egészítsük ki a létrehozott adatbázis az alkalmazás specifikus táblákkal és tárolt eljárásokkal. Ehhez le kell futtatnunk a zip fájlban található sql szkripteket. Ezt a következőképpen tehetjük meg:

- a. Indítsuk el az *SQL Server Management Studio*-t, amit a Start menü → All programs → Microsoft SQL Server 2005 menüpont alatt találhatunk meg.
- b. Csatlakozunk a felkínált adatbázishoz Windows Autentikációval. A szerver neve ebben az esetben „.” ami a lokális SQL szervert jelenti.

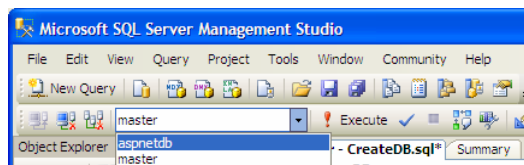


- c. Nyissuk meg a zip állományban található CreateDB.sql fájlt, a File menü → Open → File parancsával.



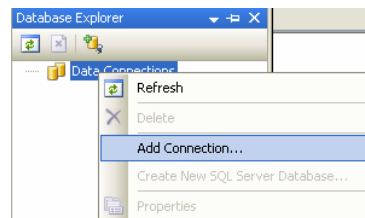


- d. Csatlakozzunk ismét az adatbázisszerverhez, a *Connect* segítségével.
- e. Majd fent a toolbaron állítsuk be, hogy az *aspnetdb* adatbázison szeretnénk lefuttatni a szkriptet, majd kattintsunk az *Execute* ikonra, ahogy azt az alábbi ábra mutatja.

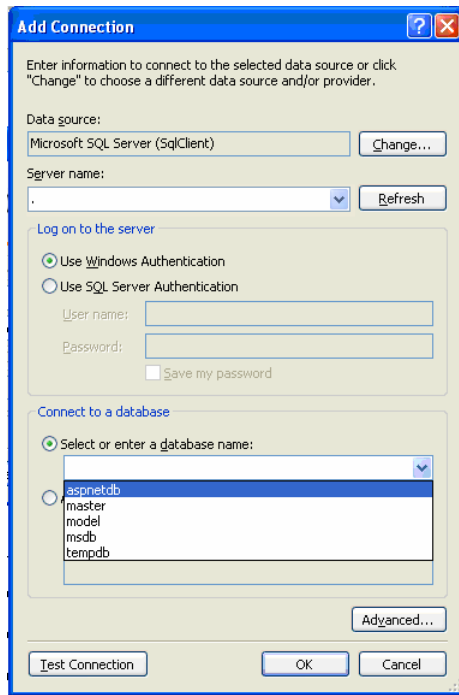


- f. Majd ismételjük meg a fenti lépéseket (c. – e.) az *AddAutos.sql* állományra is.
10. Zárjuk be az *SQL Server Management Studio*-t.
11. A *Visual Studio 2005 Express Edition* segítségével ellenőrizzük le, hogy valóban létrejött-e a megfelelő táblák.

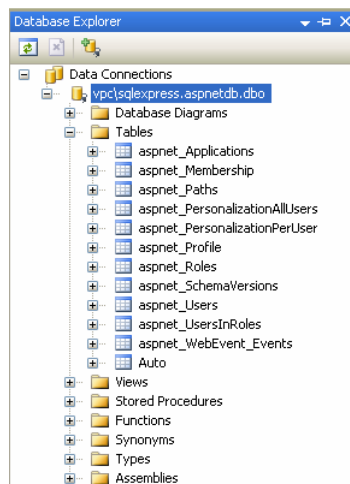
- a. Nyissuk meg a *Database Explorer* ablakot
- b. Jobb gomb a *DataConnection*-ön, és válasszuk az *Add Connection...* -t



- c. Adjuk meg az SQL szervert nevet: *.\SqlExpress*, majd a legördülő menüből válasszuk ki az *aspnetdb* adatbázist, majd kattintsunk az *OK* gombra.



- d. Ezzel sikeresen hozzáadtunk egy új adatbázis kapcsolatot a Database Explorer ablakhoz.
- e. Kattintsunk az adatbázis neve előtti „+” jelre, majd hasonlóan nyissuk meg a Tables-t. A következőt kell látnunk:



Ezzel el is készítettük az alkalmazás alapjául szolgáló adatbázist, és teszt adatokkal is feltöltöttük.

3. Adatkötés

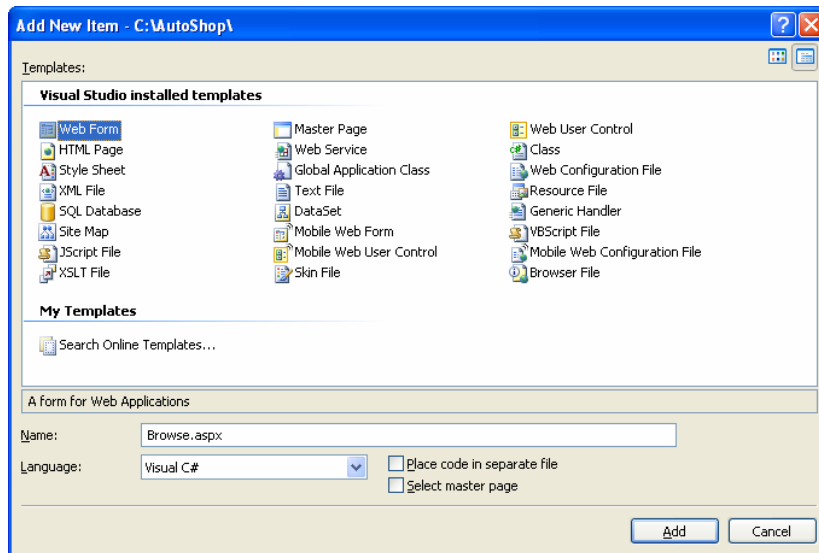
A webalkalmazások fejlesztői különösen érdekeltek olyan automatizált módszerek kidolgozásában, melyek segítségével az adatokat – mint például az adatrekordok és az adatmezők – tartósan és dinamikusán grafikus elemekhez köthetők. Erre az egyik leghatásosabb módszer az adatkötések alkalmazása.

Adatkötés alatt azt a folyamatot értjük, mely segítségével kiolvassuk a szükséges adatokat az adatforrásból, és dinamikusán egy vizuális elem egyik tulajdonságához „kötjük”, azaz a vizuális elem szóban forgó tulajdonságának értékét az adatforrás bizonyos adataitól tesszük – kölcsönösen – függővé. Az adatkötés milyensége és bonyolultsága egyaránt függ az adatforrástól – ami nem feltétlenül kell, hogy adatbázis legyen – és az adatokat megjelenítő vizuális elemtől. Így beszélhetünk egyszerű és összetett, egyirányú és kétirányú adatkötésről. Az egyszerű adatkötés az adatforrásnak csak egyetlen adatát, vagy adatrekordját képes egyszerre megjeleníteni, és általában az adatkezelő komponens a felelős az adatforrásban történő „lapozásért”, azaz az adatok cseréléséért. Ezzel szemben az összetett adatkötés több rekordot képes egyszerre megjeleníteni, függetlenül attól, hogy az adathoz kötött vizuális elem automatikusan, vagy programkódból végzi az adatlapozást. Az egyirányú adatkötés esetében az adat megváltozásával megváltozik az adat vizuális megjelenítése is.

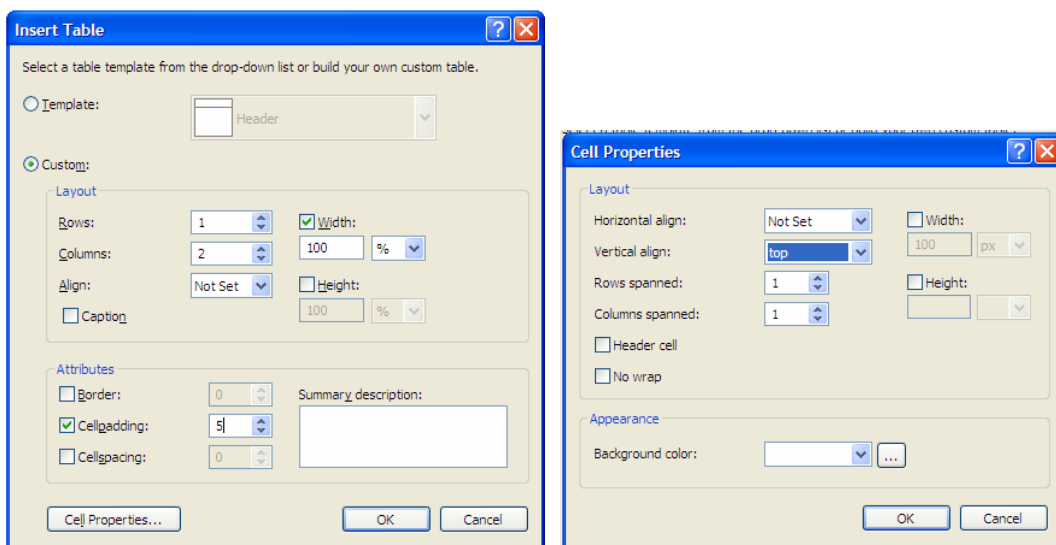
A kétirányú adatkötés ezen túl, az adat vizuális megjelenítésében véghezvitt változásokat – automatikusan vagy programkód segítségével – az adatforrásban is elvégzi.

3.1. Tallózó nézet elkészítése

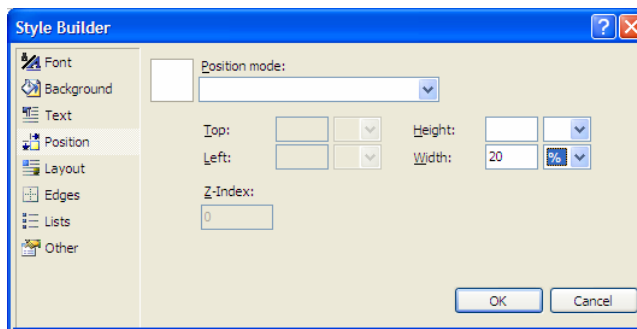
1. Adjunk a weboldalunkhoz egy új oldalt a File menü → New File... menüpont segítségével.
2. A következő képernyőn állítsuk be, hogy az oldal neve legyen *Browse.aspx*, típusa pedig *Web Form*.



3. Szúrjunk be egy 1 soros és 2 oszlopos táblázatot (Layout menü → Insert Table menüpont), melynek tulajdonságai legyenek az alábbiak:
 - a. A táblázat 100% széles.
 - b. A cellák között legyen 5 pixel kihagyás.
 - c. A cellák tartalma függőlegesen fentre legyen igazítva.



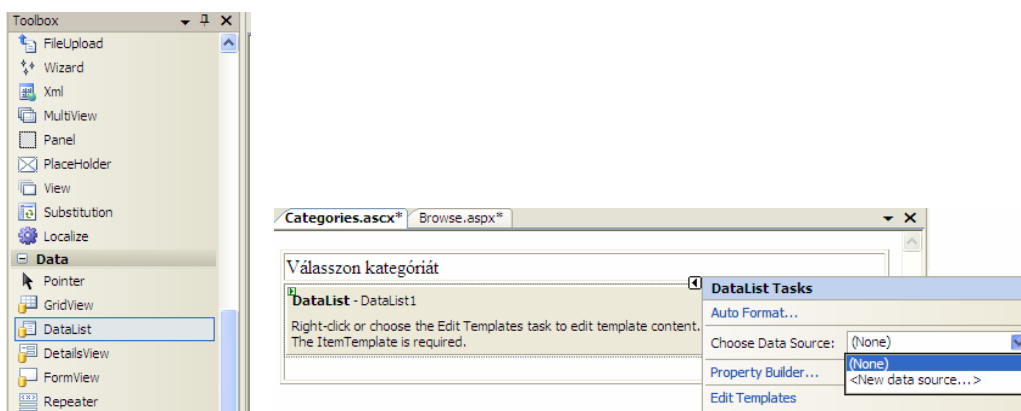
4. A bal oldali cella tulajdonságai között, amit úgy érhetünk el, hogy jobb gombbal kattintunk a cellán, majd kiválasztjuk a properties menüpontot, keressük meg a *Style*-t és kattintsunk a ...-ra, majd az előbukkanó ablakban állítsuk be a szélességet 20%-ra, ahogy azt az alábbi ábra szemlélteti.



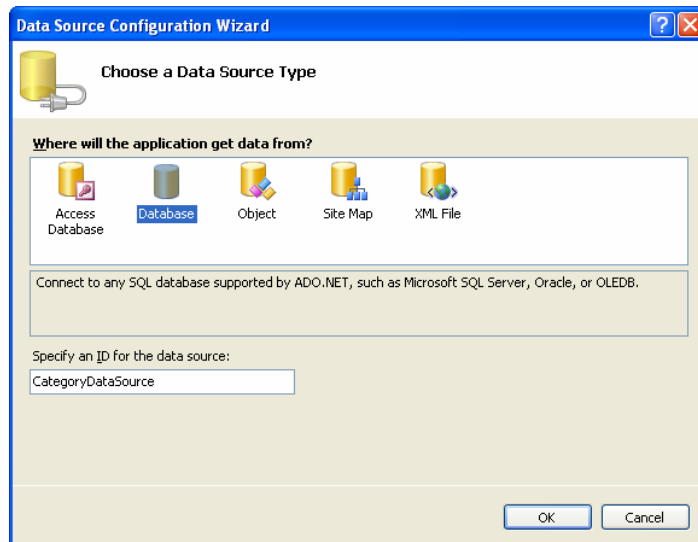
5. Ismételjük meg a 3. lépést a jobb oldali cellára is, csak ott 80%-ot állítsunk be szélességnek.
6. Az elkészített táblázat mindkét cellájába egy-egy modul fog kerülni. A bal oldaliba a *Categories.ascx*, a jobb oldaliba a *CarList.ascx*. A következő fejezetekben elkészítjük ezt a két modult, elsőként a kategória megjelenítőt.

