Johanyák Zsolt Csaba: **Webszolgáltatás és XML alapú adatbázis – oktatási segédlet**
http://www.johanyak.hu
e-mail: johanyak.csaba@gamf.kefo.hu

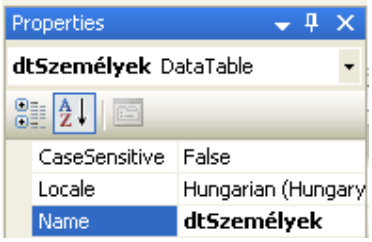
Webszolgáltatás és XML alapú adatbázis

A segédlet célja az, hogy a teljesség igénye nélkül egy egyszerű példán keresztül bemutassa, hogy hogyan készíthetünk egy olyan kétrészes (kétrétegű) alkalmazást, ahol az adatbázis szerver oldalon helyezkedik el, és egy webszolgáltatáson keresztül érhető el. Az adatok egy XML állomány formájában kerülnek tárolásra.

Az alkalmazás nem foglalkozik az adatok titkosításával, a hozzáférési jogosultságok szabályozásával, az adatintegritás megőrzésével, illetve a tranzakciókezeléssel. Feltételezzük, hogy egyszerre nem fordul két ügyfél adatrögzítési kéréssel a kiszolgálóhoz. Az alkalmazást három projekt segítségével valósítjuk meg, amelyek az adatbázishoz, a szerver oldali programhoz és az ügyfélprogramhoz kapcsolódnak. A három projektet egyetlen megoldáson belül készítjük el. Az egyes projektek kipróbálása során a Visual Studio beépített webszerverét használjuk. Mindegyik ismertetésre kerülő projekt a .NET keretrendszer 3.5-ös változatához készül.

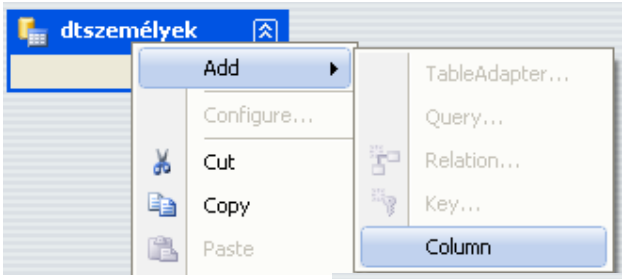
1. Az adatbázis megtervezése

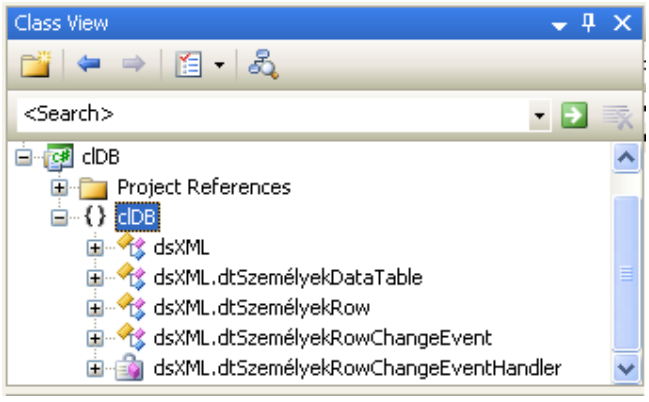
Ennél a feladatnál adatbázis tervezés alatt azt értjük, hogy a Visual Studio sématervezője segítségével leírjuk az adatbázis egyetlen tábláját, és legeneráltatjuk automatikusan az ehhez kapcsolódó típusos DataSet osztályt, illetve a kapcsolódó osztályokat.

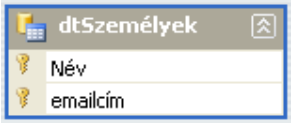


Hozzunk létre egy új Visual Studio projektet, a Visual C# típusból a Class Library sablont választva. Ez biztosítja, hogy az elkészített osztályok egy dll formájában más projektek számára is elérhetőek lesznek. A projekt neve legyen clDB, a megoldás neve legyen wsXML és készítsünk külön könyvtárat a megoldáshoz (Create directory for solution). Töröljük a projekt Class1.cs állományát a Solution Explorer segítségével, majd adjunk hozzá a projekthez egy új DataSet-et Project menü/Add New Item… Dataset sablon, és az állomány neve legyen dsXML.xsd.

