

AtomerWindowsForms

AtomerWindowsForm är en kodbas för studenter att koppla från databasen till uppvisning på AtomerForm. Koden finns på: <https://github.com/robinos/AtomerWindowsForms> men även finns beskriven här i textform. Om man tar ner det som .zip fil, kom ihåg att ta bort ”-master” delen av mappnamnet. Det går att använda WPF istället även om kodexemplet är i Windows Forms. Om man har problem med databasen, försök att öppna den en gång först innan körning. I värsta fall kan man använda hårdkodad värden som man skapar själv – det bättre än att man blir fast.

Klasser som ges färdigt:

Databas - databas kopplingen

Atom - datarad behållare

AtomerProgram - kopplar databasen till formen

Vad man måste gör:

I **AtomerForm** borde man koppla rader i Dictionary<int,Atom> Atomer till:

En combobox som innehåller alla atom Namn

Labels för Förkortning, Atomnummer, Atomvikt, Kokpunkt och Smältpunkt som ändrar för att visa information för atomen som är vald i comboboxen.

Vad man kan göra för lite extra utmaning (om man vill):

Koppla mot en MySQL databas som innehåller tabell tblAtomer istället och lägg till inloggning.

Man får ändra det färdiga koden så mycket man vill.

dbAtomer.mdf – tblAtomer.sql

dbo.tblAtomer [Design] ➔ ✕				
Update		Script File: dbo.tblAtomer.sql		
	Name	Data Type	Allow Nulls	Default
	fldAtomnummer	int	<input type="checkbox"/>	
	fldNamn	varchar(20)	<input checked="" type="checkbox"/>	
	fldAvikt	decimal(6,1)	<input checked="" type="checkbox"/>	
	fldForkortn	varchar(3)	<input checked="" type="checkbox"/>	
	fldKokpkt	decimal(6,2)	<input checked="" type="checkbox"/>	
	fldSmaltpkt	decimal(6,2)	<input checked="" type="checkbox"/>	
			<input type="checkbox"/>	
Design ↑↓ T-SQL				
<pre>CREATE TABLE [dbo].[tblAtomer] ([fldAtomnummer] INT NOT NULL, [fldNamn] VARCHAR (20) NULL, [fldAvikt] DECIMAL (6, 1) NULL, [fldForkortn] VARCHAR (3) NULL, [fldKokpkt] DECIMAL (6, 2) NULL, [fldSmaltpkt] DECIMAL (6, 2) NULL,</pre>				

Kod att ändra:

AtomerForm.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace AtomerWindowsForms
{
    /// <summary>
    /// AtomForm klassen visar upp data från tblAtomer med namn i en combobox som styr
    /// vad som stor på labels för de andra attribut.
    /// </summary>
    public partial class AtomerForm : Form
    {
        /// <summary>
        /// Constructor för AtomerForm tar emot en instans av atomerProgram. Man har
        /// tillgång till en instans av databasklassen som egenskap som i sitt tur
        /// har tillgång till egenskapen Atomer som är en Dictionary(int,Atom) av
        /// alla rader från databasen.
        /// </summary>
        /// <param name="atomerProgram"></param>
        public AtomerForm(AtomerProgram atomerProgram)
        {
            //Initialisera AtomerForm
            InitializeComponent();

            //Test: visar att allt fungerar som det ska och data läsas från tabellen
            MessageBox.Show(atomerProgram.AtomDatabas.Atomer[1].Namn + " hittades!");
        }
    }
}
```

'Färdig' kod:

Atom.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace AtomerWindowsForms
{
    /// <summary>
    /// Atom är en behållare för en rad data från tblAtom.
    /// </summary>
    public class Atom
    {
        //instansvariabel
        int atomnummer;
        string namn;
        string förkortning;
        decimal atomvikt;
        decimal kokpunkt;
        decimal smältpunkt;

        //egenskaper
        public int Atomnummer { get { return atomnummer; } set { atomnummer = value; } }
        public string Namn { get { return namn; } set { namn = value; } }
        public string Förkortning { get { return förkortning; } set { förkortning = value; } }
        public decimal Atomvikt { get { return atomvikt; } set { atomvikt = value; } }
        public decimal Kokpunkt { get { return kokpunkt; } set { kokpunkt = value; } }
        public decimal Smältpunkt { get { return smältpunkt; } set { smältpunkt = value; } }
    }
}
```