3.1.1. Kategóriák megjelenítése

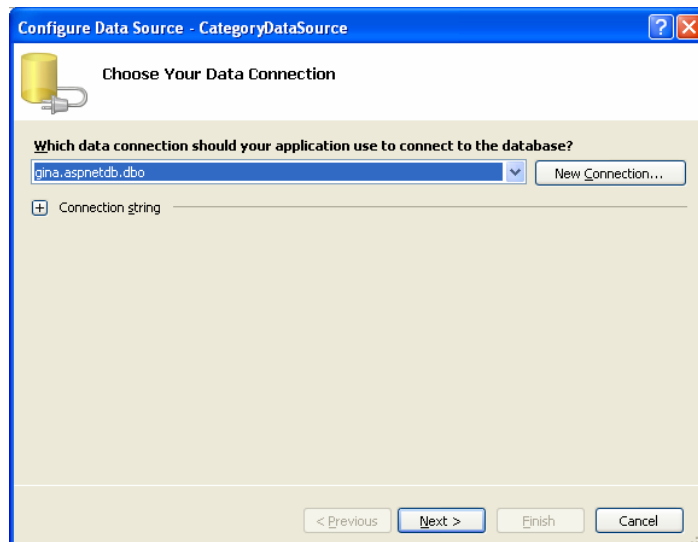
1. Adjunk a webhelyhez egy új Web User Controlt *Categories.ascx* névvel.
2. Szúrjunk be egy kétsoros táblázatot, ami 100% széles, és a keret vastagsága 1 pixel.
3. A felső cellába írjuk be a modul címét: *Válasszon kategóriát* és a cella tulajdonságai között a *Class*-t állítsuk *ModuleHeader*-re. Ezzel adjuk meg, hogy milyen CSS osztályt használjon.
4. Az alsó cellára ugyanígy állítsuk be a *ModuleBody* CSS osztályt, majd a Toolboxról húzzunk be egy *DataList* vezérlőelemet *ctlCategories* névvel ebbe a táblázat cellába.



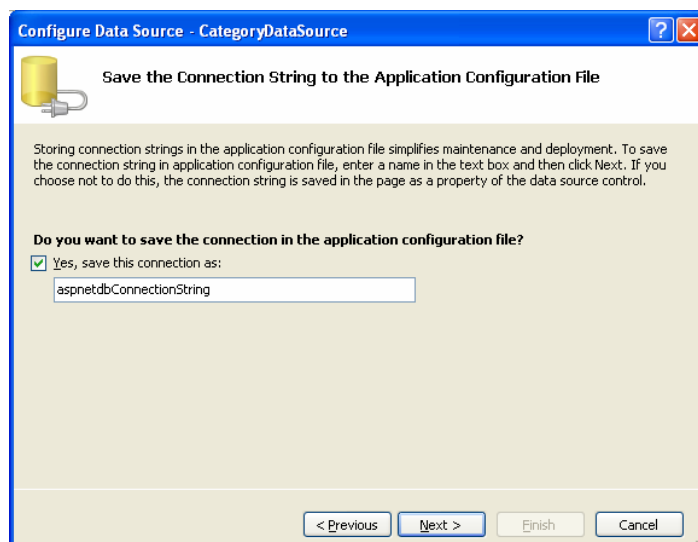
5. Állítsuk be a *DataList* tulajdonságait a következőkre:
 - a. Az adatforrás legyen egy új adatforrás, ami adatbázisból kapja az adatokat és hívjuk *CategoryDataSource*-nak.



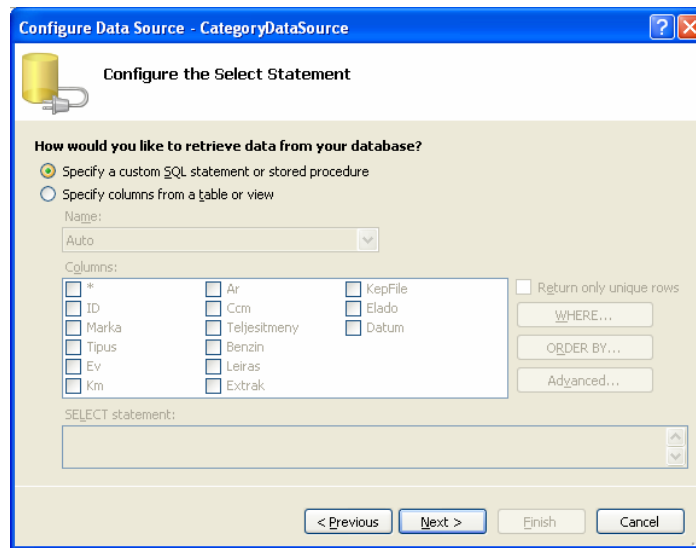
- b. Válasszuk ki melyik adatbázishoz szeretnénk csatlakozni. (*Gépnév.aspnetdb.dbo*)



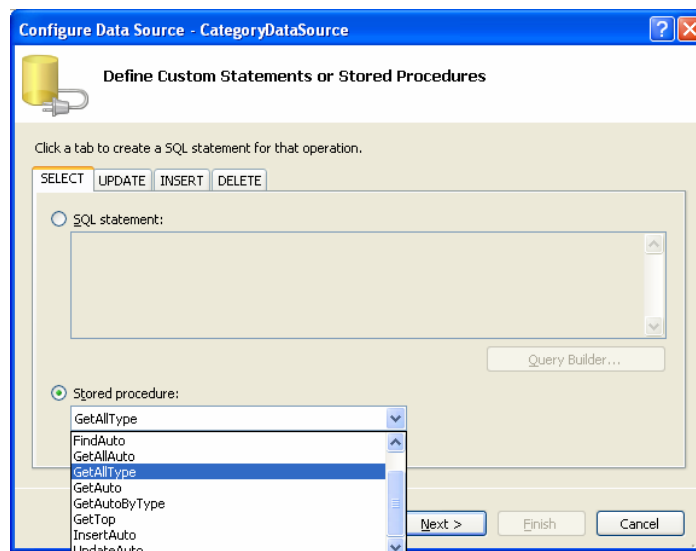
- c. Mentjük el a kapcsolódási sztringet a *web.config*-ba a felajánlott néven.



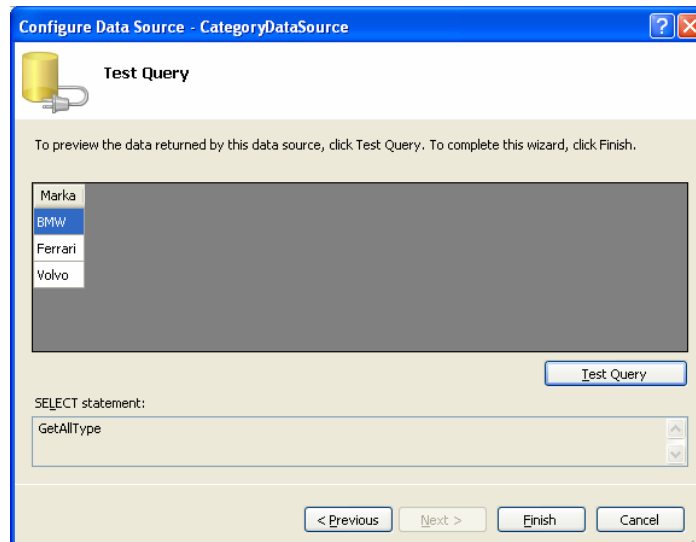
- d. Válasszuk ki, hogy saját tárolt eljárásokat szeretnénk használni.



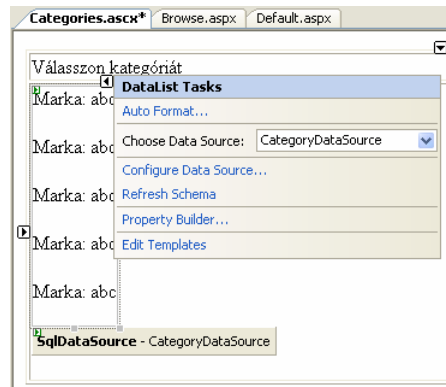
- e. Válasszuk ki, hogy a Select művelet a *GetAllType* nevű tárolt eljárás legyen.



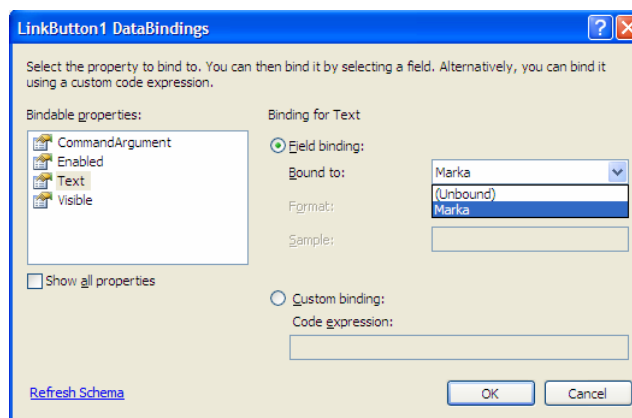
- f. Teszteljük a Select lekérdezést.



- g. Kattintsunk a befejezésre, majd válasszuk ki az *Edit Templates* parancsot GridView-hoz megjelenő intelligens címkéről.

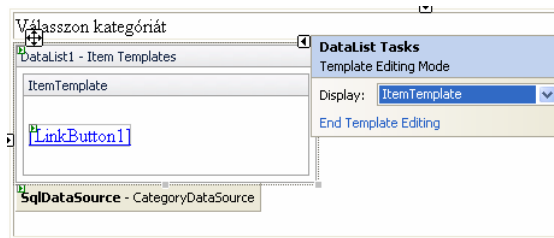


- i. Az *Item Template*-ből töröljük ki mindent.
- ii. Húzzunk be egy *LinkButton* vezérlőelemet a Toolboxról.
- iii. Majd az előbukkanó menüből válasszuk ki az *Edit DataBindings* parancsot.
- iv. A *Text* tulajdonságot kössük a *Marka* mezőhöz.

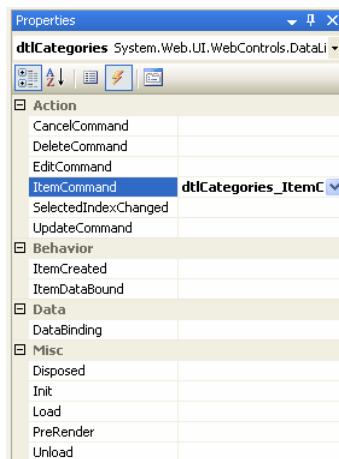


- v. Majd ugyanígy kössük a *CommandArgument* tulajdonságot is a *Marka* mezőhöz.

- vi. Végül válasszuk ki az egész DataListet, és az intelligens címkén válasszuk ki az End Template Editing menüpontot.



- h. Ez után már azt szeretnénk elérni, hogy ha egy kategóriára rákattint a felhasználó, akkor annak a kategóriának az autói jelenjenek meg az oldalon. A megjelenítést egy külön modul végzi majd (*CarList.ascx*). Most csak annyi dolgunk van, hogy implementáljuk a *DataList*-ünk *ItemCommand*-ját úgy, hogy a kiválasztott elem kerüljön be a *Session*-be. Ehhez válasszuk ki *DataList* tulajdonságait, és a *Properties* ablak tetején kattintsunk a „villámra” (Events), itt megtalálhatjuk az *ItemCommand* eseményt, amit implementálni szeretnénk. Kattintsunk rá duplán, hogy elkészítse a függvény keretét.



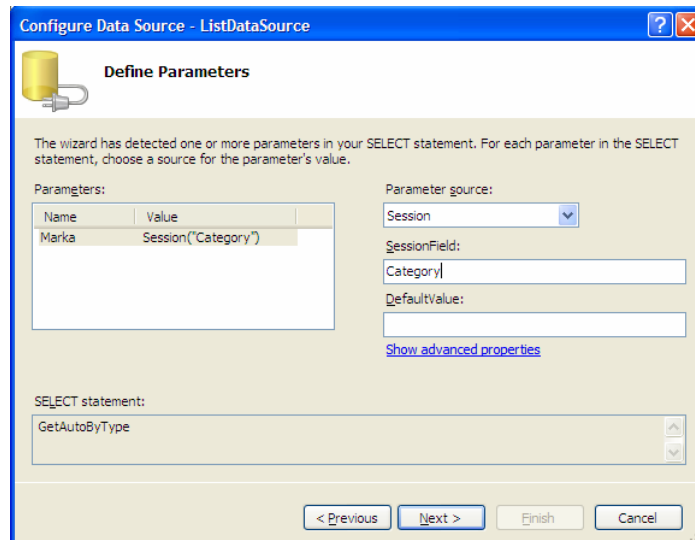
6. Implementáljuk a függvényt az alábbi módon:

```
protected void dtlCategories_ItemCommand(object source, DataListCommandEventArgs e)
{
    Session["Category"] = e.CommandArgument;
    Response.Redirect(Request.Url.ToString());
}
```

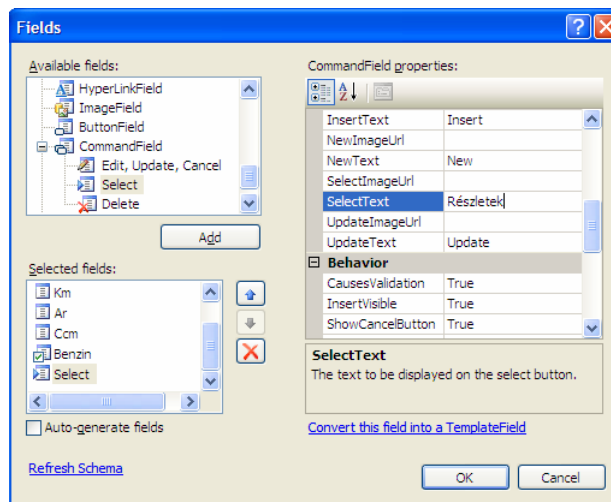
7. Végül nyissuk meg a *Browse.aspx*-et, és a Solution Explorerből húzzuk rá a *Categories.ascx*-et az oldalon található táblázat bal oldali cellájába.

