

## **ELŐZMÉNYEK:**

Cél: Oracle adatbázis elérése és használata .NET kliensből, .NET komponensből, .NET szoftverből.

A jelenlegi nekünk az önálló labor 1. féléve volt és mivel ez a technológia mindkettőnknek új volt a félévet arra szántuk, hogy megismerjük és beletanuljunk a technológiákba, nézzük meg milyen problémák merülnek fel amikor ilyet fejlesztünk.

## **.NET-ES ORACLE TECHNOLÓGIÁK**

### ORACLE DEVELOPER TOOLS FOR .NET

- integráció a Server Explorer-be
- Automatikus .NET kód generálás (drag and drop):
  - Datasources window
  - DataSet designer
  - TableAdapter Configuration wizard
- PL/SQL Editor and Debugger ⇒ az sql kód létrehozható, szerkeszthető és futtatható közvetlenül Oracle adatbázison VS-en belülről
- .NET Stored Procedure Deployment ⇒ .NET-es tárolt eljárások és függvények a MS Visual Studioban fejleszthetők és az elkészült .NET-es projekt fordítást követően .NET Deployment Wizard-dal telepíthető az Oracle szerverre amennyiben az Oracle Windows-on fut, .NET futtatókörnyezet van a gépen és telepítésre került az Oracle Database Extensions for .NET nevű server oldali Oracle termék, mely Oracle Database Windows környezetben futó verziójának azon funkciója, mely lehetővé teszi .NET felügyelt nyelveiben megírt tárolt eljárások és függvények futtatását.

### ORACLE DATA PROVIDER FOR .NET

Az ODP.NET egy natív .NET adathozzáférési driver az Oracle adatbázishoz, mely más .NET driver-ekhez képest:

- nagyobb teljesítményt nyújt az Oracle adatbázis .NET környezetből való optimalizált elérése révén

- utasítás gyorsítótárazás (statement caching)

Egy konkrét lekérdezést vagy PL/SQL utasítást többszöri végrehajtása

esetén, az ODP.NET az ismételt végrehajtási időt csökkenti, a szintaktikai elemzés megismétlésének kihagyásával. Ehhez viszont a lekérdezésekhez tartozó értékek paraméterekben történő átadása szükséges.

- kapcsolat tárazás (connection pooling)  
A kapcsolatok tárazását számos beállítással lehet testreszabni (kapcsolat élettartam és időtúllépés, minimális és maximális tárméret, granularitás)
  - adatlehívási méret szabályozása (controlling data fetch size)  
ODP.NET-ben az egy menetben lehívandó adatok mennyisége szabályozható és így a kliens feldolgozási sebességéhez igazítható (OracleCommand.FetchSize)
  - optimalizált LOB adatok  
Itt szintén szabályozható az egy menetben lehívandó adatmennyiség, az eredményhalmaz lehívása elhalasztható az adatok tényleges olvasásáig, és a véletlen elérés is támogatott (az eredményhalmaz kinyerésének elhalasztása az adatok tényleges olvasásáig)
  - tömb adatok átadásának képessége az adatbázis és a .NET Framework között.
  - kliens eredmény cache (11g) a válaszidő csökkentésére hivatott ismételten végrehajtott lekérdezések esetén, a kliensoldalon az SQL lekérdezések eredményének memóriában való gyorsítótárazásával, a kliens számára transzparens módon
- lehetővé teszi fejlett Oracle adatbázis funkciók kihasználását
    - kapcsolat tárazás a Oracle RAC (Real Application Clusters) fürtözéses adatbázis számára. A kapcsolat tárazás során egyenletes terhelés elosztást biztosít a csomópontok számára még csomópont hozzáadása vagy eltávolítása után is. Továbbá a megszakadt RAC kapcsolatok automatikusan eltávolításra kerülnek a tárból
    - XML adatbázis (native XML data type) (XML support)  
ODP.NET lehetővé teszi .NET kliensek számára XML adatbázis teljes funkcionalitását, lehetővé téve a fejlesztők számára, hogy megosszák és megváltoztassák XML-t az adatbázis és a .NET között. Az ODP.NET két natív XML adattípust tartalmaz. Az XML adatbázis a szerveren kínál XML szolgáltatásokat. Az ODP.NET lehetővé teszi a relációs és objektum-relációs adatok elérését mint XML-t, az XML megváltoztatható és visszamenthető a szerverre mint XML vagy mint relációs adat.
    - támogatja a natív Oracle adattípusokat (REF Cursor, LOB(CLOB, BLOB, NCLOB), BFILE, LONG, RAW, LONG RAW és N adattípusok), melyek a .NET-es típusokhoz hasonló módon hozzáférhetők továbbá