A komponenstár (Toolbox) DataSet palettájáról húzzunk egy DataTable komponenst a tervező felületre, majd a Properties ablakban változtassuk meg a tábla nevét dtSzemélyek-re. Jobb egérgombbal kattintva a dtszemélyek táblán a gyorsmenüből válasszuk ki az új





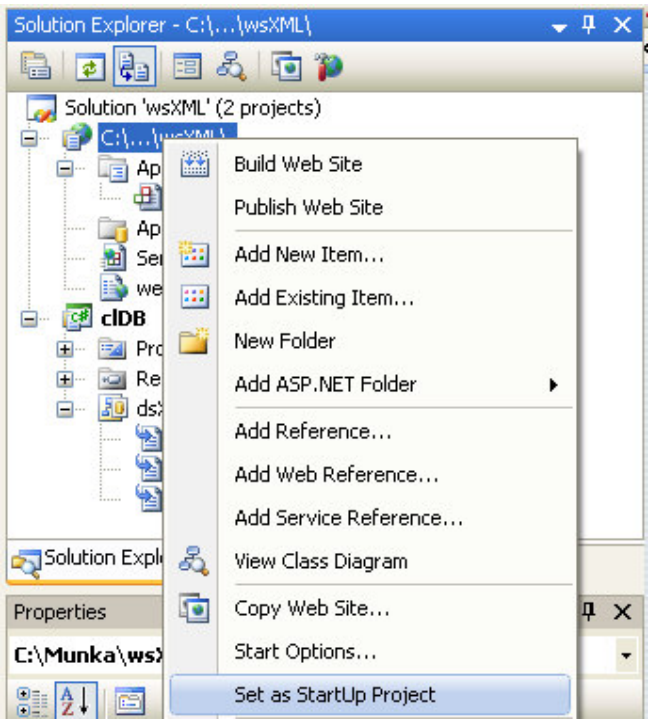


oszlop hozzáadását (Add/Column). A mező neve legyen Név, és rendelkezzen a következő tulajdonságokkal (Properties ablak): AllowDBNull=False, Unique=True, azaz egyetlen rekordban se állhat üresen ez a mező és az értéke egyedi kell legyen. Az előzőekhez hasonlóan

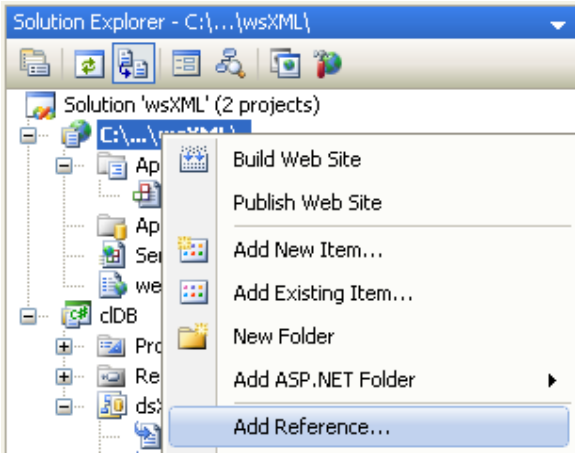
építsünk be még egy mezőt emailcím néven ugyanilyen tulajdonságokkal.
Fordítsuk le a kódot (Build), majd a Class View ablakban ellenőrizzük le a dsXML és a beágyazott osztályok meglétét.

2. Webszolgáltatás elkészítése

Második projektünkben a kiszolgáló alkalmazást készítjük el. Ez metódusok segítségével teszi az ügyfél számára hozzáférhetővé illetve bővíthetővé az adatbázist.
Adjunk a megoldásunkhoz egy új webszolgáltatás típusú projektet (File menü/Add/New Web Site …/ASP.NET Web Service sablon) wsXML néven úgy, hogy a projekt könyvtára a megoldásunk könyvtárába kerüljön.
A webszolgáltatás névterének adjuk meg az XMLpróba nevet. A webszolgáltatás osztályát nevezzük át wsKiszolgáló-ra, az azt tartalmazó service.cs állományt pedig wsKiszolgalo.cs-re. A változtatásokat hajtsuk végre a service.asmx állományban is, ugyanis erre nem terjed ki az automatikus átnevezés.



