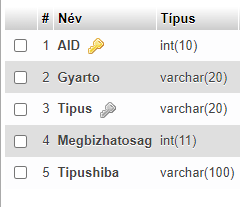
# CarScope-MyAdmin Dokumentáció

**A program célja**

A programot a projektünkben használt adatbázisban történő alapvető műveletek elvégzésére készítettük és használjuk. Segítségével SQL parancsok ismerete nélkül tud a felhasználó/admin adatokat megjeleníteni, módosítani, törölni, illetve bővíteni is képes az adatbázist új rekorddal. Ehhez az alkalmazás az Oracle MySql.Data csomagot használja.

**A program felépítése**

Az alkalmazás az adatbázisunk 3 táblájának kezelésére alkalmas. Minden táblához készíttetünk egy osztályt, amelyben megtalálhatóak az adott modelhez tartozó adatbáziskezelő függvények. A táblák oszlopait egy-egy privát tulajdonság jelöli:

class AutoModel

    {

        private int \_aId;

        public int aId

        {

            get { return \_aId; }

            set { \_aId = value; }

        }

        private string \_gyarto;

        public string gyarto

        {

            get { return \_gyarto; }

            set { \_gyarto = value; }

        }

        private string \_tipus;

        public string tipus

        {

            get { return \_tipus; }

            set { \_tipus = value; }

        }

        private int \_megbizhatosag;

        public int megbizhatosag

        {

            get { return \_megbizhatosag; }

            set { \_megbizhatosag = value; }

        }

        private string \_tipusHiba;

        public string tipusHiba

        {

            get { return \_tipusHiba; }

            set { \_tipusHiba = value; }

        }

Az adatbázisból MySqlDataReader segítségével kiolvassuk az adatokat, majd ezeket a tulajdonság nevének megfelelően tároljuk el. Valamennyi model osztály tartalmaz ezen felül négy függvényt, amely a négy Sql alapművelet (select, update, insert, delete) elvégzésére szolgál.

 public static ObservableCollection<AutoModel> select()

        {

            var lista = new ObservableCollection<AutoModel>();

            using (var con = new MySqlConnection(ConfigurationManager.ConnectionStrings["connectionString"].ConnectionString))

            {

                con.Open();

                var sql = "SELECT \* FROM auto";

                using (var cmd = new MySqlCommand(sql, con))

                {

                    using (var reader = cmd.ExecuteReader())

                    {

                        while (reader.Read())

                        {

                            lista.Add(new AutoModel(reader));

                        }

                    }

                }

            }

            return lista;

        }

A **select** függvény az adatbázishoz való kapcsolódás után soronként kiolvassa a benne található rekordokat, amiket példányosítva egy listában tárol el.

public static void update(int id, string gyarto, string tipus, int megbizhatosag, string tipushiba)

        {

using (var con = new MySqlConnection(ConfigurationManager.ConnectionStrings["connectionString"].ConnectionString))

{

con.Open();

 var sql = "UPDATE auto SET AID = @id, Gyarto = @gyarto, Tipus = @tipus, Megbizhatosag = @megbizhatosag, Tipushiba = @tipushiba WHERE AID = @id";

using (var cmd = new MySqlCommand(sql, con))

 {

  cmd.Parameters.AddWithValue("@id", id);

  cmd.Parameters.AddWithValue("@gyarto", gyarto);

  cmd.Parameters.AddWithValue("@tipus", tipus);

  cmd.Parameters.AddWithValue("@megbizhatosag", megbizhatosag);

  cmd.Parameters.AddWithValue("@tipushiba",tipushiba);

  cmd.ExecuteNonQuery();

  }

 }

}

Az **update** függvény a meghívásnál kapott id, jelenesetben autó azonosítójának sorában frissíti az értékeket.

public static void insert(string gyarto, string tipus, int megbizhatosag, string tipushiba)

        {

using (var con = new MySqlConnection(ConfigurationManager.ConnectionStrings["connectionString"].ConnectionString))

{

 con.Open();

var sql = "INSERT INTO `auto`(`Gyarto`,`Tipus`, `Megbizhatosag`," + " `Tipushiba`) VALUES (@gyarto, @tipus, @megbizhatosag, @tipushiba)";

using (var cmd = new MySqlCommand(sql, con))