többfunkcionalitásuk kihasználható. Az Oracle adattípusok natívan tárolhatók a .NET-es DataSet-ben.

- Adatbázis változás értesítés (Database Change Notification)  
ODP.NET kliens az adatbázisban történő változásokról értesítést kap még abban az esetben is ha az aktív kapcsolat már le lett zárva.

## ORACLE PROVIDERS FOR ASP.NET

A .Net 2.0-ás Frameworktől kezdődően az ASP.Net az állapot adatbázisban történő eltárolására különféle szolgáltatásokat nyújt. Ezzel egyszerű hozzáférhetőséget biztosít az alkalmazások számára az adatokhoz, míg azok egyformán elérhetők az összes webszerveren. Az OP for ASP.NET ezeket a szolgáltatásokat teszi elérhetővé Oracle adatbázis használatával. Úgy lett kialakítva, hogy a meglévő ASP.NET szolgáltatásokhoz teljesen hasonló legyen mind használatba, mind kódba, így biztosítva egyszerű használatot.

Az OP.ASP telepítésével egy egyszerű kliens is települ, mellyel elérhetjük az adatbázist. Mivel mi a teljes csomagot telepítettük, így ezt nem használtuk. (1perc)

## ADO.NET

Adatbázisnál egy kulcselem, hogy a subsystem a .NET-nek az ADO.NET. alapvetően milyen jellegű fogalmak jönnek amik eltérnek az eddigiektől, Az volt a cél, hogy ezzel a technológiával – adateléréssel .NET felől – foglalkozzunk. Nézzük meg akár WinForms-os kliensek alól, akár ASP.NET-es kliensek alól. Az a cél, hogy az adattal tudjunk bánni.

- ConnectionString  
lényeg, hogy a provider Oracle.DataAccess.Client (ODP.NET) legyen és ne System.Data.OracleClient (MS .NET Data Provider for Oracle)
- DataSet-ek
- SqlDataSource
- Stored Procedure

## ORACLE TELEPÍTÉSE

- Oracle DB

A Oracle-t egy önálló gépre telepítettük, melyet mindketten hálózaton keresztül értünk el. Telepítettük továbbá az Oracle Database Extensions for .NET-et is a szerverre, mely a .NET-es tárolt eljárások telepítését és futtatását teszi lehetővé az adatbázisszerveren.

A telepítést követően mindkettőnk számára létrehoztunk egy-egy felhasználót az adatbázisban. Megadtuk számukra a CREATE SESSION, CREATE TABLE és a CREATE VIEW ill. a CREATE PROCEDURE jogosultságokat valamint kvótát biztosítottunk számukra a USERS táblatérben. Minden további beállítást (beleértve a sémák létrehozását is VS-ből végeztük.)

- **ODAC**

A kliens gépekre telepítettük az Oracle Data Access Components nevű termékét, mely lehetővé teszi az eddig tárgyalt Oracle technológiák használatát a kliens gépen. (ODP.NET, Oracle Providers for ASP.NET, ODT for Visual Studio)