Állítsuk be aktuális indító projektnek a webszolgáltatás projektjét. Próbáljuk ki az alkalmazást engedélyezve a Debug módban történő futtatást.
Az alkalmazás futásának leállítását (Stop Debugging Shift+F5) követően töröljük a webszolgáltatás forráskódjából (wsKiszolgalo.cs) a HelloWorld metódust, mivel a továbbiakban nem lesz rá szükségünk, majd töröljük a wsKiszolgáló osztály konstruktorában levő két megjegyzésben álló sort is. A webszolgáltatásban fel kívánjuk használni az előző projektben készített DataSet osztályt, ezért projektünkben egy hivatkozást (Reference) kell elhelyezni a clDB projektre. Ehhez jobb egérgombbal kattintunk a Solution Explorerben a wsXML projekten, és a gyorsmenüben az Add Reference … menüpontot választjuk. Kis várakozás után megjelenik az Add Reference párbeszédablak, ahol a Projects fület választjuk. Ekkor egy listaablakban megjelenik a clDB projekt kijelölve. Ezután kattintunk az OK gombon. A Visual Studio a wsXML mappában létrehozott egy Bin könyvtárat, és ide bemásolta a clDB.dll-t. Vegyük fel a használt névterek listájába a clDB és a System.Data névteret.



```
using clDB;
using System.Data;
```

Hozzunk létre egy dsXML típusú és dsXML nevű adattagot a wsKiszolgáló osztályban. Ez fogja tartalmazni az adatbázis memóriabeli változatát. Hozzunk létre egy string típusú és FájlNév nevű adattagot a wsKiszolgáló osztályban. Ebben tároljuk majd az adatbázist tartalmazó XML állomány elérési útvonalát és nevét.

```
dsXML dsXML;
string   FájlNév=
    @"c:\munka\wsXML\wsXML\App_Data\adat.xml";
```

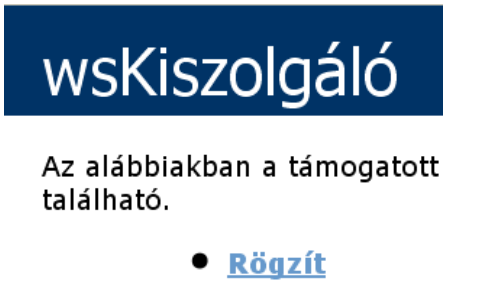
A fenti példában megadtam a teljes elérési útvonalat, ugyanis egyébként debuggolás közben, amikor a Visual Studio beépített webszerverét használjuk, a rendszer a c:\Program Files\Microsoft Visual Studio 9.0\Common7\IDE\ könyvtárban keresné az adatállományt. A program telepítésekor ezt a beállítást majd módosítani kell.
A wsKiszolgáló osztály konstruktorában hozzuk létre a DataSet objektumot, majd próbáljuk meg feltölteni az XML állományból származó adatokkal. A program első futásakor még nem létezik az XML állomány, ezért az adatbeolvasást egy kivételkezelő blokkba építjük, amelynek catch ágát üresen hagyjuk. Ezek után a konstruktor az alábbi lesz:

```
public wsKiszolgáló ()
{
  dsXML=new dsXML();
  try
  {
    dsXML.ReadXml(FájlNév,XmlReadMode.Auto);
  }
  catch
  {
  }
}
```

Készítsünk egy metódust, ami lehetővé teszi újabb rekordok felvitelét az adatbázisba. A metódus két string típusú paramétert vegyen át, visszatérési értéke legyen void. A metódus törzsben adjunk hozzá egy új sort a dtSzemélyek táblához a paraméterként kapott adatokkal, majd mentsük le az adatbázis tartalmát az XML állományba.

```
[WebMethod]
public void Rögzít(string Név, string emailcím)
{
  dsXML.dtSzemélyek.AdddtSzemélyekRow(Név, emailcím);
  dsXML.WriteXml(FájlNév, XmlWriteMode.WriteSchema);
}
```

Próbáljuk ki az alkalmazásunkat, és a webböngészőben megjelenő űrlap segítségével vigyünk fel egy rekordot az adatbázisba.



Ellenőrizzük le, hogy megjelent-e a c:\Munka\wsXML\wsXML\App_Data\ könyvtárban az adat.xml nevű állomány, és hogy tartalmazza-e a felvitt adatokat.