3.1.2. Kiválasztott kategória autóinak megjelenítése

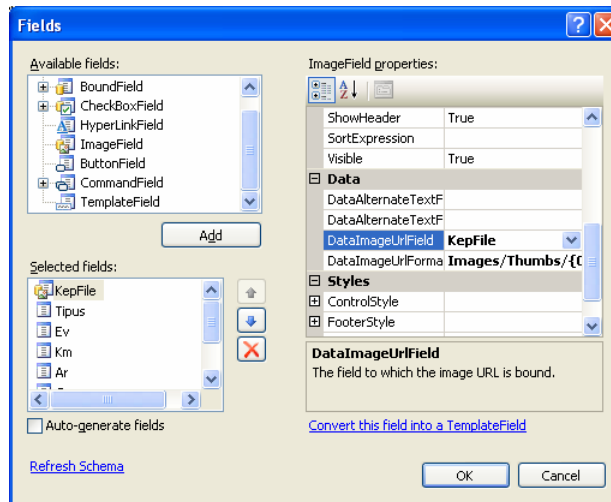
1. Adjunk a weboldalhoz egy új Web User Controlt *CarList.ascx* névvel.
2. Húzzunk a modulra egy GridView vezérlőelemet *grdList* névvel, majd állítsuk be a tulajdonságait az alábbiakra: Az adatkötés pontos menetére itt nem térek ki, hiszen azt a 3.1.1–es fejezet 5. lépésében már egyszer lépésről lépésre végignéztük.
 - a. Adjunk hozzá egy új adatforrást *ListDataSource* névvel. A lekéréshez használjuk a *GetAutoByType* nevű előre elkészített tárolt eljárást, majd ez után állítsuk be, hogy a *Session*-ből nyerve ki a *Category* kulccsal ellátott értéket, és ez legyen a tárolt eljárás bemenő paramétere.



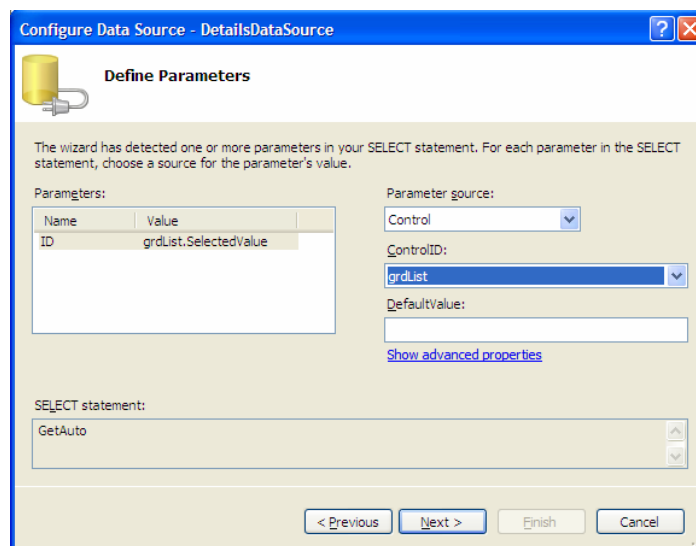
- b. Szerkesszük meg a GridView oszlopait a smart tag-en felbukkanó *Edit Columns* menüpont segítségével.
- i. Töröljük az *ID*, *Marka*, *Teljesitmeny*, *Leiras*, *Extrak*, *Kepfile*, *Elado*, *Datum* oszlopokat. Ezt úgy tehetjük meg, hogy a *Selected Fields* listából eltávolítjuk őket.



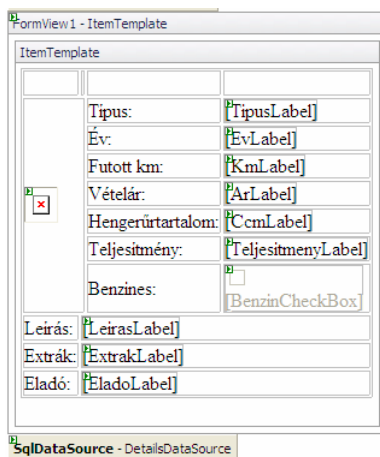
- ii. Adjunk hozzá egy *Command Field* → *Selectet*, melynek a *Selected Text* tulajdonsága legyen *Részletek*.
- iii. Adjunk hozzá egy *ImageField* típusú oszlopot is, melynek szintén állítsuk be a tulajdonságait:
 1. *DataImageUrlField*: *KepFile*
 2. *DataImageUrlFormat*: *Images/Thumbs/{0}*



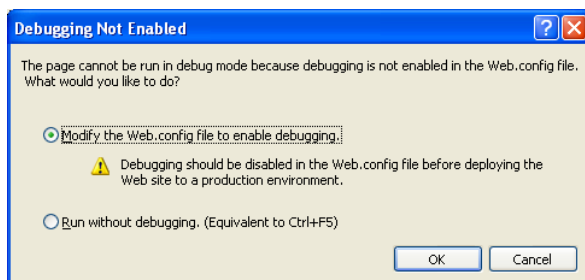
- c. A fel és le nyilak segítségével rendezzük át az oszlopokat úgy, hogy a kép legyen az első oszlop
- d. Ez után a GridView smart tag-jén engedélyezzük a lapozást (*Enable Paging*) és a rendezést (*Enable Sorting*).
3. Adjunk az oldalhoz egy FormView vezérlőt *frmDetails* névvel, majd állítsuk be a tulajdonságait az alábbiakra
 - a. Állítsunk be neki egy új adatforrást, *DetailsDataSource* névvel, ami a *GetAuto* nevű tárolt eljárással választja ki a megfelelő elemet. A tárolt eljárás bemeneti paramétere pedig a *GridView*-ban kiválasztott elem legyen.



- b. Ezt követően szerkesszük meg a FormView elrendezését, a smart tagen található Edit Template segítségével.
 - i. Szűrjünk be egy 11x4-es táblázatot az ItemTemplate-be. Ne feledkezzünk el a Cell Properties alatt kitorolni a 100px-es szélességet.
 - ii. Vonjuk össze a táblázatcellákat és mozgassuk át a mezőket úgy, hogy az alábbi elrendezést kapjuk. Cellákat úgy tudunk összevonni, hogy kijelöljük az összevonandó cellákat, majd jobb gomb → *Merge cells*. Az egyes kontrollokat drag&drop módszerrel tudjuk átmozgatni. A fel nem használt adatokat egyszerűen töröljük.

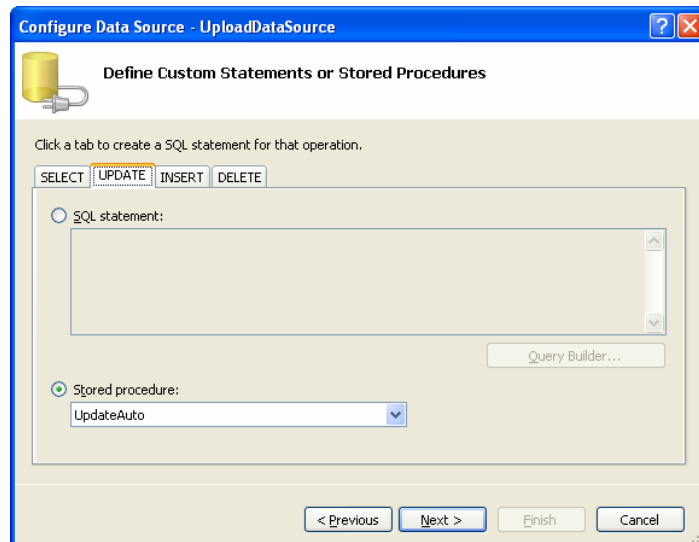


- iii. Szúrjunk be egy Image-et, melynek állítsuk be az adatkötési tulajdonságait a smart tag → Edit Databinding segítségével a következőre: *Bound to: Kepfile, Format: Images/{0}*
- iv. Az első sor celláira állítsuk be a következő szélességeket: 10%, 30%, 10%, 50%
- v. Fejezzük be a szerkesztést.
- c. A FormView tulajdonságai között keressük meg a *Header Text* tulajdonságot, és állítsuk *Részletes információ-ra*.
4. Ezzel el is készült a részletes megjelenítő. Már csak rá kell tenni a *Browse.aspx*-re. Ehhez nyissuk meg a *Browse.aspx*-et és a Solution Explorerből a *CarList.ascx*-et húzzuk rá az oldalon található táblázat jobb oldali cellájába.
5. Indítsuk el az alkalmazást debug módban a *Debug menü* → *Start Debugging* menüpont segítségével, vagy az *F5* billentyű lenyomásával. Az első indításnál a következő képernyővel fogunk találkozni. Ez azt kérdezi, hogy módosíthatja-e a *web.config*-ot úgy, hogy engedélyezze a debug módot. Kattintsunk az OK-ra.

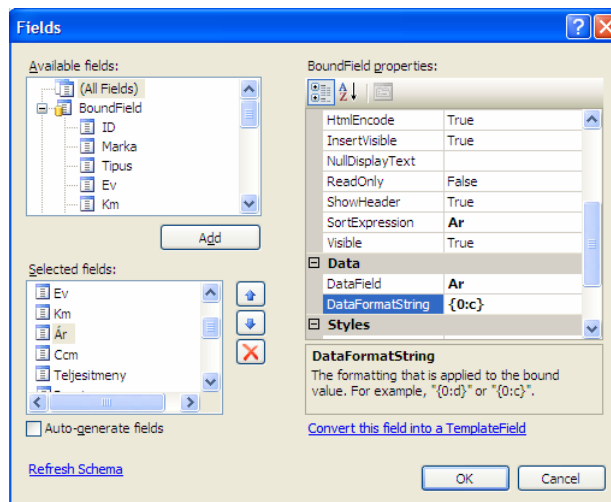


3.2. Új hirdetés feltöltése

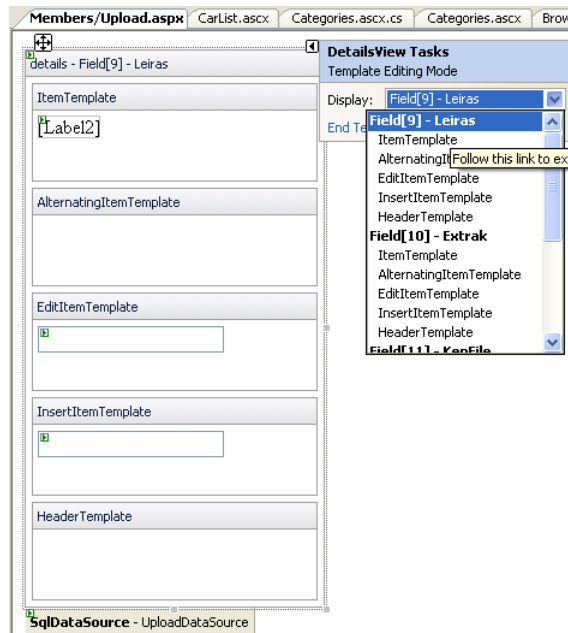
1. Hozzunk létre egy könyvtárat *Members* névvel.
2. Adjunk a *Members* könyvtárhoz új oldalt *Upload.aspx* névvel.
3. Adjunk hozzá egy *DetailsView*-t *details* névvel.
 - a. Állítsuk be az új adatforrást, melyek neve *UploadDataSource*:
 - i. SELECT: *GetAuto* nevű tárolt eljárás, és a *QueryString*-ből kapja a *CarId* kulccsal meghatározott azonosítót bemenő paraméterként.
 - ii. UPDATE: *UpdateAuto* nevű tárolt eljárás
 - iii. INSERT: *InsertAuto*
 - iv. DELETE: *DeleteAuto*



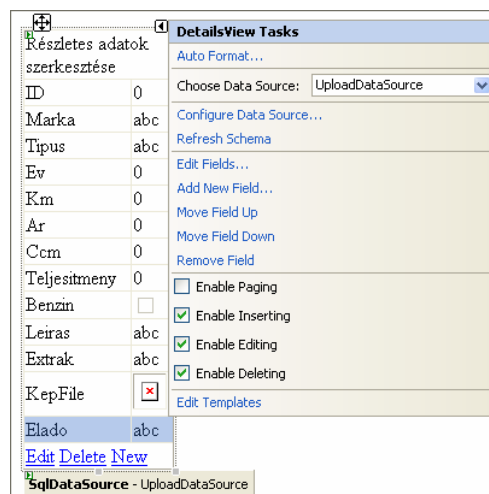
- b. Állítsuk be a DetailsView tulajdonságai között a *HeaderText*-et *Részletes adatok szerkesztésére*.
- c. A smart tag-en engedélyezzük a beszúrást, szerkesztést és törlést
- d. Ez után szerkesszük meg a mezőket a következőképpen:
 - i. **Árnál** állítsuk be *DataFormatString = {0:c}*



- ii. **Eladó:** *InsertVisibe = false, ReadOnly = True*. Ez azt jelenti, hogy beszúrásnál és szerkesztésnél az eladó nem lesz látható.
 - iii. Töröljük a **Dátum** mezőt.
 - iv. A **KepFile, Leiras** és **Extrak** mezőt alakítsuk *template*-té a fenti ábrán látható *Convert this field into a TemplateField* gomb segítségével.
- e. Ez után szerkesszük meg a DetailsView Template-jeit.