- **Verziók (10g – 11g)**

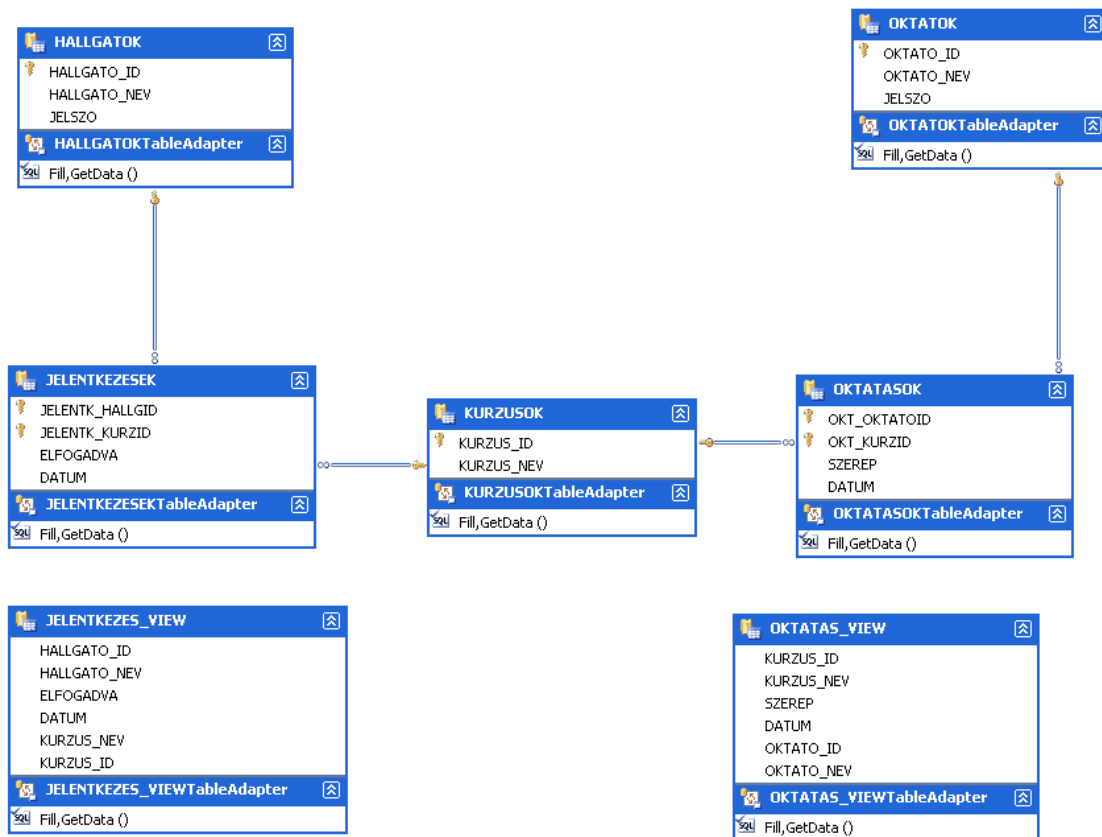
Először telepítettük a Oracle 10XE-t, majd amikor a félév közben megjelent a 11g Windows-os verziója telepítettük azt is. Ezután a kettő között variáltunk pl. 10g kliens alól próbáltunk kapcsolódni 11g server-hez, ekkor azonban számos problémába ütköztünk. (pl. password kezelés, vagy stúdió alóli teljes használhatatlanság). Majd egy hónapnyi kísérletezést követően teljes újratelepítést követően áttértünk kliens oldalon is a 11g-re és ekkor a problémák nagy része megoldódott.

## **SAJÁT PROGRAMOK**

Az Oracle adatbáziskezelőhöz két .NET alapú alkalmazást készítettünk, mindkettőből egy WinForms-os és ASP.NET-es változatot. Az egyik egy kurzusjelentkezési rendszer, melyben oktatók kiírhatnak kurzusokat, a hallgatók jelentkezhetnek rájuk, az oktatóknak pedig lehetősége van a hallgatók kurzus-jelenetkezéseit elfogadni vagy visszautasítani.

Az adatbázisséma 3 alaptáblából áll: HALGATOK, OKTATOK, KURZUSOK ill. 2 db kapcsolótáblából: JELENTKEZESEK , OKTATASOK melyek elsődleges kulcsként a HALLGATOK+KURZUSOK ill. az OKTATOK+KURZUSOK táblák elsődleges kulcsait tartalmazzák mint idegen kulcsokat. Az alábbi ábrán az Oracle-ben létrehozott adatbázis sémából a Visual Stúdióban létrehozott típusos DataSet ábrája látható. A

JELENTKEZES\_VIEW és OKTATAS\_VIEW nézetek az egyszerűbb lekérdezések érdekében lettek létrehozva.