Készítsünk egy metódust, amelynek segítségével lekérdezhetjük, hogy egy adott névhez milyen e-mail cím tartozik. A feladat megoldásához az adattábla Select metódusát fogjuk felhasználni. Mivel minden név és e-mail cím egyedi, ezért a lekérdezést követően az eredményt tartalmazó tömb is csak egysoros lesz. Így az első tömbelemből, amennyiben van ilyen, olvassuk ki az e-mail címet. Amennyiben a keresett név nem található meg az adatbázisban, akkor a metódus null értékkel tér vissza.

```
[WebMethod]
public string EmailLekérdez(string Név)
{
  dsXML.dtSzemélyekRow[] lek =
    (dsXML.dtSzemélyekRow[])dsXML.dtSzemélyek.Select(
    "Név = '" + Név + "'");
  string Eredmény;
  if (lek.Length > 0)
    Eredmény = lek[0].emailcím;
  else
    Eredmény = null;
  return Eredmény;
}
```

Próbáljuk ki a metódust a már felvitt illetve egy nem létező név megadásával.

Tesztelés

A művelet HTTP POST protokollon keresztüli teszteléséhez kattintson az Indítás gombra.

Paraméter	Érték
Név:	Lusta Ábrahám

Indítás

<string>lusta.abraham@nincsilyen.hu</string>

Készítsünk egy metódust, amelynek segítségével lekérdezhetjük, hogy egy adott e-mail címhez milyen név tartozik. A feladat megoldása hasonló az e-mail cím kereséshez.

```
[WebMethod]
public string NévLekérdez(string email)
{
  dsXML.dtSzemélyekRow[] lek =
    (dsXML.dtSzemélyekRow[])dsXML.dtSzemélyek.Select(
    "emailcím = '" + email + "'");
  string Eredmény;
  if (lek.Length > 0)
    Eredmény = lek[0].Név;
  else
    Eredmény = null;
  return Eredmény;
    }
```

Próbáljuk ki a metódust a már felvitt illetve egy nem létező név megadásával.

Utolsó, és egyben legegyszerűbb metódusunk lehetővé teszi a teljes adatbázis lekérdezését.

```
[WebMethod]
public dsXML Teljes()
{
  return dsXML;
}
```

Ezt is kipróbálhatjuk webes felületen, csak itt egy hosszabb XML állomány lesz az eredmény, aminek a végén jelenik meg az általunk felvitt adatrekord.

```
- <diffgr:diffgram>
  - <dsXML>
    - <dtSzemélyek diffgr:id="dtSzemélyek1" msdata:rowOrder="0" diffgr:hasChanges="inserted">
        <Név>Lusta Ábrahám</Név>
        <emailcím>lusta.abraham@nincsilyen.hu</emailcím>
      </dtSzemélyek>
    </dsXML>
  </diffgr:diffgram>
```

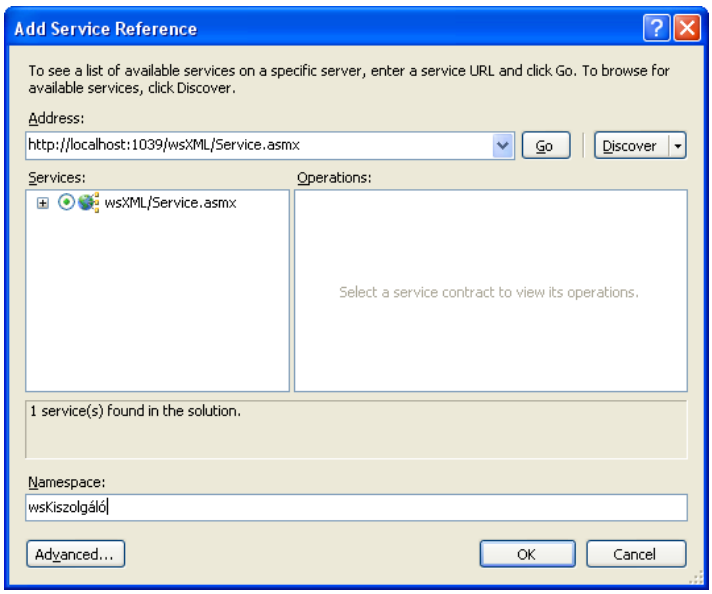
Kiszolgáló oldali alkalmazásunk kipróbálása után áttérhetünk az ügyfélalkalmazás elkészítésére.