- i. A **Leírás** mező *EditItemTemplate* és *InsertItemTemplate*-jében a szövegdobozt tegyük többsorossá, amit a tulajdonságok között szereplő *TextMode = Multiline*-ra való állításával tehetünk meg.
- ii. Az **Extrák** mező *EditItemTemplate* és *InsertItemTemplate*-jében a szövegdobozt szintén tegyük többsorossá.
- iii. A **KepFile** *ItemTemplate*-jébe a címke helyett egy képet tegyünk, melynek állítsuk be az adatkötési tulajdonságait a következőkre:
 1. *ImageUrl = KepFile*
 2. *Format = ~/Images/Thumbs/{0}*
- iv. A **Kepfile** *EditItemTemplate* és *InsertItemTemplate*-jébe a szövegdoboz helyett tegyünk be egy *FileUpload* nevű vezérlőelemet, melynek neve legyen *UploadKepfile*
- f. Fejezzük be a szerkesztését (*End Template Editing*).
- g. Engedélyezzük a *DetailsView* smart tagjén a beszúrást, szerkesztést és a módosítást.



- h. HTML nézetben a két *FileUpload* kontrollt egészítsük ki a következővel (E nélkül nem fogja elmenteni a kép nevét az adatbázisba)

```
<asp:FileUpload ID="UploadKepfile" FileName="<%# Bind("KepFile") %>"
runat="server" />
```

4. Az oldalon nyomjunk egy F7-et amivel át tudunk váltani a kódnézetre, majd Készítsük el az oldal *Page_Load*-ját úgy, hogy ha van az URL-ben CarId, akkor jelenítsük meg annak az autónak az adatait, ha nincs, akkor beszúrás módban jelenjen meg a vezérlőelem.

```
protected void Page_Load(object sender, EventArgs e)
{
    if (Request.QueryString["CarID"] != null)
        details.DefaultMode = DetailsViewMode.ReadOnly;
    else
        details.DefaultMode = DetailsViewMode.Insert;
}
```

5. A beszúrásnál és a módosításnál nekünk kell beállítani az eladót, aki nem más, mint az aktuális felhasználó. Ehhez implementáljuk a *details_ItemInserting* és a *details_ItemUpdating* eseménykezelőket.

```
using System.Drawing.Imaging;
using System.IO;
private void CreateThumbnail(String srcpath, String destpath)
{
    using( System.Drawing.Image img = System.Drawing.Image.FromFile(srcpath) )
    using( System.Drawing.Image imgthumb = img.GetThumbnailImage(100, 75, null,
System.IntPtr.Zero ) )
    {
        imgthumb.Save(destpath, ImageFormat.Jpeg);
    }
}

private void SaveImage()
{
    FileUpload f = (FileUpload)details.FindControl("UploadKepfile");
    if (f.HasFile)
    {
        String path = Server.MapPath("~/Images/" + Path.GetFileName(f.FileName));
        f.SaveAs(path);
        CreateThumbnail(path, Server.MapPath("~/Images/Thumbs/" + f.FileName));
    }
}

protected void details_ItemUpdating(object sender, DetailsViewUpdateEventArgs e)
{
    e.NewValues["Elado"] = "Teszt felhasználó";
    SaveImage();
}

protected void details_ItemInserting(object sender, DetailsViewInsertEventArgs e)
{
    e.Values["Elado"] = "Teszt felhasználó";
    SaveImage();
}
```

3.3. Keresés

1. Hozzunk létre egy új Web User Controlt *SearchModule.ascx* névvel.
 - a. Szúrjunk be egy 5x2-es táblázatot melynek oszlopai rendre 27% és 73% szélesek, a keret vastagsága legyen 1px.
 - b. A felső sor celláit vonjuk össze, és állítsuk be neki a *ModuleHeader* CSS osztályt.
 - c. A következő ábra szerint készítsük el a modult.

The image shows a search form with the following elements:

- A text input field labeled "Kereső".
- A label "Min ár:" followed by a text input field and the unit "eFt".
- A label "Max ár:" followed by a text input field and the unit "eFt".
- A label "Évjárat:" followed by a text input field.
- A "Keresés" button.

- d. A szövegdobozok *ID*-jai nevei legyenek rendre: *txtMinPrice*, *txtMaxPrice*, *txtYear*
- e. A linkbutton *ID*-ja legyen *txtSearch*, a *Text*-je pedig *Keresés*
- f. A gomb eseménykezelőjében mentjük el a *Session*-be a megadott adatokat, a következő módon:

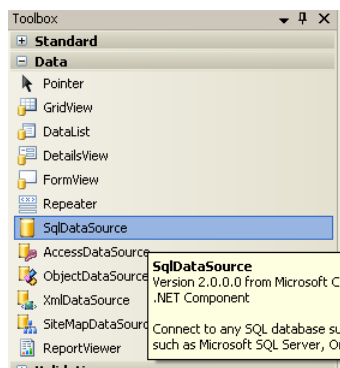
```
protected void lbtSearch_Click(object sender, EventArgs e)
{
    if (txtMinPrice.Text.Length > 0)
        Session["MinPrice"] = Int32.Parse(txtMinPrice.Text) * 1000;

    if (txtMaxPrice.Text.Length > 0)
        Session["MaxPrice"] = Int32.Parse(txtMaxPrice.Text) * 1000;

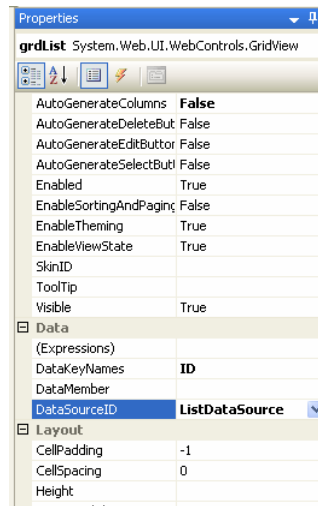
    if (txtYear.Text.Length > 0)
        Session["Year"] = Int32.Parse(txtYear.Text);

    Response.Redirect(Request.Url.ToString());
}
```

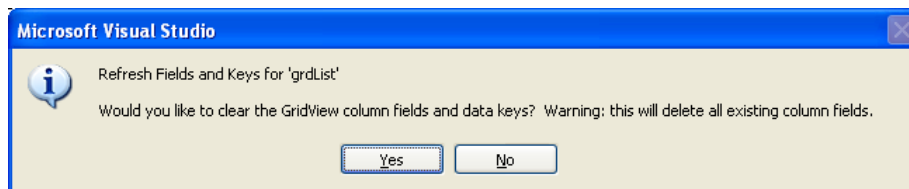
2. Hozzunk létre egy új oldalt *Search.aspx* néven.
 - a. Szúrjunk be egy kétoszlopos táblázatot, melyre állítsunk egy 5 pixeles *cellpadding*-ot, és állítsuk be, hogy a függőleges igazítás felfelé legyen.
 - b. A bal oldali cellába húzzuk bele a *SearchModule.ascx*-et.
3. Módosítsuk a *CarList.ascx*-et a következő módon:
 - a. Adjunk az oldalhoz egy új SQL adatforrást *FindDataSource* névvel



- i. A *FindAuto* tárolt eljárást használja
- ii. *Session*-ből vegye a *MinPrice*-ot, 0 alapértelmezett értékkel
- iii. *Session*-ből vegye a *MaxPrice*-ot, 30000000 alapértelmezett értékkel
- iv. *Session*-ből vegye a *Year*-t 0 alapértelmezett értékkel.
- b. Töröljük a *grdList DataSourceID*-jat



c. Ez után rákérdez, hogy frissítse-e a grdList mezőit. Válaszoljunk NO-val.



d. Ezt követően a Page_Load-ban vizsgáljuk meg, hogy mi van a Sessionben és ez alapján válasszuk ki a megfelelő adatforrást.

```
protected void Page_Load(object sender, EventArgs e)
{
    if ((Session["MinPrice"] != null) || (Session["MaxPrice"] != null) || (Session["Year"] != null))
        grdList.DataSource = FindDataSource;
    else
        grdList.DataSource = ListDataSrouce;
    grdList.DataBind();
}
```

4. Ez után módosítsuk a Browse.aspx Page_Load-ját, hogy minden betöltődéskor töröljük a Session-t:

```
protected void Page_Load(object sender, EventArgs e)
{
    if (Session["MinPrice"] != null)
        Session.Remove("MinPrice");

    if (Session["MaxPrice"] != null)
        Session.Remove("MaxPrice");

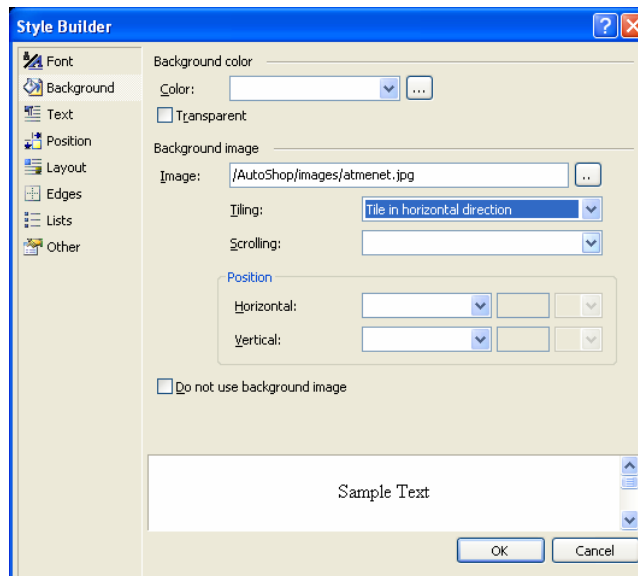
    if (Session["Year"] != null)
        Session.Remove("Year");
}
```

5. Végül módosítsuk a Search.aspx Page_Load-ját, hogy a Session-ből töröljük a Category-t:

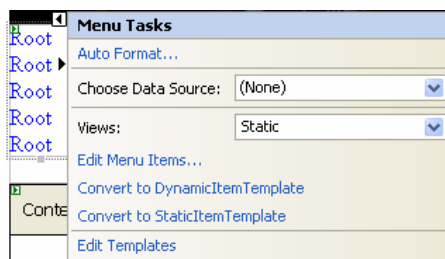
```
protected void Page_Load(object sender, EventArgs e)
{
    if (Session["Category"] != null)
        Session.Remove("Category");
}
```

4. MasterPage

1. Adjunk a projecthez egy MasterPage-et. File menü → New File és itt válasszuk ki a *Master Page*-et. A neve maradjon *MasterPage.master*.
 - a. HTML nézetben töröljük a `<div></div>` elemeket
 - b. A `ContentPlaceHolder` elé szúrjuk be egy 3x2-es táblázatot
 - i. Width: *100%*
 - ii. Cellpadding: *0px*
 - iii. Cellspacing: *0px*
 - iv. Az első és utolsó sorban vonjuk össze a cellákat.
 - c. A legelső cellába húzzuk át a `ContentPlaceHolder`-t.
 - d. A felső sorra állítsunk be egy fekete hátteret, majd helyezzünk bele egy `image` típusú vezérlőelemet a következő tulajdonságokkal:
 - i. ID: *imgBanner*
 - ii. ImageUrl: *~/Images/topBanner1024.jpg*
 - e. Középső sorra állítsuk be a következőket:
 - i. Height: *48px*
 - ii. Legyen a bal oldali cella: *70%* széles, a jobb oldali *30%*
 - iii. BackgroundImage: */Web/images/atmenet.jpg* (Itt azért nem használhatjuk a *~/Images*-t mert a `style` tag-en belül nem lehet a *~*-t kiértékelni.
 - iv. Tiling: *Tile in horizontal direction*



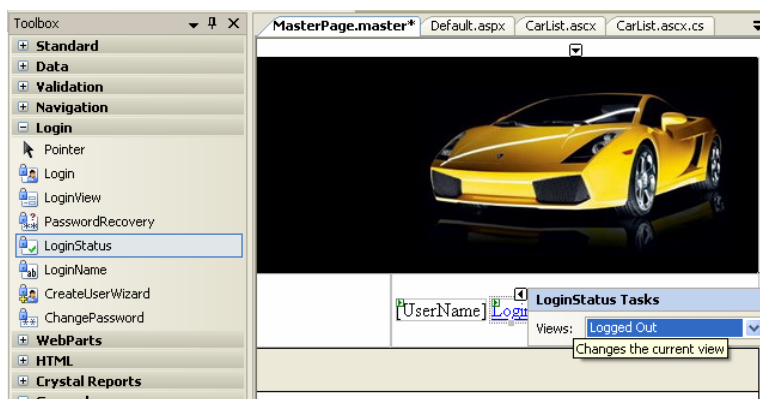
- f. Adjunk egy `Menu` kontrollt a Toolboxról a középső sor bal oldali cellájához **topMenu** névvel. Majd állítsuk be a tulajdonságait:
 - i. Orientation: *Horizontal*
 - ii. StaticDisplayLevels: *2*



- iii. Állítsuk be az adatforrást egy új SiteMap DataSource-ra, melynek neve legyen *MenuSiteMap*.
- g. Adjunk az alkalmazásunkhoz egy új SiteMap típusú elemet web.sitemap fájl névvel.
- h. Egészítsük ki a web.sitemap-et a következő módon:

```
<?xml version="1.0" encoding="utf-8" ?>
<siteMap xmlns="http://schemas.microsoft.com/AspNet/SiteMap-File-1.0" >
  <siteMapNode url="" title="" description="">
    <siteMapNode url="-/Default.aspx" title="Kezőoldal" description="" />
    <siteMapNode url="-/Browse.aspx" title="Böngészés" description="" />
    <siteMapNode url="-/Search.aspx" title="Keresés" description="" />
    <siteMapNode url="-/Members/Upload.aspx" title="Autó feltöltés" description="" />
    <siteMapNode url="-/Members/PersonalData.aspx" title="Személyes adatok" description="" />
  />
</siteMapNode>
</siteMap>
```

- i. A jobb oldali cellában egy Belépés illetve Kilépés gombot szeretnénk megjeleníteni, attól függően, hogy a felhasználó bejelentkezett-e. Ehhez a LoginStatus vezérlőelemet kell használnunk. Ugyanebben a cellában jelenítsük meg a felhasználó nevét, ha már belépett. Ezt a LoginName vezérlőelemmel tehetjük meg. Tegyük fel ezt a két vezérlőelemet a Toolboxról a mesterlapra.