A következő ábrán a két kliens bejelentkezési ablaka látszik:

The WinForms login form is titled "Kurszus jelentkezési rendszer". It contains a dropdown menu for "Szerepkör:" with "Hallgatók" selected. Below it are text input fields for "Felhasználónév:" and "Jelszó:". A "Bejelentkezés" button is located at the bottom of the form.

WinForms

The ASP.NET login form is titled "KURZUS JELENTKEZÉSI RENDSZER". It features a dropdown menu for "Szerepkör:" with "Hallgató" selected. Below it are text input fields for "Felhasználónév:" and "Jelszó:". A "Bejelentkezés" button is located at the bottom of the form.

ASP.NET

A WinForms-os alkalmazás indítást követően feltölti a DataSet-et az adatbázisból a Visual studio által generált TableAdapter objektumok Fill() metódusaival. Majd a felhasználói adatokat a DataSet-ben történő kereséssel ellenőrzi, a DataSet HALLGATOK táblájának Select metódusának meghívásával:

```
strExpr = "hallgato_nev='" + username.Text + "' and jelszo='" +
        password.Text.GetHashCode().ToString() + "'";
foundRows = dS.HALLGATOK.Select(strExpr);
```

Az ASP.NET -es alkalmazásban az adatelérés hasonló a különbség annyi, hogy itt a DataSet objektumot a cache-ből kell előszedni.

```
strExpr = "hallgato_nev='" + username.Text + "' and jelszo='" +
        password.Text.GetHashCode().ToString() + "'";
foundRows = DSCode.GetDS(Cache).HALLGATOK.Select(strExpr);
```

Ha a felhasználó oktatóként jelentkezik be a következő ablak jelenik meg számára a WinForms ill. a ASP.NET kliens esetén:

Az Új feliratú gombra kattintva lehet új kurzust létrehozni a rendszerben. WinForms kliens esetén a DataSet objektum KURZUSOK táblájában jön létre egy új sor ahova az új kurzus adatai kerülnek. Az új sor létrehozása a Visual Stúdió által generált DataSet objektum NewKURZUSOKRow(); metódusával van létrehozva:

```
DS.KURZUSOKRow p = ds.KURZUSOK.NewKURZUSOKRow();
long k=System.DateTime.Now.GetHashCode();
p.KURZUS_ID = (k>0)?k:-1*k;
p.KURZUS_NEV = adat_ablak.nev.Text;
ds.KURZUSOK.Rows.Add(p);
```

Módosítás esetén a már létező sor a DataSet .FindByKURZUS\_ID() metódusával kereshető elő a KURZUSOK táblából:

```
DS.KURZUSOKRow p =
    ds.KURZUSOK.FindByKURZUS_ID((long) kurzusok.SelectedValue);
```

ASP.NET-es kliens esetén adatbázis elérése DataSource objektumokkal történik.

Az új kurzus létrehozása és a módosítása a DataSource objektum Insert() és Update() metódusaival lehetséges. A DataSource objektum UpdateQuery és InsertQuery

tulajdonságaiban vagy sql parancsot kell megadni vagy egy előre elkészített tárolt eljárást. Jelen esetben pl. az IupdateQuery-ben az alábbi SQL utasítás van megadva:

```
UPDATE KURZUSOK SET KURZUS_NEV=:KURZUS_NEV WHERE KURZUS_ID=:KURZUS_ID
```

A KURZUS\_NEV paraméter a textbox-hoz van kötve, amit a felhasználó kitöltött. A KURZUS\_ID paraméter egy session változóval kerül átadásra:

```
long k = System.DateTime.Now.GetHashCode();  
Session["newCourseID"] = (k > 0) ? k : -1 * k;
```

A saját kurzusok tábla GridView segítségével van megjelenítve mely egy DataSource objektum SelectQuery tulajdonságában megadott sql lekérdezéshez van kötve:

```
SELECT * FROM OKTATAS_VIEW WHERE OKTATO_ID = :OKTATO_ID
```

A hallgatók táblát egy másik GridView jeleníti meg amelyhez tartozó DataSource SelectQuery tulajdonságában megadott

```
SELECT * FROM JELENTKEZES_VIEW WHERE KURZUS_ID =:KURZUS_ID
```

sql utasítás KURZUS\_ID paramétere a saját\_kurzusok GridView SelectedValue tulajdonsága határozza meg.