3. Ügyfélalkalmazás elkészítése

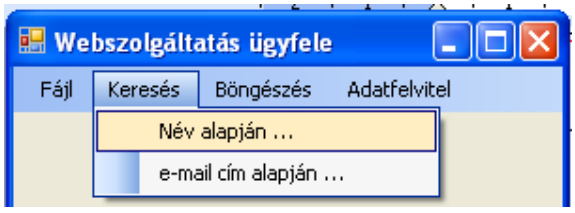
Ügyfélalkalmazásunk egy Windows Forms program lesz, amelynek segítségével megnézzük, hogy hogyan vehetjük igénybe a webszolgáltatás nyújtotta lehetőségeket. Adjunk hozzá a megoldáshoz egy Windows Forms projektet wsUgyfel néven. Solution Explorerben a projektet nevezzük át wsÜgyfél-re, illetve az ablak állományát nevezzük át Form1.cs-ről frmFoablak.cs-re. Az ablak osztálya legyen frmFőablak nevű. Az ablak fejlécében helyezzük el a Webszolgáltatás ügyfele feliratot. Térjünk át kódnézetbe, és változtassuk meg a projekt névterét wsÜgyfél-re.

Állítsuk be alapértelmezett indító projektként a wsÜgyfél projektet. A projektben fel kívánjuk használni az első projektben készített DataSet osztályt, ezért projektünkben egy hivatkozást (Reference) kell elhelyezni a clDB projektre. Ehhez jobb egérgombbal kattintunk a Solution Explorerben a wsÜgyfél projekten, és a gyorsmenüben az Add Reference … menüpontot választjuk. Kis várakozás után megjelenik az Add Reference párbeszédablak, ahol a Projects fület választjuk. Ekkor egy listaablakban megjelenik a clDB projekt kijelölve. Ezután kattintunk az OK gombon.

A webszolgáltatás eléréséhez egy szolgáltatás referenciával egészítjük ki a wsÜgyfél projektet. Ehhez jobb egérgombbal kattintunk a Solution Explorerben a wsÜgyfél projekten, és a gyorsmenüben az Add Service Reference … menüpontot választjuk. Kis várakozás után megjelenik az Add Service Reference párbeszédablak, ahol a Discover listában kiválasztjuk a Services in Solution-t, majd a Namespace mezőben megadjuk, hogy milyen névtérben kívánjuk legenerálni a webszolgáltatást megszemélyesítő helyi osztályokat. Legyen ez a névtér a wsKiszolgáló.



Lássuk el menüvel az alkalmazásunkat. A menüobjektum neve legyen msFőmenü. A Fájl (tsmiFájl) legördülő menüben egy menüpont legyen Kilépés (tsmiKilépés) néven. A Keresés (tsmiKeresés) legördülő menüben két menüpont szerepeljen: Név alapján … (tsmiNévalapján) és e-mail cím alapján (tsmiEmailCímAlapján). A Böngészés (tsmiBöngészés) egy menüpont. Az Adatfelvitel (tsmiAdatfelvitel) szintén menüpont.

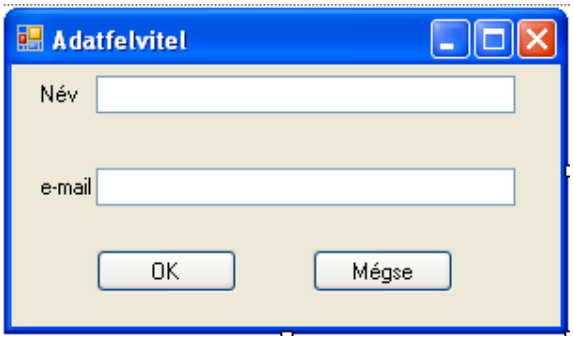


3.1. Kilépés funkció

Elsőként készítsük el a Kilépés menüpont eseménykezelőjét.