- 2. Ezek után már csak azt kell elérnünk, hogy az eddig elkészített oldalak (*Default.aspx*, *Browse.aspx*, *Search.aspx*, *Upload.aspx*) használják az elkészített mester oldalt.
 - a. Nyissuk meg a *Browse.aspx*-et, HTML nézetben
 - b. Egészítsük ki a Page direktívát a következőre:

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="Browse.aspx.cs" Inherits="Browse"
MasterPageFile="~/MasterPage.master" %>
```

- c. Majd töröljük minden HTML elemet, ami a form tag-en kívül van, illetve magát a formot is. Erre azért van szükség, mert ezeket az elemeket a mester oldalon már definiáltuk, így ezeket az egyes oldalakon már nem kell, sőt nem is szabad megadni. Pontosán a következőket kell törölni:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
```

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head runat="server">
  <title>Untitled Page</title>
</head>
<body>
  <form id="form1" runat="server">
```

```
</form>
</body>
</html>
```

- d. Ezt követően helyezünk el egy Content tag-et az oldalunkra. (Ezen belül kell lennie annak a tartalomnak, ami majd bekerül a mester oldalon lévő ContentPlaceHolder-be. Ezek után így fog kinézni az oldalunk, mely már használja a mester oldalt.

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="Browse.aspx.cs" Inherits="Browse"
MasterPageFile="~/MasterPage.master" %>
```

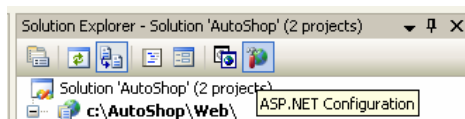
```
<%@ Register Src="Categories.ascx" TagName="Categories" TagPrefix="uc1" %>
```

```
<%@ Register Src="CarList.ascx" TagName="CarList" TagPrefix="uc2" %>
```

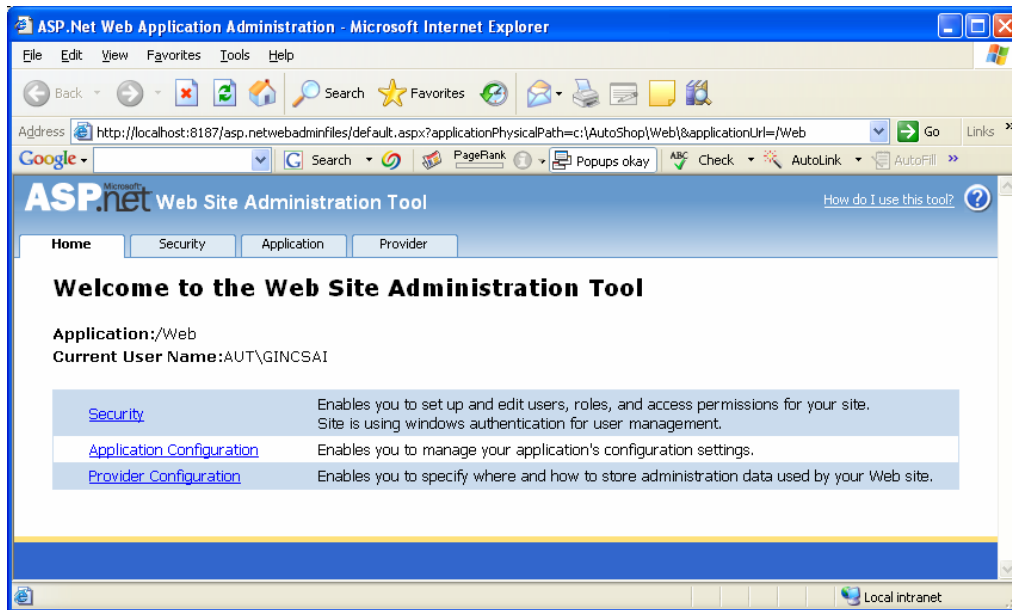
```
<asp:Content id="Content1" runat="server" ContentPlaceHolderID="ContentPlaceHolder1">
  <table cellpadding="5" style="width: 100%">
    <tr>
      <td style="width: 20% valign="top">
        <uc1:Categories ID="Categories1" runat="server" />
      </td>
      <td style="width: 80% valign="top">
        <uc2:CarList ID="CarList1" runat="server" />
      </td>
    </tr>
  </table>
</asp:Content>
```

- e. Ismételjük meg a fenti lépéseket (a-d) a fennmaradó oldalakra. (*Default.aspx*, *Search.aspx*, *Upload.aspx*)
3. Indítsuk el az alkalmazásunkat.

- a. Sajnos a belépés gombunk nem működik rendesen, és a felhasználó név sem stimmel. Ennek az az oka, hogy az alkalmazásunk jelenleg Windows hitelesítéssel fut és nem úrlap alapúval. Ezt a WebSite Admin Tool segítségével egyszerűen át tudjuk állítani, amit az alábbi ábrán kijelölt ikonnal tudunk elindítani, vagy a *WebSite menü* → *ASP.NET Configuration* menüpont segítségével.



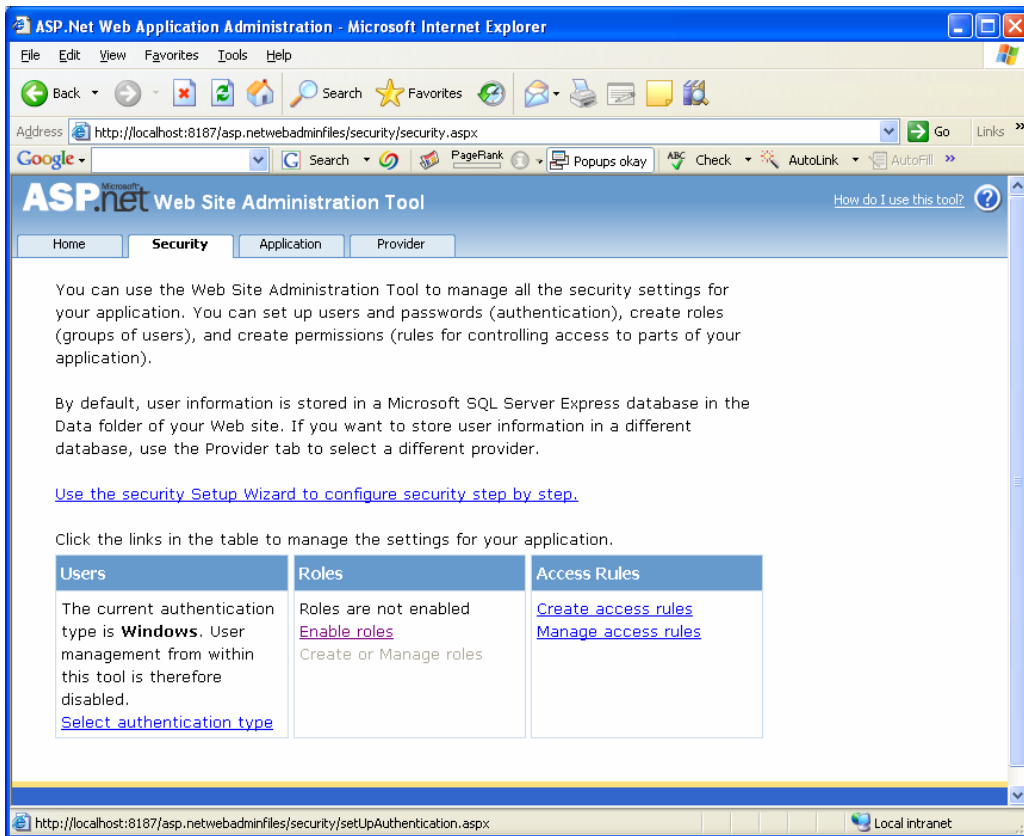
- b. Elindítás után az alábbi a képernyővel találkozunk. Itt kattintsunk a Security fülre. Ha Microsoft SQL Server 2005 Express Edition-t használunk, akkor a c. és d. pontot hagyjuk ki!



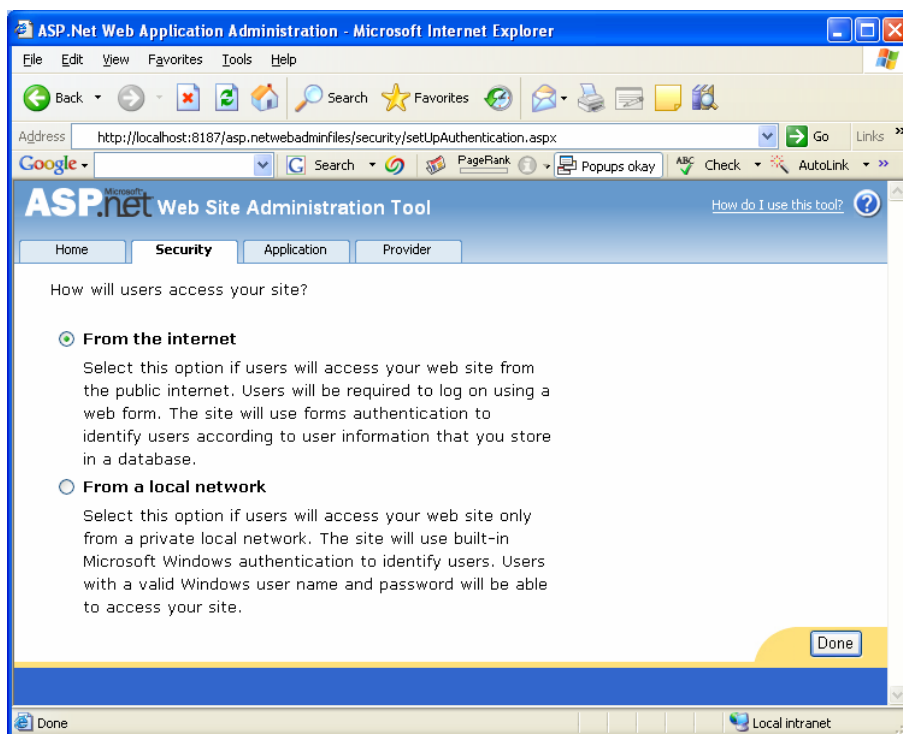
- c. Ha SQL Server 2005-öt használunk, és nem Express Editiont akkor a Security fül dob egy hibát, hogy nem tud kapcsolódni az adatbázisszerverhez. Ennek az oka, hogy a WSAT (Web Site Admin Tool) megpróbál csatlakozni az adatbázishoz egy a machine.config fájlban bedrótázott kapcsolódási sztringgel, melynek neve LocalSqlServer. Alapértelmezés szerint ez egy SQL Express adatbázishoz szeretne csatlakozni. Ahhoz, hogy megoldjuk ezt a problémát a web.configban felül kell definiálnunk ezt a kapcsolódási sztringet a következő módon:

```
<connectionStrings>
  <remove name="LocalSqlServer" />
  <add name="LocalSqlServer" connectionString="Data Source=.;Initial
Catalog=aspnetdb;Integrated Security=True" providerName="System.Data.SqlClient" />
  <add name="aspnetdbConnectionString" connectionString="Data Source=.;Initial
Catalog=aspnetdb;Integrated Security=True" providerName="System.Data.SqlClient" />
</connectionStrings>
```

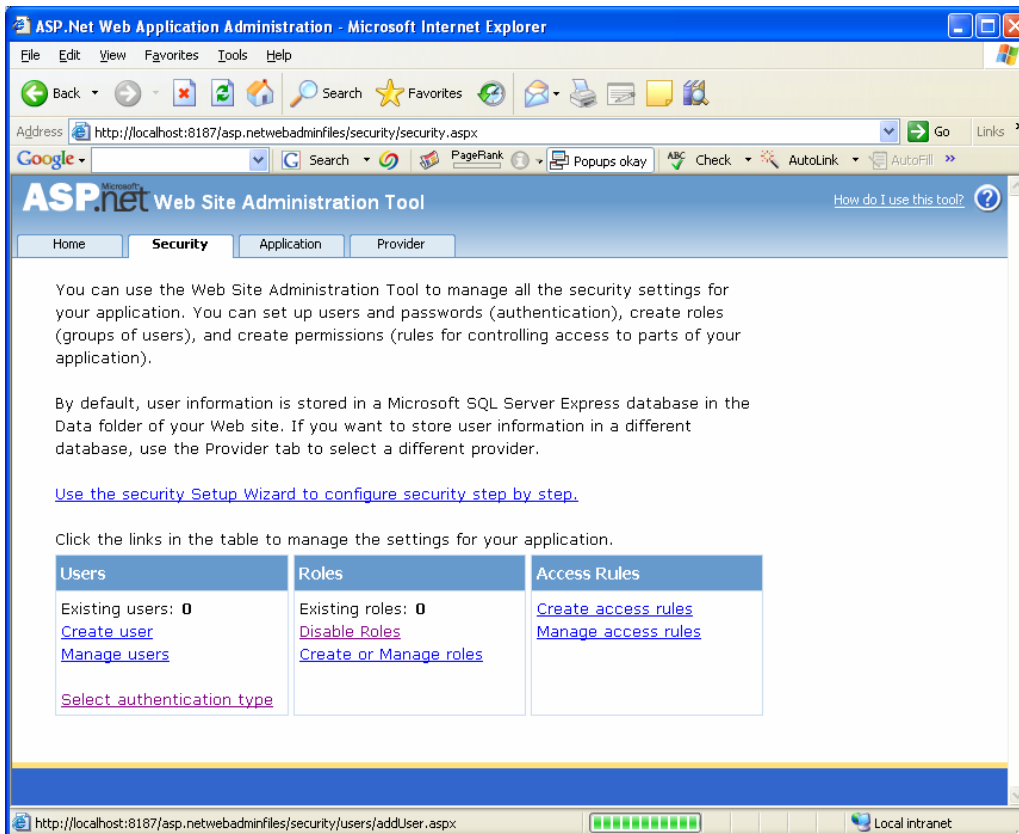
- d. Tehát először eltávolítjuk a bedrótázott LocalSqlServer kapcsolódási sztringet, majd megadjuk a helyes sztringet, ami már SQL Server 2005-höz kapcsolódik és nem SQL Expresshez.
- e. Ezek után bejön a Security fül.