Ha az oktató megváltoztatja a hallgatójelentkezés elfogadottságának állapotát akkor az a HallgatokDataSource Update metódusában megdott UPDATE\_ELF\_IN\_JELENTKEZESEK tárolt eljárással kerül az adatbázisban is azonnal módosításra.

## SAJÁT PROGRAMOK #2

### A PROGRAM LEÍRÁSA

Egy egyszerű bevásárlói rendszer elkészítése volt a cél. Adminisztrátori és felhasználói hozzáféréssel. A felhasználó (vásárló) a termékek között tud válogatni, majd egy kiválasztott termékből bizonyos számút a kosárába helyezni. A kosárból visszarakhatja a terméket, vagy a kosár teljes tartalmát megvásárolhatja. Az adminisztrátor a termékeket és a termékkategóriákat módosíthatja, továbbá a felhasználók kosarait ürítheti. Lehetősége van még az egyes felhasználók jelszavának módosítására is.

### ADATBÁZIS TERVEZÉSE

Az adatokat adatbázisban fogom tárolni, ehhez szükségem van Products (termékek), Categories (kategóriák), Basket (kosár) táblákra. A felhasználók kezelését az ASP.NET providereivel (Membership és Role) szeretném megoldani. Két kliens alkalmazást készítettem hozzá, Winforms és ASP.NET klienst. Mivel Winforms alól az ASP.NET providereinek szolgáltatását nem lehet elérni, ezért szükségem van még egy Users (felhasználók) táblára is és egy Users\_View nézetre, ami az ASP:NET-es alkalmazás felhasználóinak alapadatait tartalmazza (név és jogkör).

## TELEPÍTÉS

A félév során többféle verziójú Oracle DB Servert és ODAC csomagot telepítettem, sajnos gyakori inkompatibilitási problémával találkoztam, ezeket nehezen tudtam orvosolni, sokszori újratelepítés után általában az újabb verzió megjelenése hozta a megoldást. Végül a Visual Studio 2008 Professional Edition, ODAC 11.1.0.6.10 és az Oracle Database Server 11.1.0.6.0 verziójával dolgoztam.