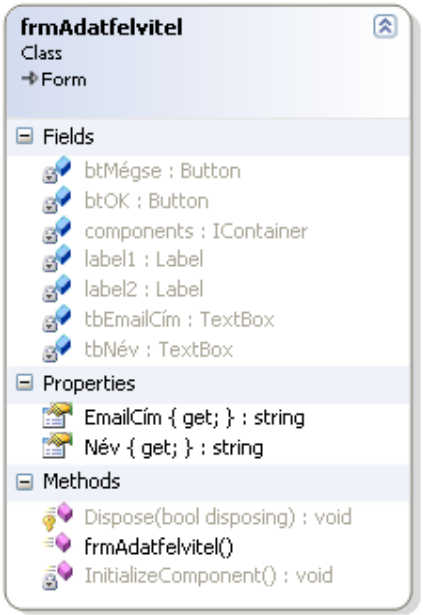
```
private void tsmiKilépés_Click(object sender, EventArgs e)
{
  Application.Exit();
}
```

3.2. Adatfelvitel funkció



Készítsük el az adatfelvitelt megvalósító űrlapot. Project menü/Add Windows Form…. Az állomány neve legyen frmAdatfelvitel.cs. Az ablak fejlécében jelenjen meg az Adatfelvitel felirat. Helyezzünk el két címkét és két szövegmezőt az ablakra a mellékelt kép alapján. A szövegmezők neve legyen tbNév és tbEmailCím. Helyezzünk el továbbá két nyomógombot is az ablakon OK felirattal és btOK névvel valamint Mégse felirattal és btMégse névvel. A btOK nyomógomb DialogResult tulajdonsága legyen OK. A btMégse nyomógomb DialogResult tulajdonsága legyen Cancel. Ezáltal a gombokon történő kattintás a párbeszédablak bezárását eredményezi OK vagy Cancel visszatérési értékkel.

A szövegmezőkbe beírt adatok csak az ablak osztályán belül érhetőek el, ezért készítünk egy-egy tulajdonságot az frmAdatfelvitel osztályban, aminek olvasásával kívülről is hozzáférhetővé válnak ezek az információk. Ehhez készítsük el a wsÜgyfél projekt osztálydiagramját, majd hozzuk létre a két tulajdonságot Név és EmailCím néven. Mindkettő string típusú lesz és mindkettőnél csak a get elérőre van szükségünk.




```
public string Név
{
  get
  {
    return tbNév.Text;
  }
}
```

```
public string EmailCím
{
  get
  {
    return tbEmailCím.Text;
  }
}
```

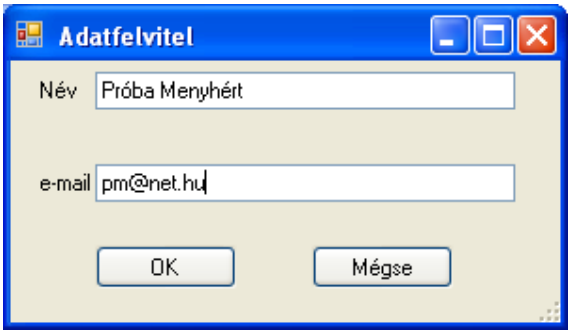
Készítsük el a főablak Adatfelvitel menüpontjának eseménykezelőjét. Ebben bekérjük a felhasználótól az adatokat, majd a webszolgáltatás segítségével tároljuk azokat a szerveren. Ehhez létrehozunk az frmFőablak osztályban egy adattagot a webszolgáltatáshoz.

```
wsKiszolgáló.wsKiszolgálóSoapClient wsKiszolgáló=new
  wsÜgyfél.wsKiszolgáló.wsKiszolgálóSoapClient();
```

Az adatfelvitel menüpont eseménykezelőjében létrehozzuk az frmAdatfelvitel párbeszédablak egy példányát, majd megjelenítjük azt. Ha a felhasználó OK-val zárja be az ablakot, akkor a webszolgálttás Rögzít metódusával tároljuk az adatokat a kiszolgálón.