{

  cmd.Parameters.AddWithValue("@gyarto", gyarto);

  cmd.Parameters.AddWithValue("@tipus", tipus);

  cmd.Parameters.AddWithValue("@megbizhatosag", megbizhatosag);

  cmd.Parameters.AddWithValue("@tipushiba", tipushiba);

  cmd.ExecuteNonQuery();

  }

 }

}

Az **insert** függvénnyel új rekordot tudunk felvinni az adatbázisba, a hozzáadott rekordok azonosítója automatikusan növekszik.

public static void delete(int id)

        {

            using (var con = new MySqlConnection(ConfigurationManager.ConnectionStrings["connectionString"].ConnectionString))

            {

                con.Open();

                var sql = "DELETE FROM `auto` WHERE AID = @id";

                using (var cmd = new MySqlCommand(sql, con))

                {

                    cmd.Parameters.AddWithValue("@id", id);

                    cmd.ExecuteNonQuery();

                }

            }

        }

A **delete** függvény értelemszerűen törli a kapott id-hoz tartozó rekordot az adatbázisból, és az azzal megegyező idegen kulccsal rendelkező rekordokat a többi táblából.

**Indítás után**

A program indulás után beolvassa az adatbázisban található adatokat, majd táblánként azokat egy külön listában tárolja el.

A képen asztal látható

Automatikusan generált leírás

Az ablak felső részében található egy táblázat, amelyben alapértelmezetten az autok lista elemeit láthatjuk. Az efölött található lenyíló listában választhatjuk ki a megjeleníteni kívánt táblát. Váltás esetén a táblázat adatforrása frissül, és az alatta található űrlap is megváltozik. Ehhez **if** függvényt használunk.

Az ablak tetején továbbá találunk egy keresésre szolgáló beviteli mezőt, amely a kiválasztott lista elemei közötti keresésre szolgál. Továbbá itt láthatjuk az aktuálisan megjelenített sorok számát is.

private void cb\_databases\_SelectionChanged(object sender, SelectionChangedEventArgs e)

        {

if (cb\_databases.SelectedItem.ToString()=="Autók")

{

              autok = AutoModel.select();

              DG\_adatok.ItemsSource = autok;

              TB\_searchbar.Text = "";

      SP\_infokInsert.Visibility = Visibility.Collapsed;

              SP\_gumikInsert.Visibility = Visibility.Collapsed;

              SP\_autokInsert.Visibility = Visibility.Visible;

}

else if (cb\_databases.SelectedItem.ToString()=="Gumiabroncsok")

{

 gumik = GumiModel.select();

 DG\_adatok.ItemsSource = gumik;

 TB\_searchbar.Text = "";

 SP\_infokInsert.Visibility = Visibility.Collapsed;

 SP\_autokInsert.Visibility = Visibility.Collapsed;

 SP\_gumikInsert.Visibility = Visibility.Visible;

}

else if (cb\_databases.SelectedItem.ToString()=="Info")

{

 infok = InfoModel.select();

 DG\_adatok.ItemsSource = infok;

 TB\_searchbar.Text = "";

 CB\_autoAzon.Items.Clear();

 foreach (var item in autok)

{

CB\_autoAzon.Items.Add(item.aId + ":" + " " + item.gyarto + " " + item.tipus);

}

 CB\_autoAzon.SelectedIndex = 0;

 SP\_autokInsert.Visibility = Visibility.Collapsed;

 SP\_gumikInsert.Visibility = Visibility.Collapsed;

 SP\_infokInsert.Visibility = Visibility.Visible;

}

LBL\_recordCount.Content = "Rekordok száma: " + DG\_adatok.Items.Count.ToString();

}

Táblaváltásnál a táblázat (**Datagrid DG\_adatok**) adatforrása a kiválasztott modelhez tartozó lista lesz, mely minden váltásnál frissíti a tartalmát. Törlődik a keresőmezőbe beírt szöveg, valamint a megfelelő űrlap kerül megjelenítésre. Az űrlapok elemeit Stackpanelbe csoportosítottuk.   
Végül frissül a megjelenített sorok számát mutató Label.