## WINFORMS ALAKLMAZÁS

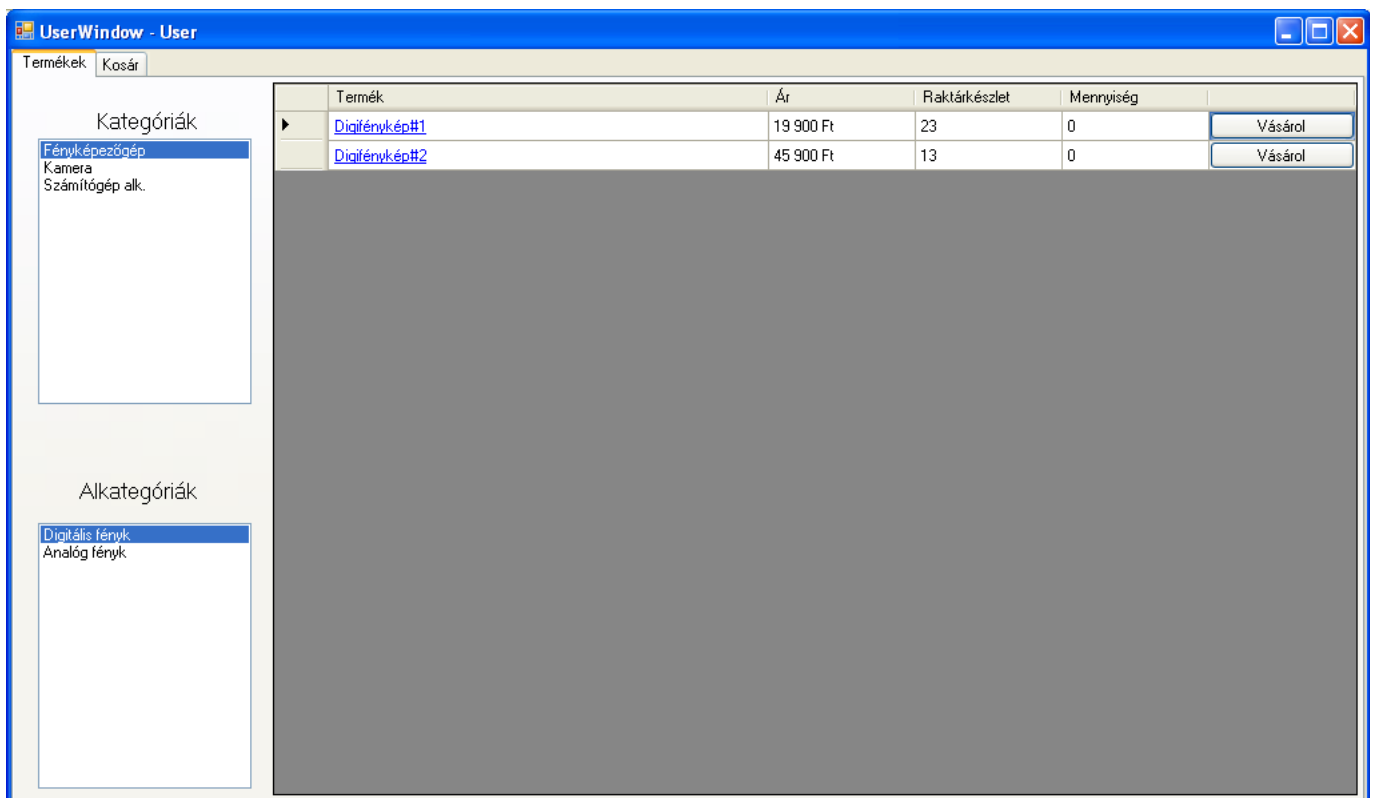
Az alkalmazáshoz 3 ablak tartozik: bejelentkező, adminisztrátori és felhasználói ablak. Az adatokat datasetben tárolom, ebben megtalálhatók az adatbázis táblái, és az ezekből lezármaztatott táblák, melyek az alkalmazáshoz igazodnak: Users, Products, Categories, Basket és Users\_View. A Products\_DT a kiválasztott kategória termékeit tartalmazza, a Main- és SubCategories\_DT a fő és mellékkategóriákat tartalmazza, a Basket\_DT pedig a felhasználó kosarának tartalmát.

## A BEJELENTKEZŐ ABLAK

A bejelentkező ablakban lehet a programba bejelentkezni, meg kell adni a felhasználónevet és a hozzá tartozó jelszót, majd a Bejelentkezés megnyomására a program megpróbál az adott adatokkal belépni. Ellenőrzi, hogy az adatmezők nem üresek, majd a Users\_View datatable-ből ellenőrzi, hogy létezik-e az adott felhasználó.

```
Database.USERS_VIEWRow dr = null;
foreach (DataRow row in database.USERS_VIEW.Rows)
{
    if (((Database.USERS_VIEWRow) row).USERNAME == TB_UserName.Text)
    {
        dr = (Database.USERS_VIEWRow) row;
    }
}
```

Ha a felhasználó létezik, akkor ellenőrzi, hogy lett-e beállítva jelszó, illetve egyezik-e. Ha nem volt beállítva, akkor felvesz a Users táblában egy új sort, ami az új felhasználónév és jelszó párost tartalmazza. A felhasználó jogkörének megfelelő ablakot nyitja meg, majd a bejelentkező ablakot „bezárja”(Visible = false).