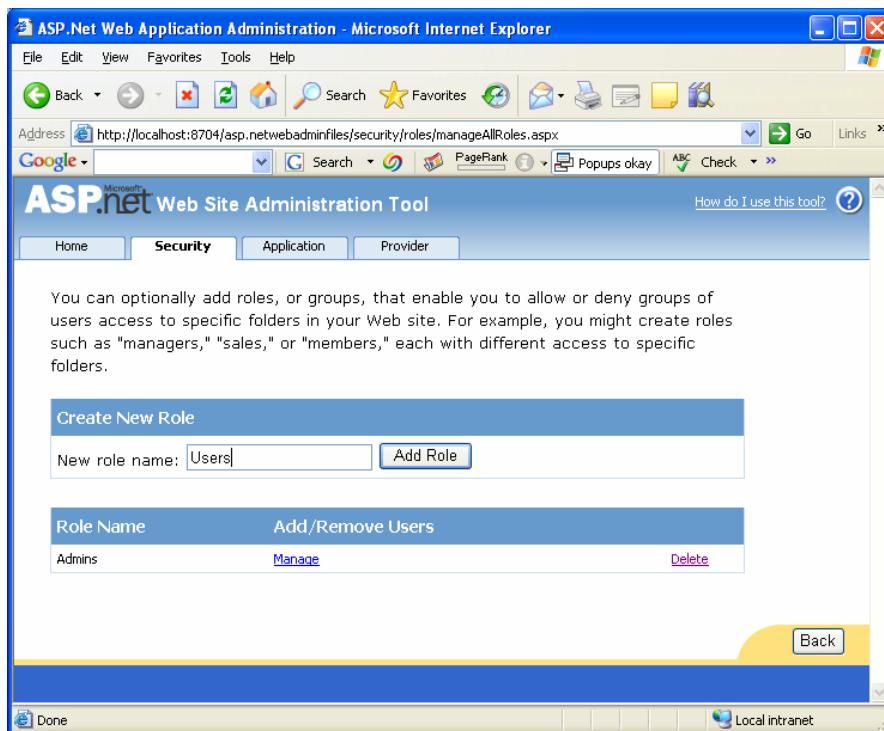
- f. Állítsuk át először az azonosítást Windows alapúról űrlap alapúra.
- i. Kattintsunk a *Select authentication type*-ra, majd válasszuk a *From the Internet* opciót.



- g. Ez után visszakerülünk az előző oldalra, ahol engedélyezzük a szerepköröket (Enable Roles menüpontra kattintva.)

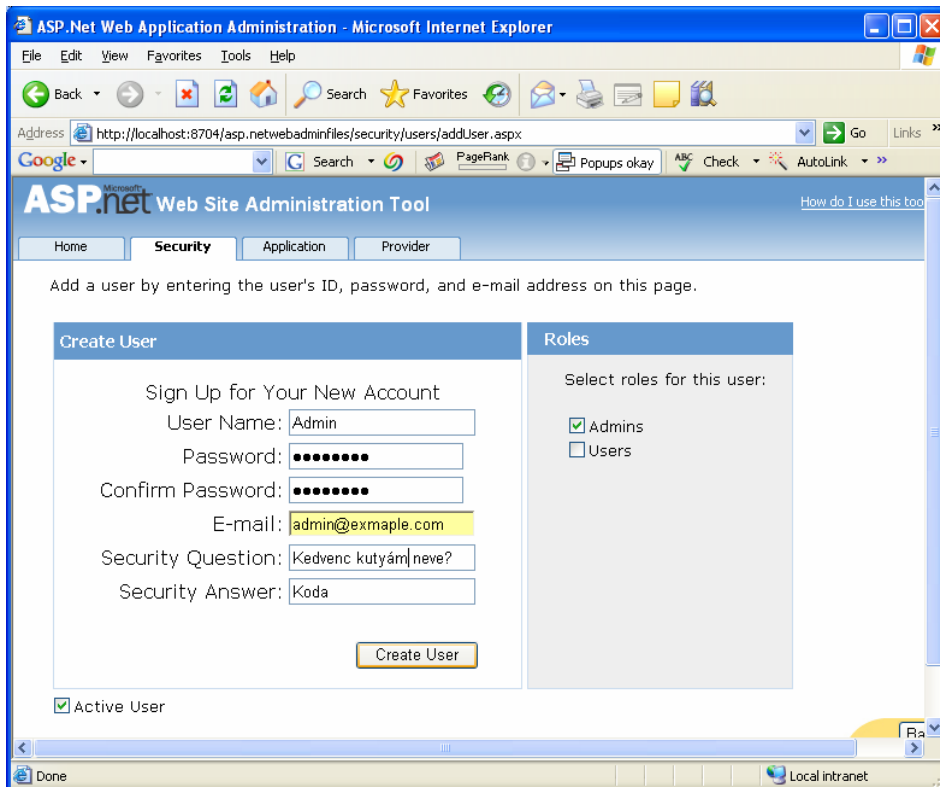


- h. Majd hozzunk létre egy *Admins* és egy *Users* szerepkört, a Create or Manage roles gomb segítségével.

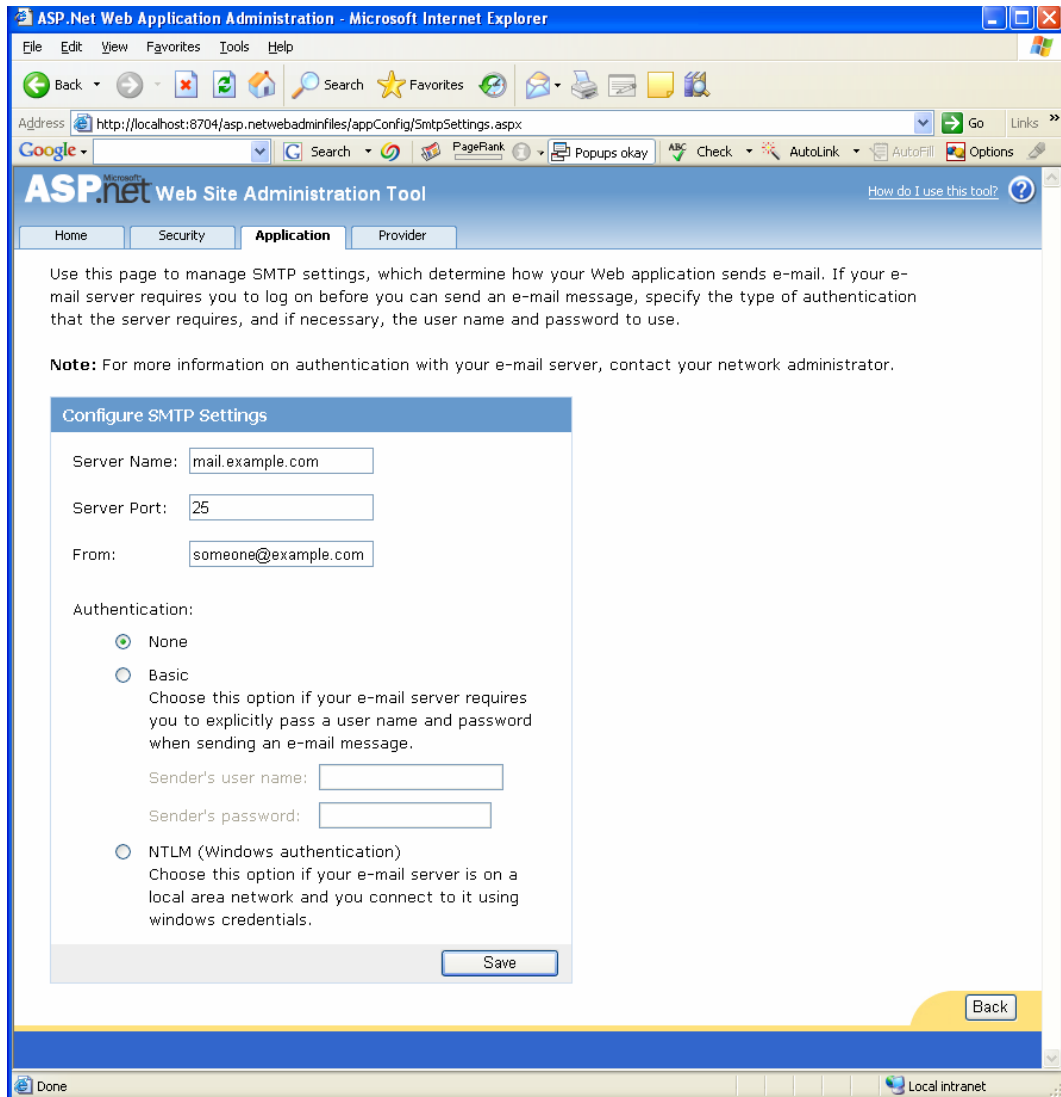


- i. Ezután menjünk vissza a Security fülre, és hozzunk létre két felhasználót (Admin és User névvel, és jelöljük ki nekik a megfelelő szerepkört) a Create User gomb segítségével.

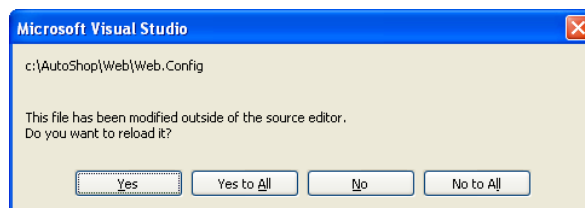
- i. Fontos, hogy a jelszónak tartalmaznia kell alapértelmezés szerint egy nem alfanumerikus karaktert is.



- j. Végül az Application fül alatt állítsuk be az SMTP szervert, hogy a jelszó emlékeztető ki tudja küldeni az új jelszót e-mailben. A példában az example.com domain szerepel, azonban ezt mindenkinek le kell cserélnie a saját SMTP szerverének az adataira.
 - i. Server Name: mail.example.com
 - ii. Port: 25
 - iii. From: someone@example.com



- k. Zárjuk be a WSAT ablakot, majd töltsük újra a web.config-ot. (Yes to All)

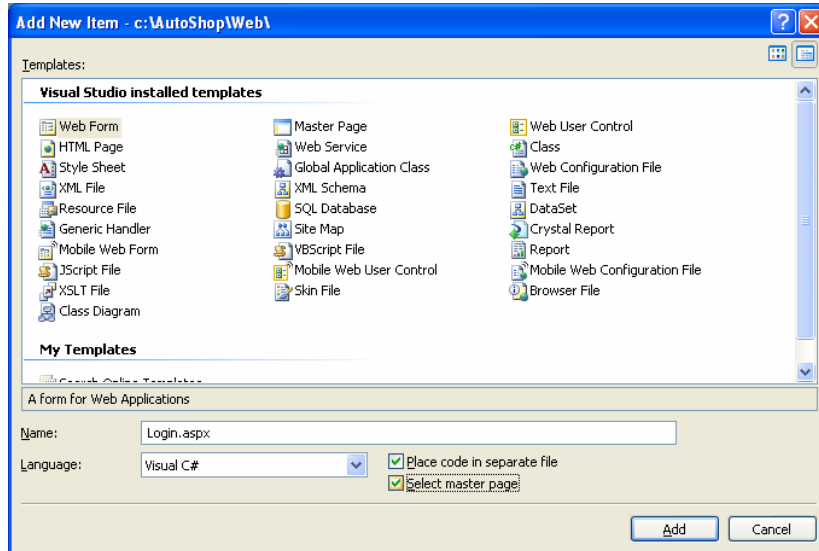


- l. Ezek után a web.config-ban nem fog működni az IntelliSense, amit úgy tudunk életre kelteni, hogy a configuration elemből kitöröljük az xmlns=""... részét.

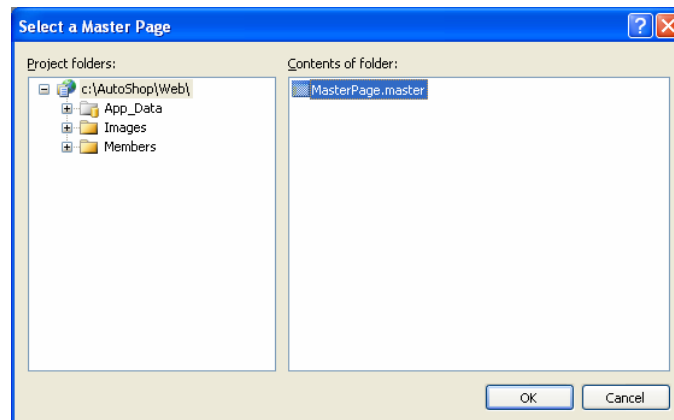
```
<configuration xmlns=""http://schemas.microsoft.com/.NetConfiguration/v2.0">
```

5. Login kontrollok

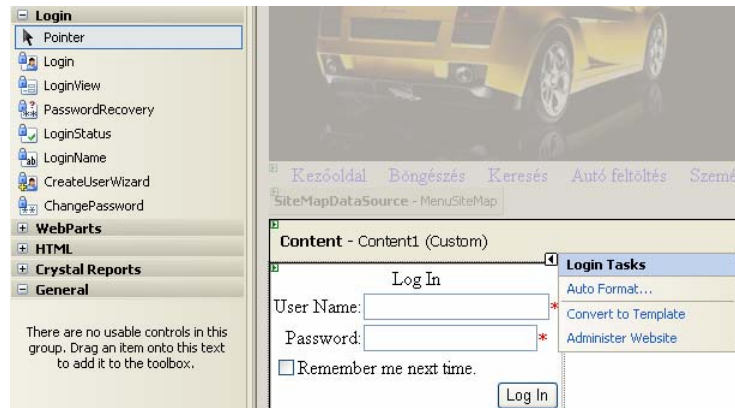
1. Hozzunk létre egy új oldalt *Login.aspx* névvel.
 - a. A létrehozásnál figyeljünk, hogy a *Select Master Page* opció be legyen pipálva.