**Keresés**

else if (cb\_databases.SelectedItem.ToString()=="Gumiabroncsok")

{

if (TB\_searchbar.Text != "")

{

DG\_adatok.IsReadOnly = true;

var filteredList = gumik.Where(x => x.gyarto.ToLower().StartsWith(TB\_searchbar.Text.ToLower())

                    || x.evszak.ToLower().StartsWith(TB\_searchbar.Text.ToLower()));

DG\_adatok.ItemsSource = filteredList;

BTN\_Delete.IsEnabled = false;

BTN\_Save.IsEnabled = false;

}

else

{

DG\_adatok.IsReadOnly = false;

DG\_adatok.ItemsSource = gumik;

BTN\_Delete.IsEnabled = true;

BTN\_Save.IsEnabled = true;

}

}

Ha a keresőmezőbe szöveget gépelünk, eseménykezelő segítségével a program érzékel és meghatározza, hogy mely adatbázisban kell keresést végrehajtani, majd a megfelelő listából egy másik listába kigyűjti a beírt szöveggel kezdődő sorokat.

A képen asztal látható

Automatikusan generált leírásÍgy a beviteli mezőbe beírt szöveg alapján kereshetünk egy adott rekordot vagy rekordokat az adatbázisban:

Volkswagen típusú autók

- autót gyártó és típus szerint,

- autó infó-t rendszám és

alvázszám szerint,

- gumiabroncsot pedig gyártó és

évszak alapján.

A **Datagrid** adatforrása ezt követően a szűrt lista lesz, egészen addig, amíg ki nem töröljük a keresőmezőbe beírt szöveget.

**Módosítás és törlés**

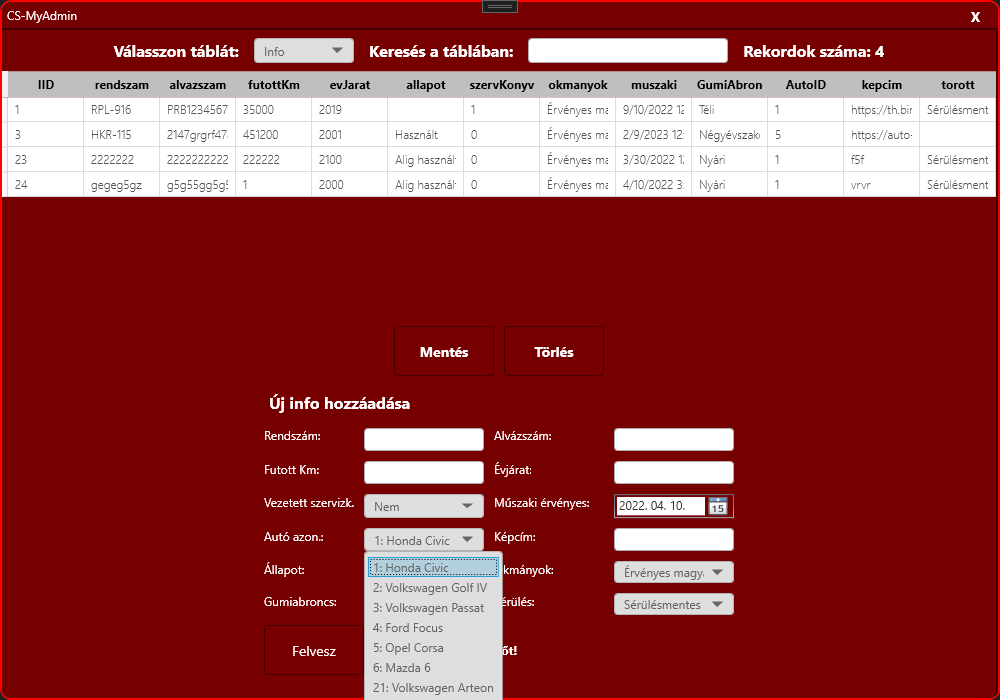
A Datagrid és az űrlap között található egy Mentés és egy Törlés gomb. Ha egy mezőre duplán kattintunk, tartalma módosíthatóvá válik. A Mentés gombra kattintva az adatok frissülnek a listában, és ezáltal a táblázatban is. A Törlés gombbal törölhetjük a kiválasztott rekordot az adatbázisból.  
Annak eldöntésére, hogy mely táblában hajtódjon végre a módosítás/törlés, az adatforrás kiválasztását végzővel megegyező szerkezetű if függvényt használunk.