## FELHASZNÁLÓI ABLAK

A felhasználói ablak két fülből áll (TabControl segítségével két panel). Az egyik fülön a termékek közül lehet válogatni, míg a másikon a kosarunk tartalmát látjuk.

Kategóriák között két ListBox segítségével navigálhatunk, majd a termékeket egy DataGridView-n tekinthetjük meg. Adatkötéssel a dataset tábláihoz kötöttem ezeket. A Listboxok SelectedItem\_Changed eseményét kidolgoztam, itt végzek műveleteket az adattáblákon.

```
private void LB_MainCategories_SelectedIndexChanged(object sender, EventArgs e)
{
    if (LB_MainCategories.SelectedIndex == -1) return;
    database.SubCategoriesDT.Clear();
    decimal ParentID = (decimal)LB_MainCategories.SelectedValue;
    DataRow[] rows = database.CATEGORY.Select("ParentID=" +
ParentID);
    foreach (Database.CATEGORYRow row in rows)
    {
        database.SubCategoriesDT.Rows.Add(row.ItemArray);
    }
    database.ProductsDT.Clear();
}
private void LB_SubCategories_SelectedIndexChanged(object sender, EventArgs e)
{
    if (LB_SubCategories.SelectedIndex == -1) return;
    database.ProductsDT.Clear();
    decimal CategoryID = (decimal)LB_SubCategories.SelectedValue;
    DataRow[] rows = database.PRODUCTS.Select("CategoryID=" +
CategoryID);
    foreach (Database.PRODUCTSRow row in rows)
    {
        database.ProductsDT.Rows.Add(row.ItemArray);
    }
}
```

A termékeket tartalmazó DataGridView-t kiegészítettem két cellával, egy a vásárolandó mennyiséget tartalmazó TextBox, a másik a vásárlást engedélyező gomb. Továbbá a termék nevét tartalmazó cellát LinkColumn cellává változtattam, erre kattintva előugrik a termék adataival egy MessageBox. A cellaesemények kezelését egy későbbi részben fejtem ki. A kosár ablakban látható a felhasználó kosarának tartalma. Itt is egy DataGridView-ban látható a kosár tartalma. A lapra való belépéskor tölti fel a személyes kosár tartalmát (Basket\_DT). Ebből a kosárból való törléskor egy saját függvénnyel visszaállítom a raktár tartalmát.

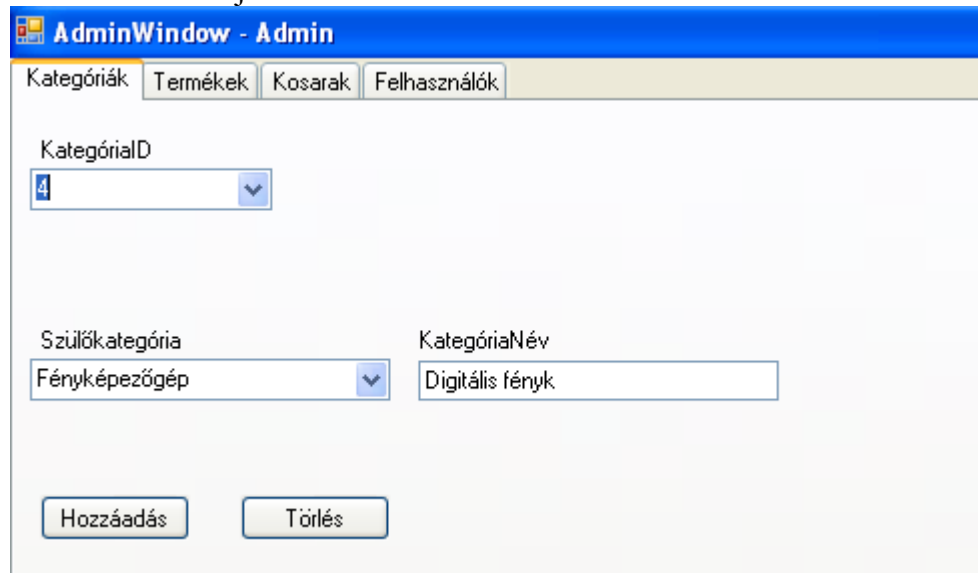
```
private void DGW_BasketDT_UserDeletingRow(object sender, DataGridViewRowCancelEventArgs e)
{
    Database.BasketDTRow row =
(Database.BasketDTRow) database.BasketDT.Rows[e.Row.Index];
    DataRow[] row1 = database.BASKET.Select("ITEMID=" +
row.ITEMID);
    DataRow[] row2 = database.PRODUCTS.Select("PRODUCTID=" +
((Database.BASKETRow) row1[0]).PRODUCTID);
    ((Database.PRODUCTSRow) row2[0]).QUANTITY -= row.QUANTITY;
    database.BASKET.Rows.Remove(row1[0]);
}
```