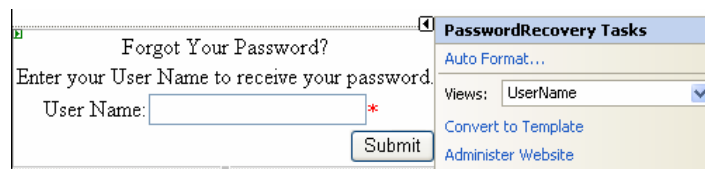
- b. Ezt követően válasszuk ki a kívánt mester oldalt.



2. Adjunk az oldalhoz egy Login kontrollt, és állítsuk be a következő tulajdonságait:
 - a. *VisibleWhenLoggedIn*: *False*
 - b. *CreateUserText*: *Regisztráció*
 - c. *CreateUserUrl*: *~/Registration.aspx*



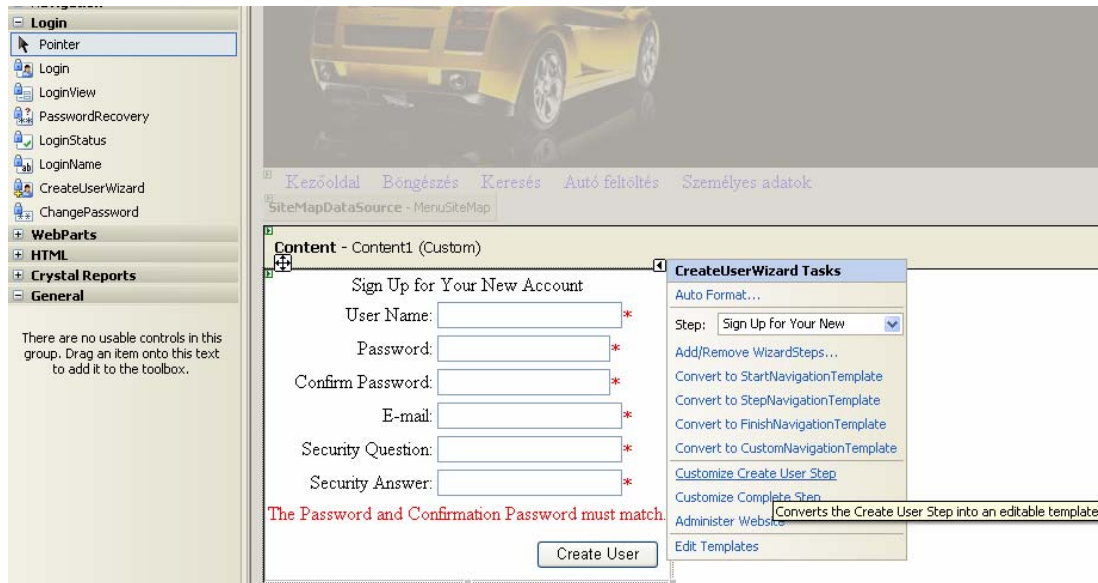
3. Adjunk az oldalhoz egy *PasswordRecovery* kontrollt is, hogy ha a felhasználó elfelejtette a jelszavát, akkor tudjunk neki emlékeztetőt, vagy újat küldeni. A levélküldéshez már az SMTP szervert bekonfiguráltuk a WSAT segítségével.



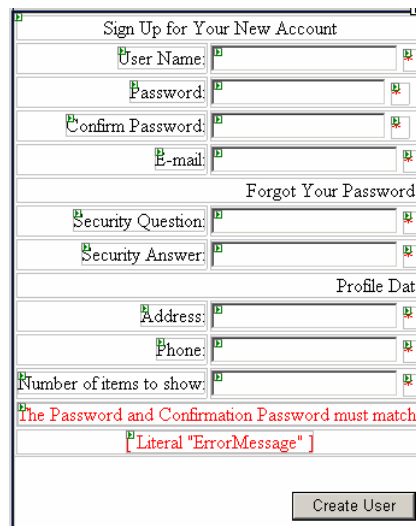
4. Ezen kívül még szükségünk lesz egy regisztrációs oldalra, ahol a kötelező adatok mellett be szeretnénk kérni egy pár személyes adatot is, mint például a cím, telefonszám és hogy a legújabb elemek közül mennyit jelenítsen meg a Top10 modul (ezt a későbbiek során fogjuk elkészíteni). Ezen adatok tárolására és kezelésére az ASP.NET 2.0 profil szolgáltatása lesz segítségünkre. Azonban, hogy ezt használni tudjuk a *web.config*-ban meg kell adni, hogy milyen extra információkat szeretnénk eltárolni a felhasználóról. Ezt a következőképpen tehetjük meg:

```
<profile enabled="true">
  <properties>
    <add name="Phone" type="string"/>
    <add name="Address" type="string"/>
    <add name="NumberOfItems" type="int"/>
  </properties>
</profile>
```

5. Ez után hozzuk létre a regisztrációs oldalt *Registration.aspx* névvel
 - a. Adjunk hozzá egy *CreateUserWizard*-ot, majd szabjuk testre a Create User lépést (*Customize Create User Step*).



- b. Egészítsük ki a regisztrációs modult a felhasználó személyes adatainak (cím, telefonszám, megjelenítendő elemek száma) a bekérésével. A szövegdobozok nevei legyenek rendre *txtAddress*, *txtPhone* és *txtNumberOfItems* Annak érdekében, hogy a felhasználónak mindenképpen ki kelljen töltenie ezeket a szövegdobozokat, használjunk *RequiredFieldValidator*-t, amit a Toolboxnak a *Validation* szekciójában találhatunk meg. Állítsuk be a validátorok *ErrorMessage* tulajdonságát *-ra, a *ControlToValidate* tulajdonságát pedig a megfelelő szövegdoboz azonosítójára. Azonban ahhoz, hogy a most hozzáadott validátorok a gombra kattintáskor valóban lefussanak, be kell állítanunk, hogy melyik validációs csoportba tartoznak ezek a validátorok. Erre azért van szükség, hogy egy oldalon több validációs csoportot is tudjunk definiálni, és ha rákattintunk egy gombra, akkor csak egy meghatározott csoport validátorai jelezzék az esetleges hibát. Tehát állítsuk be a *ValidationGroup* tulajdonságot a *CreateUserWizard* azonosítójára, jelen esetben: *CreateUserWizard1*-ra.



- c. Majd a tulajdonságok között állítsuk be a *ContinueDestinationUrl*-t a *Default.aspx*-re.
- d. Az így kiegészített regisztrációs modul azonban csak a kötelezően megadandó adatokat fogja automatikusan eltárolni, a profil adatok elmentéséről nekünk

kell gondoskodnunk. Ezt úgy tehetjük meg, hogy az alábbi módon implementáljuk a `CreateUserWizard CreatedUser` eseményét, ami a felhasználó létrehozása után fut le. (Érdemes megfigyelni, hogy a `web.config`-ban megadott profil adatokat típusosan felkínálja az IntelliSense.)

```
protected void CreateUserWizard1_CreatedUser(object sender, EventArgs e)
{
    ProfileCommon p = (ProfileCommon)ProfileCommon.Create( CreateUserWizard1.UserName, true);
    p.Address=((TextBox)CreateUserWizard1.CreateUserStep.ContentTemplateContainer.FindControl(
"txtAddress")).Text;

    p.Phone=((TextBox)CreateUserWizard1.CreateUserStep.ContentTemplateContainer.FindControl(
"txtPhone")).Text;

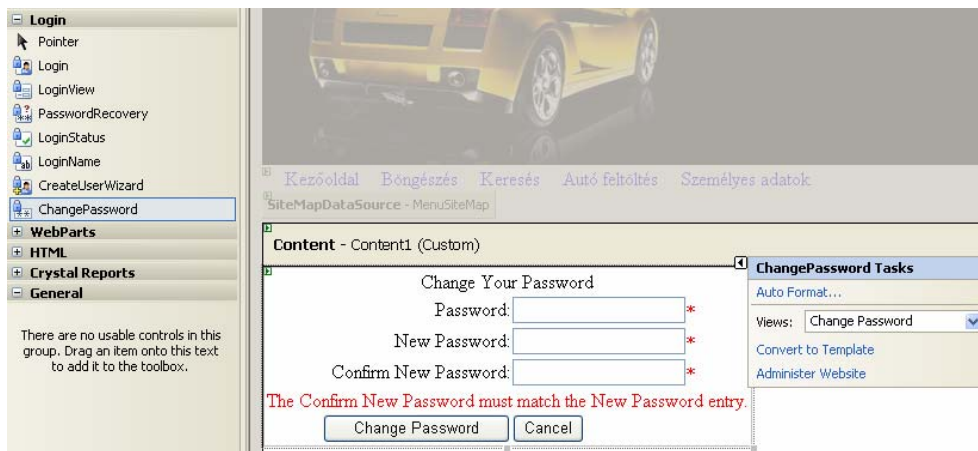
    p.NumberOfItems=Int32.Parse(((TextBox)CreateUserWizard1.CreateUserStep.
ContentTemplateContainer.FindControl( "txtNumberOfItems")).Text);

    p.Save();

    if (!Roles.RoleExists("Authenticated"))
        Roles.CreateRole("Authenticated");

    Roles.AddUserToRole(CreateUserWizard1.UserName, "Authenticated");
}
```

6. Hozzuk létre a *Members* könyvtár alatt a *PersonalData.aspx* oldalt, mely a profil adatok és a jelszó módosítására fog szolgálni.
 - a. Tegyük az oldalra egy *ChangePassword* kontrollt.



7. Hozzuk létre egy új modult *ProfileData.ascx* névvel.
 - a. Készítsük el az alábbi felhasználó felületet.
 - i. Táblázat szélessége legyen *60%*
 - ii. Felső sor CSS: *ModuleHeader*, többi: *ModuleBody*
 - iii. Szélességek rendre: *40%, 30%, 30%* legyenek
 - iv. Keret legyen *1px*
 - v. Szövegdobozok nevei rendre: *txtAddress, txtPhone, txtNumberOfItems*, a *LinkButton* pedig *lbtSave* legyen

Profil adatok módosítása		
Cím:	<input type="text"/>	A cím kötelező
Telefon:	<input type="text"/>	A telefon kötelező
Megjelenítendő elemek száma:	<input type="text"/>	A szám kötelező
<input type="button" value="Mentés"/>		

b. Implementáljuk a *Page_Load* és a Mentés gombra kattintás eseményeket

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

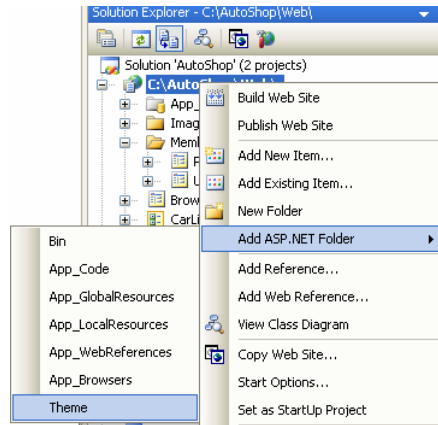
```
{
    if (!IsPostBack)
    {
        txtAddress.Text = Profile.Address;
        txtPhone.Text = Profile.Phone;
        txtNumberOfItems.Text = Profile.NumberOfItems.ToString();
    }
}
```

```
protected void lbtSave_Click(object sender, EventArgs e)
```

```
{
    if (Page.IsValid)
    {
        Profile.Address = txtAddress.Text;
        Profile.Phone = txtPhone.Text;
        Profile.NumberOfItems = Int32.Parse(txtNumberOfItems.Text);
        Profile.Save();
    }
}
```

6. Témák és bőrök

- Adjunk a weboldalhoz egy *Theme* nevű *ASP.NET Folder*-t. Ekkor az *App_Themes* könyvtár alatt automatikusan létrejön egy könyvtár, melynek neve legyen *Yellow*, amibe másoljuk bele a zip állományban található megfelelő könyvtár tartalmát. (Egy skin és egy css fájlt)



- Nézzük meg, hogy mit is találunk a skin állományban. A bőrök használatának az előnye, hogy gyorsan és egyszerűen tudunk megadni egy egységes formázást az összetett vezérlőelemekre. Sajnos a skin állományban nem áll rendelkezésünkre az IntelliSense, így érdemes a formázást az adott oldalon elkészíteni, majd a HTML kódból átmásolni a már kész kódot. Azonban ügyeljünk arra, hogy az ID-t töröljük ki a skin állományból, illetve csak formázási tulajdonságokat állítsunk be.

```
<%-- LinkButton stílusa --%>
<asp:LinkButton runat="server" CssClass="BlackOrange" />

<%-- Vízszintes navigációs menu stílusa --%>
<asp:Menu runat="server" MaximumDynamicDisplayLevels="2" Orientation="Horizontal"
StaticDisplayLevels="2" CssClass="Yellow">
  <StaticMenuItemStyle CssClass="OrangeWhite" />
  <StaticHoverStyle CssClass="OrangeWhite" />
</asp:Menu>
```

- A fent megadott formázások automatikusan érvényesülni fognak minden azonos típusú objektumra. Ha azt szeretnénk, hogy egyes vezérlőelemek másképpen nézzenek ki, akkor a skin fájlban adjunk meg egy *SkinID* attribútumot is. Így lehetőségünk van adott típusú vezérlőelemhez (pl. menu) több különböző skin-t készíteni. A vezérlőelem tulajdonságai között állíthatjuk be, hogy melyik SkinID-val definiált bőrt szeretnénk hozzá használni.
- Ahhoz, hogy az egyes oldalak használják a létrehozott témát, a web.config-ba tegyük be az alábbi sort.

```
<system.web>
  <pages theme="Yellow"></pages>
</system.web>
```