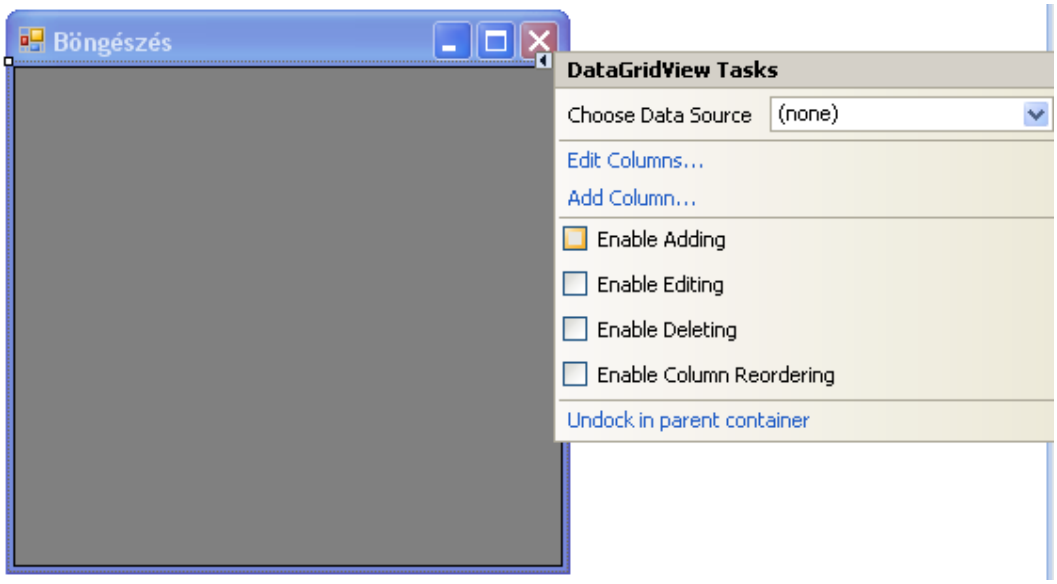
```
private void tsmiAdatfelvitel_Click(object sender,
  EventArgs e)
{
  frmAdatfelvitel faf=new frmAdatfelvitel();
  if(faf.ShowDialog()==DialogResult.OK)
  { wsKiszolgáló.Rögzít(faf.Név,faf.EmailCím);
  }
}
```

Próbáljuk ki az adatfelvitelt, majd ellenőrizzük le annak eredményességét, pl. az XML állomány tartalmának megtekintésével.



3.3. Böngészés funkció

A böngészés funkció (menüpont) azt jelenti, hogy egy adatrácsban megjelenítjük a teljes adattáblát. Először készítsük el az adatmegjelenítést végző ablakot. A Form állomány neve legyen frmBongeszes.cs, az ablakosztály neve legyen frmBöngészés, az ablak felirata legyen Böngészés. Az ablakra helyezzünk el egy adatrácsot dgvRács néven úgy, hogy töltse ki a teljes ablakot. A törlést, szerkesztést és hozzáadást tiltsuk le.



Az adatrács DataSource tulajdonságát tegyük elérhetővé az frmBöngészés ablak osztályán kívülről is. Erre azért van szükség, hogy az ablak megjelenítése után hozzárendelhessük a megjeleníteni kívánt adattáblát.
A feladatot egy tulajdonsággal oldjuk meg, amelyet az frmBöngészés osztályban helyezünk el. A tulajdonságnak csak set elérőre van szüksége.

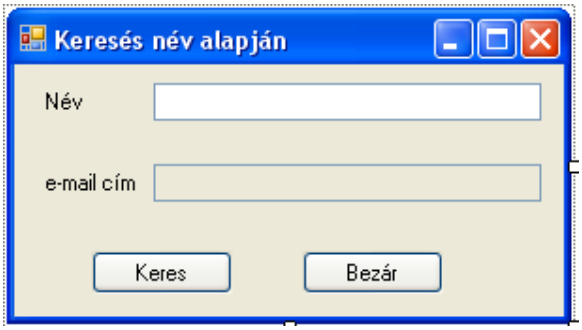
```
public DataTable Adatforrás
{
  set
  { dgvRács.DataSource=value;
  }
}
```

Készítsünk egy eseménykezelőt a főablak Böngészés menüpontjához. Ebben létrehozzuk az frmBöngésző ablak egy példányát, a webszolgáltatás segítségével lekérdezzük az adatbázist, és gondoskodunk annak megjelenítéséről. A feladatot megoldó kódrészlet a következő.

```
private void tsmiBöngészés_Click(object sender,
  EventArgs e)
{
  frmBöngészés fb=new frmBöngészés();
  wsKiszolgáló.dsXML.dtSzemélyekDataTable dtSzemélyek=
    wsKiszolgáló.Teljes().dtSzemélyek;
  fb.Adatforrás=dtSzemélyek;
  fb.ShowDialog();
}
```

Próbáljuk ki a böngészés funkciót.