**Új adat felvitele**

Az űrlap Textboxait kötelező kitölteni, a lenyíló listák alapértelmezetten a 0. index-szel rendelkező opciót jelenítik meg, a dátumválasztó alapértelmezett értéke a mindenkori mai dátum. Ha egy mezőt üresen hagyunk, a Felvesz gombra kattintás után egy hibaüzenet jelenik meg, és nem történik meg az új adat felvitele. A rendszám, alvázszám, Futott km és évjárat mezőkbe csak számot lehet begépelni, a rendszámnak 7-8, az alvázszámnak 17, az évjáratnak 4 karakter hosszúnak kell lennie. Ha a felhasználó úgy vált mezőt, hogy valamely feltétel nem teljesül, annak megfelelő hibaüzenet kerül megjelenítésre, és a hibás beviteli mező értéke törlődik, megakadályozva a felvitelt.

A képen szöveg látható

Automatikusan generált leírás

Új info hozzáadásánál az Autó azon. lenyíló listában ki kell választanunk, hogy milyen típusú autóhoz szeretnénk adatokat felvinni. Ehhez a program betölti az adatbázisban található autók id-ját, gyártóját ás típusát, előbbit ’:’- al elválasztva utóbbi kettőtől. Inzertáláskor a kiválasztott listaelemből egy kételemű tömböt készítünk, melyben a sor ’:’ előtti tartalma, azaz az id kerül elválasztásra a gyártótól és a típustól, ez lesz a tömb (splittedText) 0. eleme, amit int típussá kovertálva fel tudunk vinni az adatbázisba idegen kulcsként.

Fent: Kiválaszthatjuk, melyik autótípushoz szeretnénk új járműadatokat hozzáadni. Lent: A kiválasztott listaelem azonosítója bekerül az adatbázisba(pirossal)

string[] splittedText = CB\_autoAzon.SelectedItem.ToString().Split(':');

if (TB\_Rendszam.Text !="" && TB\_Alvazszam.Text !="" && TB\_FutottKm.Text != "")

{

InfoModel.insert(TB\_Rendszam.Text, TB\_Alvazszam.Text, Convert.ToInt32(TB\_FutottKm.Text), int.Parse(TB\_Evjarat.Text), CB\_allapot.SelectedItem.ToString(), CB\_szervkönyv.SelectedIndex, CB\_okmanyok.SelectedItem.ToString(), DP\_muszaki.SelectedDate.Value, CB\_GumiInfo.SelectedItem.ToString(),

Convert.ToInt32(splittedText[0]),TB\_kepCim.Text,

CB\_Torott.SelectedItem.ToString());

infok = InfoModel.select();

DG\_adatok.ItemsSource = infok;

}

**Az űrlapok visszaállítása**

Inzertálás után a mezők alapértelmezett értékre való visszaállításához a **ResetSp** függvényt használjuk:

Mivel az űrlapok egy Stackpanelből és az abban levő egymásba ágyazott Stackpanelekből állnak, ezért legkönnyebben függvény segítségével érhetjük el a belső Stackpanelek objektumait. A függvénynek meghíváskor megadjuk, melyik űrlapban végezze el a visszaállítást. Megkeresi az abban található Stackpaneleket, amiket egy listába gyűjt egy foreach függvény segítségével. A kigyüjtött Stackpanelekben megkeressük a gyerekobjektumokat, amik a mi esetünkben már azok a vízszintes orientáltságú stackpanelek lesznek, amikben a beviteli mezők találhatók. Foreach használatával megkeresünk minden Textbox, Combobox, és Datepicker típusú objektumot, és az indításkori állapotra állítjuk vissza őket.

static void ResetSP(StackPanel Stackname)

{

 List<Object> spanels = new List<Object>();

 foreach (Object vizsgaltItem in Stackname.Children)

  {

    if (vizsgaltItem.GetType() == typeof(StackPanel))

     {

       spanels.Add(vizsgaltItem);

     }

}

      foreach (StackPanel spanel in spanels)

       {

        foreach (Control item in spanel.Children)

         {

           if (item.GetType() == typeof(TextBox))

            {

               ((TextBox)item).Text = string.Empty;

            }

              else if (item.GetType() == typeof(ComboBox))

                {

                     ((ComboBox)item).SelectedIndex = 0;

                }

  else if (item.GetType() == typeof(DatePicker))

   {

     ((DatePicker)item).SelectedDate = DateTime.Now;

   }

  }

 }

}