AtomerProgram.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace AtomerWindowsForms
{
    /// <summary>
    /// AtomerProgram innehåller en instans av Databas klassen och en Main metod som startar
    /// AtomerForm om det gick bra att läsa från databas.
    /// </summary>
    public class AtomerProgram
    {
        //instansvariabler
        private Databas databas = new Databas();

        //egenskaper
        public Databas AtomDatabas { get { return databas; } }

        /// <summary>
        /// Main metoden gör en instans av AtomerProgram och försöker anropa LäsaProdukter
        /// från Databas klassen. Om det lyckas köra AtomerForm med en referens till
        /// AtomerProgram för att nå AtomDatabas egenskapen och vad som helst annat man vill.
        /// </summary>
        [STAThread]
        static void Main()
        {
            //instans av AtomerProgram
            AtomerProgram atomerProgram = new AtomerProgram();

            Application.EnableVisualStyles();
            Application.SetCompatibleTextRenderingDefault(false);

            //Om det gick bra att läsa från databasen, starta formen
            if (atomerProgram.databas.LäsaAtomer() == true)
            {
                Application.Run(new AtomerForm(atomerProgram));
            }
            //Annars visar en felmedellande
            else
                MessageBox.Show("Kopplingen till databasen misslyckades.");
        }
    }
}
```

Databas.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.IO;
using System.Reflection; //IO och Reflection används för att hämta applikationsmapp
using System.Windows.Forms; //för MessageBox
using System.Data.SqlClient; //för Microsoft SQL databas
using System.Data; //för DataSet, DataRow, etc.

namespace AtomerWindowsForms
{
    /// <summary>
    /// Databas klassen kopplar upp sig till databasen och läser rader från tabellen tblAtomer
    /// till Dictionary(int,Atom). Man behöver inte ändra något här om man inte vill - det är
    /// bara en uppvisning på hur man kan nå databasen utan det inbyggda systemet i Visual Studio.
    ///
    /// Vill man använda MySQL versionen av tblAtomer istället finns där anvisningar från sidan
    /// 24-28 i häftet från Databashantering. Mest viktigt är att man använda
    /// using MySql.Data.MySqlClient,
    /// MySqlConnection,
    /// en annan kopplingssträng,
    /// och en MySqlDataAdapter
    /// </summary>
    public class Databas
    {
        private string kopplingssträngen = @"Data Source=(LocalDB)\v11.0;"
            + "AttachDbFilename=" + Path.GetDirectoryName(Assembly.GetEntryAssembly().Location)
            + "\\dbAtomer.mdf;" + "Integrated Security=True;";

        private Dictionary<int, Atom> atomer;
        private SqlConnection kopplingen;

        /// <summary>
        /// Constructor för Databas
        /// </summary>
        public Databas()
        {
            //initialisera Dictionary atomer
            this.atomer = new Dictionary<int, Atom>();
            //initialisera kopplingen till databasen (öppnas inte än)
            kopplingen = new SqlConnection(kopplingssträngen);
        }

        //get till atomer
        public Dictionary<int, Atom> Atomer { get { return atomer; } }

        /// <summary>
        /// Öppnar kopplingen till databasen och returnerar sann om det
        /// lyckades och annars visar en felmeddelande och returnerar falsk.
        /// </summary>
        /// <returns>sann om det lyckades, annars falsk</returns>
        private bool ÖppnaKopplingen()
        {
            try
            {
```

```

        //Öppna kopplingen och skickar tillbaka sann
        kopplingen.Open();
        return true;
    }
    catch (SqlException ex)
    {
        //Öppnades inte. Skickar tillbaka falsk
        MessageBox.Show("Kan inte koppla till databasen. Kontakt administratören. " +
ex.Message);

        return false;
    }
}

/// <summary>
/// Stänger kopplingen till databasen och returnerar sann om det
/// lyckades och annars visar en felmedellande och returnerar falsk.
/// </summary>
/// <returns>sann om det lyckades, annars falsk</returns>
private bool StängKopplingen()
{
    try
    {
        //Stänger koppling och skickar tillbaka sann
        kopplingen.Close();
        return true;
    }
    catch (SqlException ex)
    {
        //Stängdes inte. Skickar tillbaka falsk
        MessageBox.Show("Databasen stängdes inte ner! " + ex.Message);
        return false;
    }
}

/// <summary>
/// Läser in rader från databasen och förvandla de till Atom objekt som sedan
/// sätts i Dictionary atomer med Atomnummer som nyckel till objektet.
/// Returnerar sann om det lyckades att öppna kopplingen, om tabellen existerar,
/// om tabellen har rader, och om tabeller har en kolumn fldAtomnummer. Annars
/// returneras falsk.
/// </summary>
/// <returns>sann om det lyckades, annars falsk</returns>
public bool LäsAtomer()
{
    //Öppna databasen
    bool lyckades = ÖppnaKopplingen();

    //Om man inte kunde öppna databasen, slutar och returnerar falsk
    if (!lyckades) return false;

    //DataSet är en behållare/mellansteg för inläst databas-data (kan
    //inhålla flera tabeller)
    DataSet dataSet = new DataSet();

    //En ny DataAdapter skapas med ett select sql command redan inbyggd
    //(DataAdapter används sedan för att fylla en DataSet)
    SqlDataAdapter dataAdapter = new SqlDataAdapter("SELECT * FROM tblAtomer",
kopplingen);