3.4. Keresés név alapján



A név alapján történő e-mail cím keresés megvalósításához elsőként hozzuk létre az űrlapot. Az ablakosztály állományának neve legyen frmKeresesNevAlapjan.cs, az osztály neve pedig legyen frmKeresésNévAlapján. Az ablak fejlécében a „Keresés név alapján” szöveg jelenjen meg.
A szokásos elnevezési konvenciót alkalmazzuk. A szövegmezők neve legyen tbNév és tbEmailCím, a nyomógomboké btKeres és btBezár. A tbEmailCím legyen csak olvasható (ReadOnly=True). A Bezár gombon történő kattintás idézze elő az ablak bezárását automatikusan (DialogResult=OK).
A Keres gombon történő kattintás hatására meghívjuk a webszolgáltatás EmailLekérdez metódusát. A webszolgáltatás eléréséhez létrehozunk egy adattagot az frmKeresésNévAlapján osztályban, feladata a webszolgáltatás referenciájának tárolása. Ez az adattag az ablak osztályán kívülről beállítható kell legyen, ezért a hozzáférést public szinten állítjuk be.

```
public wsKiszolgáló.wsKiszolgálóSoapClient wsKiszolgáló;
```

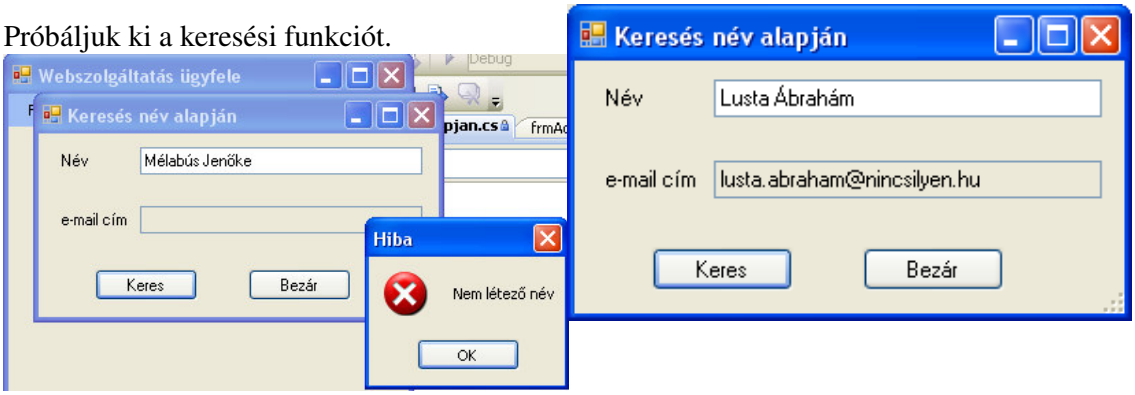
Készítsünk egy eseménykezelőt a Keres nyomógombhoz. Ebben a tbNév mezőből kiolvasott névvel meghívjuk a webszolgáltatás EmailLekérdez metódusát, és ha az eredmény nem null, akkor a visszakapott e-mail címet beírjuk a tbEmailCím mezőbe. Ellenkező esetben hibaüzenetet jelenítünk meg.

```
private void btKeres_Click(object sender, EventArgs e)
{
  tbEmailCím.Text = "";
  string email = wsKiszolgáló.EmailLekérdez(tbNév.Text);
  if (email == null)
    MessageBox.Show("Nem létező név", "Hiba",
      MessageBoxButtons.OK, MessageBoxIcon.Error);
  else
    tbEmailCím.Text = email;
}
```

Készítsünk egy eseménykezelőt a főablak Keresés/Név alapján … menüpontjához. Ebben létrehozzuk a párbeszédablakot, beállítjuk a wsKiszolgáló adattag értékét, majd megjelenítjük a párbeszédablakot.

```
private void tsmiNévAlapján_Click(object sender,
  EventArgs e)
{
  frmKeresésNévAlapján fkna=new frmKeresésNévAlapján();
  fkna.wsKiszolgáló=wsKiszolgáló;
  fkna.ShowDialog();
}
```

Próbáljuk ki a keresési funkciót.



4. Feladat

1. Készítsük el az e-mail cím alapján történő keresést.
2. Egészítsük ki az adatfelviteli űrlap osztályát a bevitt adatok ellenőrzésével.