7. WebPartok

1. Nyissuk meg a *Default.aspx* oldalunkat.
2. Húzzunk az oldal tetejére egy WebPartManager vezérlőt a Toolboxról. Fontos, hogy ez legyen az oldal tetején, mert a webkijelzők csak ez után szerepelhetnek.
3. Szúrjunk be egy 3 oszlopos és 1 soros táblázatot, ahol a szélességek legyenek rendre: 30%, 30%, 40%. Illetve állítsuk be a *valign=top*-ot.
4. Tegyük a baloldali és a középső cellába egy-egy *WebPartZone*-t
5. A jobb oldali cellába helyezzünk el egy *DropDownList*-et *ddlModes* névvel, és állítsuk be az *AutoPostBack* tulajdonságát *true*-ra.
6. A *Page_Load* eseményben töltsük fel a *DropDownList*-et a rendelkezésre álló megjelenítési módokkal.

```
protected void Page_Load(object sender, EventArgs e)
{
    if (!IsPostBack)
    {
        foreach (WebPartDisplayMode mode in WebPartManager1.SupportedDisplayModes)
        {
            if (mode.IsEnabled(WebPartManager1))
                ddlModes.Items.Add( new ListItem(mode.Name, mode.Name) );
        }
    }
}
```

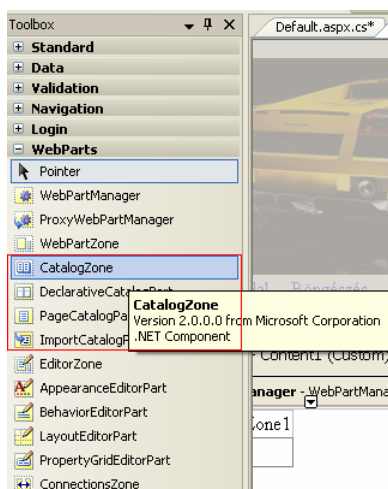
7. Implementáljuk a *DropDownList SelectedIndexChanged* eseményét.

```
protected void ddlModes_SelectedIndexChanged(object sender, EventArgs e)
{
    string selectedMode = ddlModes.SelectedValue;

    WebPartDisplayMode mode = WebPartManager1.SupportedDisplayModes[selectedMode];

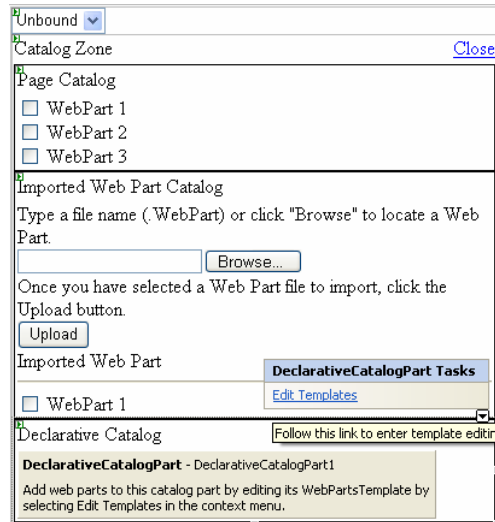
    if (mode != null)
        WebPartManager1.DisplayMode = mode;
}
```

8. A jobb oldali cellába a *DropDownList* alá helyezzünk el egy *CatalogZone* vezérlőt. Ez arra fog szolgálni, hogy futási időben innen lehet majd a webpartokat az oldalra helyezni.

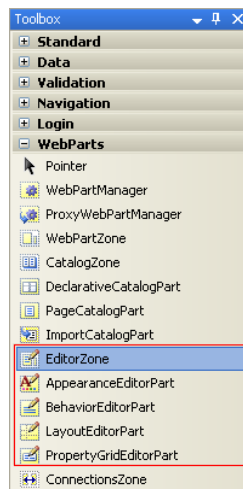


- a. Helyezzünk el egy *PageCatalogPart*-ot a *CatalogZone*-ba. Ha bezárunk egy webkijelzőt, akkor innen fogjuk tudni visszatenni az oldalra.

- b. Adjunk a CatalogZone-hoz egy ImportCatalogPartot. Saját webkijelzőt tudunk feltölteni ennek a vezérlőelemnek a segítségével.
- c. Tegyük a CatalogZone-ba egy DeclarativeCatalogPart-ot, aminek szerkesszük meg a Template-jét. Ha ide behúzzunk egy pár modult (pl. *Categories.ascx*), akkor ezeket később futási időben rá tudjuk tenni az oldalra. Később látunk erre is példát. (*DisplayText.ascx*)



9. Ezt követően a jobb oldali cellába tegyük egy EditorZone-t, amibe helyezzük el az alábbi kontrolokat:

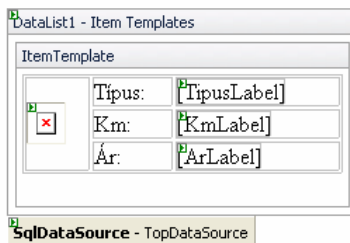


- a. AppearanceEditorPart
- b. BehaviorEditorPart
- c. LayoutEditorPart
- d. PropertyGridEditorPart

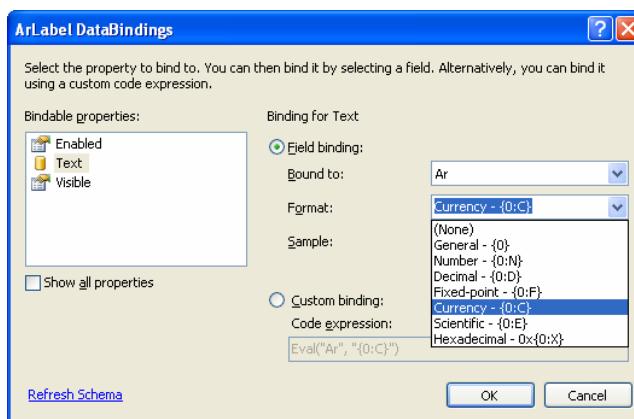
7.1. Legújabb 5 elemet megjelenítő webkijelző elkészítése

1. Adjunk a webalkalmazáshoz egy új Web User Control-t *top5.ascx* névvel.
2. Húzzunk rá egy DataList-et *dtlTop* névvel
 - a. Az adatforrás neve legyen *TopDataSource*
 - b. Az adatokat a *GetTop* tárolt eljárás olvassa ki az adatbázisból.

- c. A bemenő paramétert pedig a Profilból nyerjük ki (*NumberOfItems*), és alapértelmezés szerint legyen 5 az értéke.
3. Szerkesszük meg a DataList Template-jét
- a. Tegyük rá egy 3x3-as táblázatot, ami 100% széles, és a cellái legyenek rendre: 30%, 30%, 40% szélesek. Majd tegyük rá az alábbi vezérlőelemeket:



- b. Az ár mezőre állítsuk be, hogy a pénznemet is megjelenítse.



- c. A képhez kössük hozzá a képfájlt.
- i. `ImageUrl = KepFile`
- ii. `Format = images/Thumbs/{0}`
4. Az elkészült modult húzzuk rá a *Default.aspx* egyik webpartzone-jába.

7.2. DisplayText.ascx elkészítése

- Adjunk a webalkalmazáshoz egy *DisplayText.ascx* nevű modult.
- Tegyük rá egy Label kontrollt *labText* névvel. Alapértelmezés szerint jelenítse meg a következő szöveget: „Üdvözljük az autókereskedésben!”
- Adjunk hozzá egy *TextToShow* tulajdonságot, az alábbi módon

```
private string _textToShow = "Üdvözljük az autókereskedésben";
```

```
[WebBrowsable]
[Personalizable]
public string TextToShow
{
    get { return _textToShow; }
    set { _textToShow = value; }
}
```

4. A *Page_Load*-ban a *labText* szövegét állítsuk át a *textToShow*-ra

```
protected void Page_Load(object sender, EventArgs e)
{
    labText.Text = this._textToShow;
}
```

5. Az elkészített modulban implementáljuk az IWebPart interfészt

```
public partial class DisplayText : System.Web.UI.UserControl, IWebPart
```

6. Miután beírtuk az *IWebPart*-ot az I betű alatt megjelenik egy kis vonal, ha erre rákattintunk, akkor a stúdióval le tudjuk implementáltatni az interfészben definiált függvények és tulajdonságok szkeletonját. Ezt egészítsük ki a következőre.

```
#region IWebPart Members

private string _catalogIconImageUrl;
public string CatalogIconImageUrl
{
    get { return this._catalogIconImageUrl; }
    set { this._catalogIconImageUrl = value; }
}

private string _description;
public string Description
{
    get { return this._description; }
    set { this._description = value; }
}

private string _subTitle;
public string Subtitle
{
    get { return this._subTitle; }
}

private string _title = "DisplayText";
public string Title
{
    get { return this._title; }
    set { this._title = value; }
}

private string _titleIconImageUrl;
public string TitleIconImageUrl
{
    get { return this._titleIconImageUrl; }
    set { this._titleIconImageUrl = value; }
}

private string _titleUrl;
public string TitleUrl
{
    get { return this._titleUrl; }
    set { this._titleUrl = value; }
}

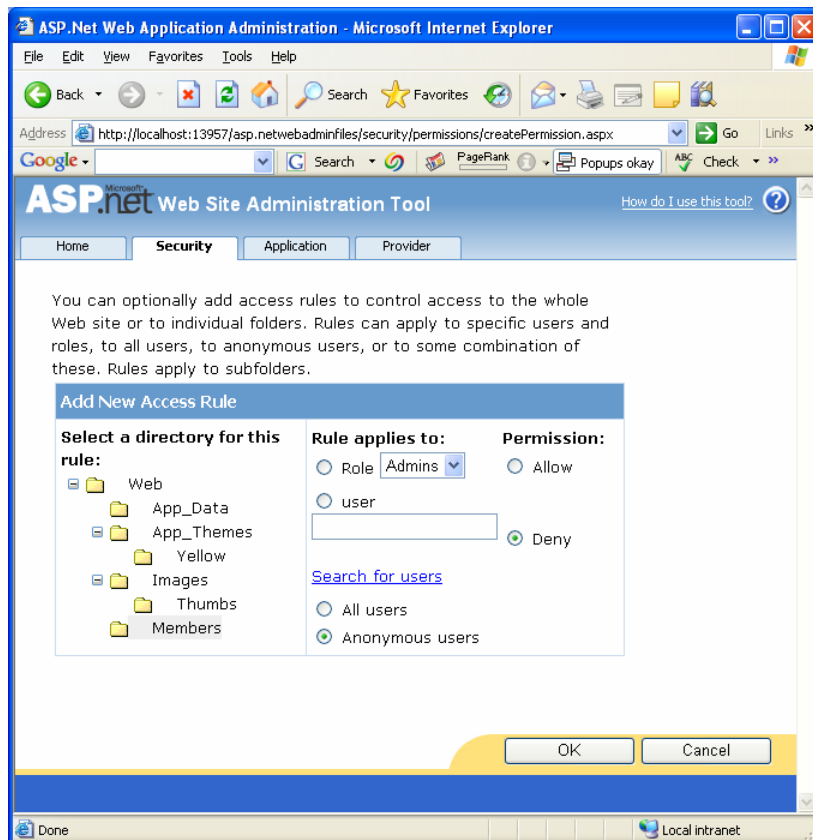
#endregion
```

7. Végül tegyük be a *Default.aspx* egyik webpartzone-jába, illetve a DeclarativeCatalogPart-ba.

8. Indítsuk el az alkalmazásunkat az F5 billentyűvel debug módban.

8. Jogosultságok beállítása

1. WebSite Admin Tools-ban a *Security* fül alatt, a *Create Access Rules* alatt a *Members* könyvtárra adjunk Deny jogot a nem azonosított (anonymous) felhasználókra, és *Allow* jogot az *Authenticated* felhasználóknak.



2. Ezek után így néz ki a web.config, ami a *Members* könyvtár alá kerül. Ahhoz, hogy ez a stúdióban is megjelenjen, frissítsük a könyvtár tartalmát.

```
<?xml version="1.0" encoding="utf-8"?>
<configuration>
  <system.web>
    <authorization>
      <allow roles="Authenticated" />
      <deny users="?" />
    </authorization>
  </system.web>
</configuration>
```

3. Web.config-ban engedélyezzük a sitemap-en a szerepkör szerinti szűrést. Ezzel azt érhetjük el, hogy csak azok a menüpontok fognak látszani a menüben, amit az aktuálisan bejelentkezett felhasználó (vagy a nem bejelentkezett anonymous) ténylegesen meg is nézhet.

```
<siteMap defaultProvider="XmlSiteMapProvider" enabled="true">
  <providers>
    <add name="XmlSiteMapProvider"
      description="Default SiteMap provider."
      type="System.Web.XmlSiteMapProvider, System.Web, Version=2.0.0.0, Culture=neutral,
      PublicKeyToken=b03f5f7f11d50a3a"
      siteMapFile="Web.sitemap"
      securityTrimmingEnabled="true" />
```

```
</providers>  
</siteMap>
```

4. web.sitemap-be tegyük bele a legfelső elemre a *roles="*"*-ot. Ez azt jelenti, hogy a legfelső elemet, ami ebben a példában üres, mindenki láthatja.

```
<?xml version="1.0" encoding="utf-8" ?>  
<siteMap xmlns="http://schemas.microsoft.com/AspNet/SiteMap-File-1.0" >  
  <siteMapNode url="" title="" description="" roles="*">  
    <siteMapNode url="/Default.aspx" title="Kezőoldal" description="" />  
    <siteMapNode url="/Browse.aspx" title="Böngészés" description="" />  
    <siteMapNode url="/Search.aspx" title="Keresés" description="" />  
    <siteMapNode url="/Members/Upload.aspx" title="Autó feltöltés" description="" />  
    <siteMapNode url="/Members/PersonalData.aspx" title="Személyes adatok" description="" />  
  />  
</siteMapNode>  
</siteMap>
```

5. Próbáljuk ki. (F5)