```

```

//Fyll metoden på DataAdapter används för att faktiskt utföra fyllning
//av Datasetet ds från tabellen Produkter
dataAdapter.Fill(dataSet, "tblAtomer");

//Om där finns inga tabeller, returnerar falsk
if (dataSet.Tables.Count == 0)
    return false;

//Om där finns inga rader i första tabellen, returnera falsk
var table = dataSet.Tables[0];
if (table.Rows.Count == 0)
    return false;

//Om där finns ingen ID kolumnen, returnera falsk
if (!table.Columns.Contains("fldAtomnummer"))
    return false;

//Om där finns ingenting i ID kolumnen, returnera falsk
var row = dataSet.Tables[0].Rows[0];
if (row.IsNull("fldAtomnummer"))
    return false;

//Temporär Atom variabel
Atom atomTemp;

//Loopa genom varje rad i tblAtomer tabellen (Tables[0] för att enligt
//SELECT frågan är tblAtomer första (och enda) tabellen i DataSet ds)
foreach (DataRow dataRow in dataSet.Tables[0].Rows)
{
    //Läser värdena från en rad till Atom objektet
    atomTemp = new Atom();
    atomTemp.Atomnummer = int.Parse(dataRow["fldAtomnummer"].ToString());
    atomTemp.Namn = dataRow["fldNamn"].ToString();
    atomTemp.Förkortning = dataRow["fldForkortn"].ToString();
    atomTemp.Atomvikt = decimal.Parse(dataRow["fldAvikt"].ToString());
    atomTemp.Kokpunkt = decimal.Parse(dataRow["fldKokpkt"].ToString());
    atomTemp.Smältpunkt = decimal.Parse(dataRow["fldSmaltpkt"].ToString());

    //Sätt atomTemp objekt i Dictionary atomer med Atomnummer som nyckel
    //(om en Atomnummer redan finns i atomer, uppdatera bara värden)
    if (!atomer.ContainsKey(atomTemp.Atomnummer))
        atomer.Add(atomTemp.Atomnummer, atomTemp);
    else
        atomer[atomTemp.Atomnummer] = atomTemp;
}

//Stäng databasen
StängKopplingen();

return lyckades;
}
}
}

```