}

Vásárlás gomb megnyomására üríti a kosarat és kiírja a vásárlás végösszegét.

### ADMINISZTRÁTORI ABLAK

Az adminisztrátori ablak 4 fülből áll. Itt módosíthatjuk a kategóriákat, termékeket, kosarakat és a felhasználók jelszavait.



A kategóriák fülön adatkötéssel a KategóriaId ComboBox kiválasztott elemének megfelelő adatok jelennek meg a többi helyen. Hozzáadáskor egy üres főkategória jön létre név nélkül. Törléskor az aktuális kategória törlődik, ha nincsenek hozzá tartozó alkategóriák, illetve termékek. A termékek fülön hasonlóan láthatók az egyes termékek. A kosarak és felhasználók fülön egyszerűen egy DataGridView-ban jelennek meg az adatok.

### FEJLESZTÉSI TAPASZTALATOK

#### AZ OP FOR ASP.NET KONFIGURÁLÁSA:

Az OP for ASP.NET telepítését a VS-ból tehetjük, meg melybe az ODT-nak köszönhetően beépült egy SQL\*Plus scrip futtató eszköz. A korábbi verziókban egyenként kellett telepítenünk az egyes szolgáltatásokat, de az utolsó Beta2-es verzióban már szerepel egy az összes szolgáltatást telepítő script is. Ha ezt megtettük már csak a számítógépünk machine.config-jában `/*Windows//Microsoft.NET/Framework/verzió/Config*/` kell beállítanunk az elérést. Ezután az ASP.NET Web Site Administration Tools-ban be kell állítanunk a megfelelő providereket. És ezzel be is fejeztük a konfigurálást. Mivel a providerek ASP.NET alá íródtak, így az általuk kínált szolgáltatások sajnos Winforms alól nem érhetőek el, így ott a felhasználó adminisztrációra saját megoldást kellett alkalmazni.

## DATAGRIDVIEW, CELLAESEMÉNYEK KEZELÉSE

Winforms alatt gyakran használunk DataGridView-t az adatok megjelenítésére. Gyakran felmerülő probléma, hogy szeretnénk, ha a felhasználó az adott cellára kattintva valamilyen szolgáltatást érne el. Ezt a DGV CellContentClick eseményét kezelve valósíthatjuk meg. A függvény paraméterei itt láthatók. A sender jelen esetben a DGV és e DGVEventsArgs. Ellenőrizhetjük, hogy melyik cellára kattintottunk, úgy hogy ellenőrizzük hogy az e-ben található cellindex által azonosított cella megegyezik-e az általunk kívánttal.

## A KAPCSOLAT HOZZÁADÁSA A SERVER EXPLORERHEZ

Az ODT.NET beépülő szolgáltatásai révén lehetőségünk van a Server Explorerben az Oracle-ös adatbázisunk kapcsolatát felvenni. Az Add Connection ablakban a Data Sourcenál Oracle Database Serverhez kapcsolódjunk, lehetőségünk van kiválasztani a .NET Framework Oracle Providerét vagy az Oracle ODP.Net-et. A Data source name mezőben az adatbázisunk elérését kell megadnunk. Az azonosításhoz szükséges adatokat adjuk meg. Beléphetünk általános vagy adatbázis adminisztrátori jogokkal. Ezután tesztelhetjük a kapcsolatot, majd siker esetén el is tárolhatjuk.